

Chapter 1. Is There a Security Problem in Computing?

1.0 Objectives

1.1 1.1. What Does "Secure" Mean?

1.2 Attacks

1.2.1 Vulnerabilities, Threats, Attacks, and Controls

1.3 The Meaning of Computer Security

1.4 Computer Criminals

1.5 Methods of Defense

1.6 Review Question

1.7 References

DRAFT

1.1.What Does "Secure" Mean?

How we can protect are valuable assets? One of the safest place our money in bank in term of cash and gold jewelry. In old day most of the money in banks are kept as cash instead of check, is not easily traceable. Bank would protect our assets from robbery by keeping the asset in lockers and provides security by keeping guards outside. In olden days , communication and transportation was primitive enough , the legal administrative officer would informed the police about the robbery in the bank , the security personals would reach the site very late that time already the criminal escape from the bank . It would be very difficult to trace the criminal .That time the bank robbery is profitable business. So protecting our assets was very difficult and not always very effective.

But today, assets protection becomes very easier, with many effective technique against the criminals, by means of putting sophisticated alarm, CCTV place in the bank premises by tracking at activity of peoples inside and outside of the banks. The technique of the criminal investigation have become more effective in terms by taking the finger print , DNA , retinal pattern , iris recognition , voice of the person and person can be identify above the mentioned properties . The assets would be store much safer form for an example, Many banks now contains less cash than some retail stores because much of the bank's business is done though the check ,electronic transfers, credit cards , debit cards and so on. In Banks sites, large amount of hard cash and currency are stored in many layer of security levels: several physical layers, many complex locks are implement and multiple system party are required among the several peoples to allow to access the assets. There are significant improvements in transaction and communications mean that the police would reach the crime site in a minutes. Sophisticated alarm and CCTV would dispatch the alert to other officers in seconds about the suspect to watch for. From the criminal point of views, it very difficult for robbers to commit crime in the banks.

Protecting Valuables

This books is not dedicated how to protect the money or gold jewelry, it dedicated how to protect the computer resources. Form analysis of banks, we have learned some basic principle of protection. In other words, when we learn about the protecting the valuable information, we learn lot about the protecting the other valuable resources. The Table 1.1 give the difference between how people protect the computing resource and how bank protect the money.

Table 1-1. Protecting Money vs. Protecting Information.

Characteristic	Bank Protecting Money	People Protecting Information
Size and portability	Banks sites stores money are large , not all portable, Building are required to store, Guards are requires, vaults , Many levels of physical security to protect the money	Items storing valuable assets are small and portable. The physical device are required to protect the valuable information are so small, it contains of thousand rupees.
Ability to avoid physical contact	Difficult, when banks deals with physical money or currency, a criminal can demand the money and steal the money physical from the bank's premises.	Simple. When the information handle electronically, physical contact are not necessary. Instead banks handled the money electronically , transaction are done without any physical contact .Here money can be transferred through the computers, mobile , telephone and email
Value of assets	Very High.	Variable., It can be from high top low or vice versa, some sensitive information like medical history, tax payment, education background are kept confidential , some of the information like troop movement , sales strategies are very sensitive information, still other information like phone number and address may be have no

		consequence and can access by other means
--	--	---

You can develop an understanding of the basic problems underlying computer security and the methods available to deal with them.

In particular, we do the following:

- examine the risks of security in computing
- consider available countermeasures or controls
- stimulate thought about uncovered vulnerabilities
- identify areas where more work is needed

In this chapter, we will be examining different kinds of vulnerabilities computing systems are more prone to systems. What are reason for vulnerabilities for exploiting: the different kinds of attacks that are possible to happen in the system the kinds of people are involved or contribute in the security? Finally, we introduce how to prevent possible attacks on systems

Characteristics of Computer Intrusion

Criminal can target to any part of a computing system to commit the crime. When we said about **computing system**, it's mean a collection of hardware, software, storage media, data, and people .the following resources are used by the organization for computing the task. Sometimes, the organization is least bother about the computing resource and consider the resources are not valuable to an outsider, they make mistake for not consider valuable assets. For an example, people consider most valuable property in a bank is the cash, gold, or silver in the bank's vault. But the people forget the most valuable is the customer information kept in the bank's computer. The bank's customer's information is stored on paper, recorded on a storage devices like tape drives, hard disk or information may resident in memory, or transmitted over telephone lines or satellite links, and this information can be used many way for making a money illicitly. A competitor bank can use this information to steal the details of the customers or even to disrupt service and discredit the bank. An anonymous user or hacker could move money from one account to another bank account without the permission of owner. A group of impostor could contact large depositors and convince them to invest in fraudulent schemes. The variety of targets and attacks makes computer security very difficult

Any system is consider more vulnerable than it has weakest point, for example a robber intention was to steal something from the house if a window gives him as easier access instead of two thick metal door for penetration . We can codify this idea as one of the principles of computer security.

The principles of computer system should be consider by security specialists by means different possible ways of penetration in to the system. Whenever the security parameters change, according to the policy of an organization the penetration analysis must repeatedly scan the vulnerability in

the systems. Sometimes, the People underestimate the determination or creativity of attackers. It has to remember that computer security is a key roles for the defending team only its means , The attackers can (and will) use any technique for penetrating into the system .The security specialists has to think out of box , how to prevent the system form the attackers .

1.2 Attacks

When we test any computer system in the organization, the main jobs of ours to imagine that how the system could malfunction. Then ours responsibility to improve the system's design so the system can withstand any of the problems we have identified. In the same way, computer security specialist analyze a system from a security perspective, thinking about different ways in which the system's security can malfunction and diminish the value of its assets.

1.2.1 Vulnerabilities, Threats, Attacks, and Controls

A computer system has three separate valuable components: **hardware**, **software**, and **data**. Each valuable assets offers different values to different members of the community which are affected by system. The security analyst has to do different ways of brainstorming about the system or its information can leads to some kind of loss or harm to the system. For example, security analyst has to understand what can be data format or what kind of data contents should be protected in different way. Security analyst want the system secure such a way there should sure be data should not disclosed to unauthorized parties. He has to ensure that the data should be modified in illegitimate ways. At the same time, he must ensure that legitimate users have access to the data. In this way, we can identify weaknesses in the system.

What is a vulnerability?

According to the Definition “the quality or state of being exposed to the possibility of being attacked or harmed, either physically or emotionally.”

In the security term, a **vulnerability** is a weakness or flaw in the security system, for example, while in designing, procedure, or implementation of any system in term of application that might be exploited to cause loss or harm. For instance, a particular system may be vulnerable to people for accessing an unauthorized data, unauthorized people would manipulate the data because the system is failed to the verification of his / her identity and allow them to access unauthorized data.

What is threat?

According to the Definition “a statement of an intention to inflict pain, injury, damage, or other hostile action on someone in retribution for something done or not done”.

A threat, in the context of computer security, refers to anything that has the potential to cause serious harm to a computer system. Let see the difference between a threat and a vulnerability, consider the illustration in Figure 1-1.

In the figure 1.1 , As a wall is holding a water its back ,so the water on the left side of the wall is a threat to the man stand on the right side of the wall, If the water could rise above the wall level, it could overflowing onto the man, or it could stay behind the wall, due to amount water pressure applied on the wall which cause the wall to be collapse. So it leads to threat of harm is for the man to get wet, get hurt, or be drowned. For now, the wall is intact, so the threat to the man is unrealized.

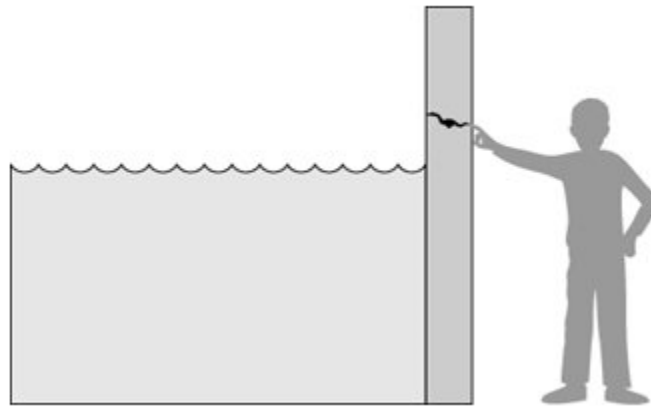


Figure 1.1. Threats, Controls, and Vulnerabilities.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

However, we can see a small crack appear in the wall, it leads to a vulnerability that would threatens for security of man. If the water level rises beyond the level of the crack, it could exploit the vulnerability and cause a harm to man.

There are ample of threats to a computer system, including human- intervention and computer-intervention. We all are experience human errors while designing an application, flaws in hardware designing, and software failures. But there is an also natural disasters as threats, that it can bring a system down when the computer room is flooded or the data center collapses from an earthquake.

A human who cause an exploits can leads to a vulnerability perpetrates an **attack** on the system. An attacker might launched an attack from another system by sending an overwhelming set of messages to another system, it lead to virtually shutting down system's ability to function. Such attack known as denial-of-service attacks, which flood servers with more messages so the system will not handle the message so the system could function properly.

How do we solve the problems of vulnerability and threat? By means of control as a protective measure. A control is an action, device, procedure, or technique that removes or reduces a vulnerability in the system

In Figure 1.1, the man is placed his finger in the hole, it could control the threat of water being leaks until he could find permanent solution to the problem. In general, we can describe the relationship among threats, controls, and vulnerabilities in this way:

A threat is blocked by control of a vulnerability.

To invent the plan control, we must know as much as possible threat in the systems, we can view four way of threat in the system: interception, interruption, modification, and fabrication. Each threat could exploits vulnerabilities of the assets in computing systems; the threats are illustrated in Figure 1.2.

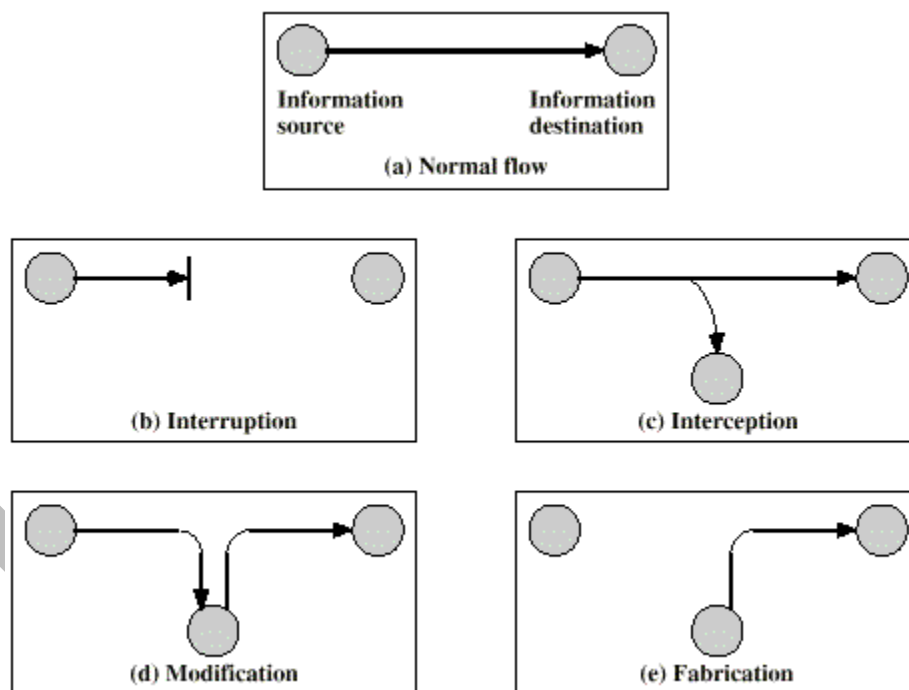


Figure 1.2 Types of System Security Threats.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

- a) Normal flow : An entity or person is send an information from source systems to destination system
- b) An interruption, an asset of the system becomes lost, unavailable, or unusable. An example is malicious destruction of a hardware device, erasure of a program or data file, or malfunction of an operating system file manager so that it cannot find a particular disk file.
- c) Interception: An interception means that some unauthorized party has gained access to an asset. The outside users can be a person, a program, or a computing system. Examples of

this type of failure are illicit copying of program or data files, or wiretapping to obtain data in a network.

- d) **Modification:** some unauthorized user not only access the unauthorized data, but also modified the data. For example hacker can modified the data value in the databases.it may alter the program computation part, it additionally perform the computation apart. The data can electronically modified, some hardware can also be modified.
- e) **Fabrication:** An unauthorized party inserts counterfeit objects into the system. This is an attack on the authenticity. Examples include the insertion of spurious messages in a network or the addition of records to a file

Method, Opportunity, and Motive

A malicious attacker must have three things:

- **Method:** the skills, knowledge, tools, and other things with which to be able to pull off the attack
- **Opportunity:** the time and access to accomplish the attack
- **Motive:** a reason to want to perform this attack against this system

1.3 The Meaning of Computer Security

The main aim of computer security is to provide many ways to prevent the flaw, weakness from being exploited. In order to understand what precaution measure has to take at most it make the sense to say whether a system is "secure."

Security Goals

In our daily lives, we the word "security" in many ways. A "security system" key word came in the picture when we want to protect our house from intruders, if intruders tries to get in our house, we warning our neighbors or try to contact the police nearby in our locality. "Financial security" is the word say our involvement in different set of investments that are adequately funded; so we further hope our investments will grow in value over period of time so that we have enough money to survive later in life. Then we speak of children's "physical security," hoping that our children are safe from potential harm. Just as above terms is use for the specific domain, so we too does the phrase "computer security."

When we say about the term as "computer security", it mean that we are addressing three important principle of any computer-related security i.e. **Confidentiality, integrity, and availability.**

- The term "Confidentiality" mean that to ensure about computer-related assets are accessed only by authorized users. That is, only those person or an entity should have access to something will have a right to access the things. The term "access," are not meant only reading but also for viewing, printing, or simply knowing whether a particular assets is exist or not. Confidentiality is sometimes called **secrecy** or **privacy**.

- The term “Integrity” means that particular assets can be modified or tampered only by authorized user only in authorized ways. The context regarding modification includes writing, changing, changing status, deleting, and creating of assets.
- The term “Availability” means that assets are accessible to only by an authorized users at particular times. In other words, if some person or system have an access to legitimate systems or a set of objects, then access should not be prevented for the authorized users. For this reason, availability is sometimes known by its opposite, denial of service.

Security in computing addresses the three challenges. One of the challenges is to build a secure system to finding the right balance for achieving the goals, which often conflict. For example, it is very easy to protect a particular object's confidentiality in a secure system simply by not allowing everyone from reading that object. However, this system is unsecure, because it does not meet the standard requirement of availability for proper accessing the object. So there should be a balance between confidentiality and availability.

But balance is not meant at all ,In fact, these three characteristics can be independent from each other, it can be overlap (as shown in Figure 1.3), and can be mutually exclusive even . For example, we have seen the above example stating about strong protection of confidentiality which can severely restrict availability for the system. Let us we goes in depth of each of the three qualities.

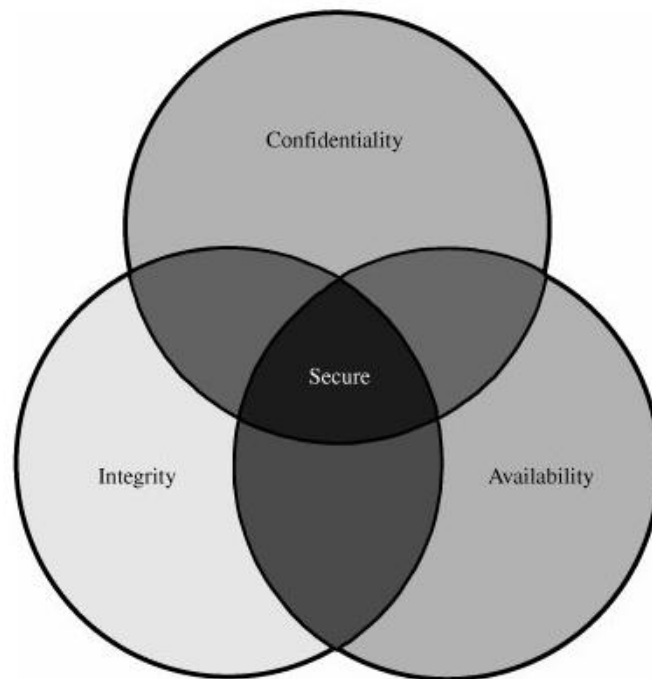


Figure 1.3. Relationship between Confidentiality, Integrity, and Availability.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Confidentiality

You may find the concept of confidentiality is to be straightforward: its say that only authorized people or systems can access protected data. However, have seen later in the chapter, that it is very difficult to ensure about the confidentiality. For example, it is very difficult to decide who is authorized person or system to determine which people or systems are authorized to access the current system? By "accessing" it difficult to determine whether an authorized user have access a single bit or the whole collection or pieces of data out of context. It also difficult to determine can someone who is authorized to disclose those data to other parties?

Confidentiality is roughly equivalent to privacy. Measures undertaken to ensure confidentiality are designed to prevent sensitive information from reaching the wrong people, while making sure that the right people can in fact get it: Access must be restricted to those authorized to view the data in question. It is common, as well, for data to be categorized according to the amount and type of damage that could be done should it fall into unintended hands. More or less stringent measures can then be implemented according to those categories.

Integrity

The Integrity word say that to ensure that data should be correct or accurate and it should not be change during the transmission. One type of security attack could be interceptor intercept some important data and make changes to it before sending it on to the intended receiver.

Availability

Availability on of the important principle in security, it ensure that the information should be readily available to the authorized user all the times. Some types of security attack could deny access to the appropriate user For example, by breaking the web site for a particular search engine, a rival may become more popular.

Vulnerabilities

When we decide to test a system, we usually try to find out how the system can fail; we then find out the different ways in which the requirements, design, or code can leads to failure of the systems. In the same way, when we prepare to specify, design, code, or test a secure system, we try to find out different types of vulnerabilities that would prevent us from reaching one or more of our three security goals.

It could sometime find that it is easier to consider that vulnerabilities could apply to all three broad categories of system resources (hardware, software, and data), rather than to start with the security goals themselves. Figure 1.4 shows the types of vulnerabilities that could apply to the assets of hardware, software, and data. These three assets and the connections among them are all potential security weak points. Let us look in turn at the vulnerabilities of each asset.

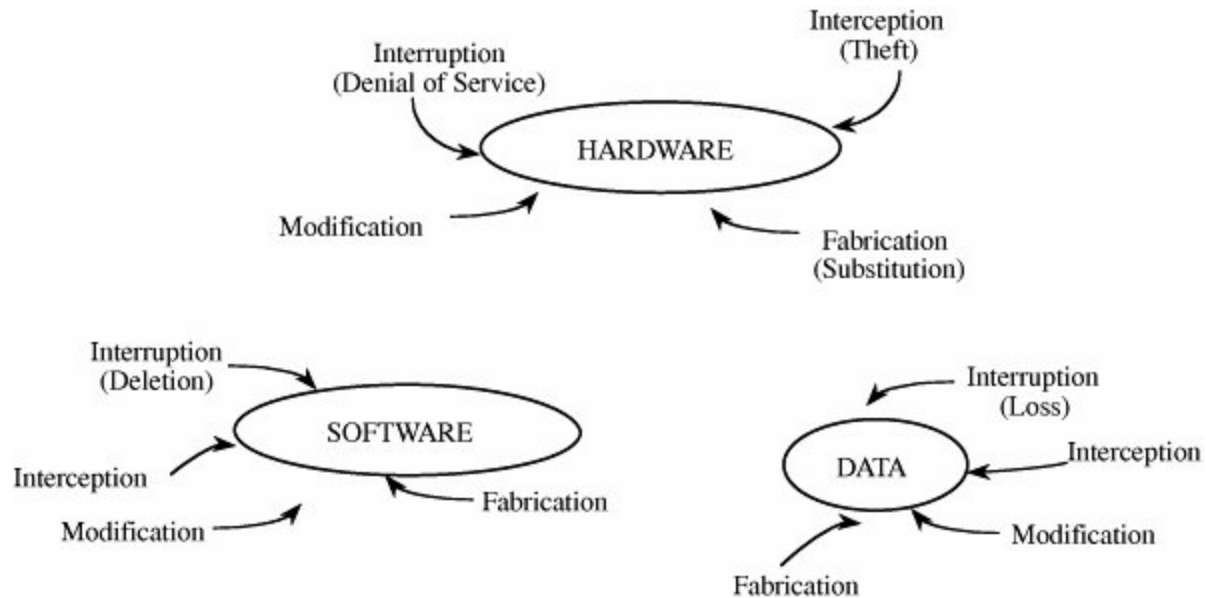


Figure 1.4. Vulnerabilities of Computing Systems.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Hardware Vulnerabilities

A hardware vulnerability is a flaw in a computer system that enables attacker attack the system hardware through remote or physical access.

The attack could be adding device, removing the devices, changing the devices, it could intercept the traffic to them, flooding the traffic until the functionality of the system become no longer.

There are other means of attack to the computer hardware physically. Computers have been drenched with water, burned, frozen, gassed, and electrocuted with power surges. People cloud spilled soft drinks, corn chips, ketchup, beer, and many other kinds of food on computing devices. Mice have chewed through cables. Particles of dust, and especially ash in cigarette smoke, have threatened precisely engineered moving parts. Computers have been kicked, slapped, bumped, jarred, and punched. Although such attacks might be intentional, most are not; this abuse might be considered "involuntary machine slaughter": accidental acts not intended to do serious damage to the hardware involved.

Another type of hardware vulnerability is an unexpected flaw in operation that allows attackers to gain control of a system by elevating privileges or executing code. These vulnerabilities can sometimes be exploited remotely, rather than requiring physical access.

Software Vulnerabilities

A software vulnerability can be seen as a flaw, weakness or even an error in the system that can be exploited by an attacker in order to alter the normal behavior of the system

A vulnerable software system can be exploited by attackers and the system could be compromised, the attacker might take control of the system to damage it, to launch new attacks or obtain some privileged information that he can use for his own benefit

A vulnerability in IIS, detailed in Microsoft Security Bulletin MS01-033, is one of the most exploited Windows vulnerabilities ever. A large number of network worms have been written over the years to exploit this vulnerability, including 'CodeRed'. CodeRed was first detected on July 17th 2001, and is believed to have infected over 300,000 targets. It disrupted a large number of businesses, and caused huge financial losses around the world. Although Microsoft issued a patch for the vulnerability along with the MS01-033 security bulletin, some versions of the CodeRed worm are still spreading throughout the Internet.

Software Deletion

Software is can easy to be delete. Each of us in our careers, accidentally erased a file or saved a bad copy of a program, destroying a good previous copy. So software's becomes high value to a commercial computing center, so accessing to software is usually carefully handled or controlled through a process called configuration management so that software cannot be deleted, destroyed, or replaced accidentally. Configuration management uses several techniques to ensure that each version of software or release retains its integrity. Whenever the new software is released the configuration management thoroughly tested to verify that the improvements work correctly without degrading the functionality and performance of other functions and services, then the old version or release can be replaced with a newer version only .

Software Modification

Software is can become an exploitable vulnerable to system when there is modifications in the software which cause it to fail the systems or it cause to perform an unintended task. As, a software is become more susceptible to one errors, it is quite easy to modify to it. Even changing a bit or two in software can convert a working program into a failing one. It depend upon on which bit was changed, the program may crash when it begins or it may execute for some time before it falters.

Even more change in the software can leads to an error in the working of the software. In most of the time, program works well but fails in specialized circumstances. For instance, the software can be maliciously modified so that the system could failed when certain conditions are met or when a certain date or time is reached, such delay in the effect , such a program is called as a **logic bomb**. For example, an angry employee may modify a crucial program so that it accesses the system date and halts abruptly after August 1. The employee might quit on June 1 and plan to be at a new job miles away by August

Other categories of software modification include

Trojan horse: a Trojan horse is a program that appears harmless, but is, in fact, malicious. Unexpected changes to computer settings and unusual activity, even when the computer should be idle, are strong indications that a Trojan is residing on a computer.

Virus: A computer virus is a malicious program that self-replicates by copying itself to another program. In other words, the computer virus spreads by itself into other executable code or documents. The purpose of creating a computer virus is to infect vulnerable systems, gain admin control and steal user sensitive data.

Trapdoor: A computer trapdoor, also known as a back door, provides a secret -- or at least undocumented -- method of gaining access to an application, operating system or online service.

Information leaks in a program: code that makes information accessible to unauthorized people or program

Software theft

Software theft means the unauthorized or illegal copying, sharing or usage of copyright-protected software programs. Software theft may be carried out by individuals, groups or, in some cases, organizations who then distribute the unauthorized software copies to users.

Software theft is committed when someone performs any of the following:

- Steals software media
- Deliberately erases programs
- Illegally copies or distributes a program
- Registers or activates a software program illegal.

Data Vulnerabilities

Security data, especially vulnerability data, have many concepts that translate nicely from the software quality realm. Vulnerabilities can be tracked in the same way as bugs, e.g., using modern issue tracking systems. Vulnerabilities manifest themselves as design flaws or coding mistakes in the system, much like bugs. However, the malicious nature of their use and the conceptual difference of preventing unintended functionality means that any analysis of vulnerabilities are subject to a variety of caveats.

When it comes to data security, a threat is any potential danger to information or systems. Threats could be an intruder network through a port on the firewall, a process accessing data in a way that violates the security policy, a tornado wiping out a facility, or an employee making an unintentional mistake that could expose confidential information or destroy a file's integrity.

Data security suggests the second principle of computer security.

Principle of Adequate Protection: Computer items must be protected only until they lose their value. They must be protected to a degree consistent with their value.

Figure 1.5 illustrates how the three goals of security apply to data. In particular, confidentiality prevents unauthorized disclosure of a data item, integrity prevents unauthorized modification, and availability prevents denial of authorized access.

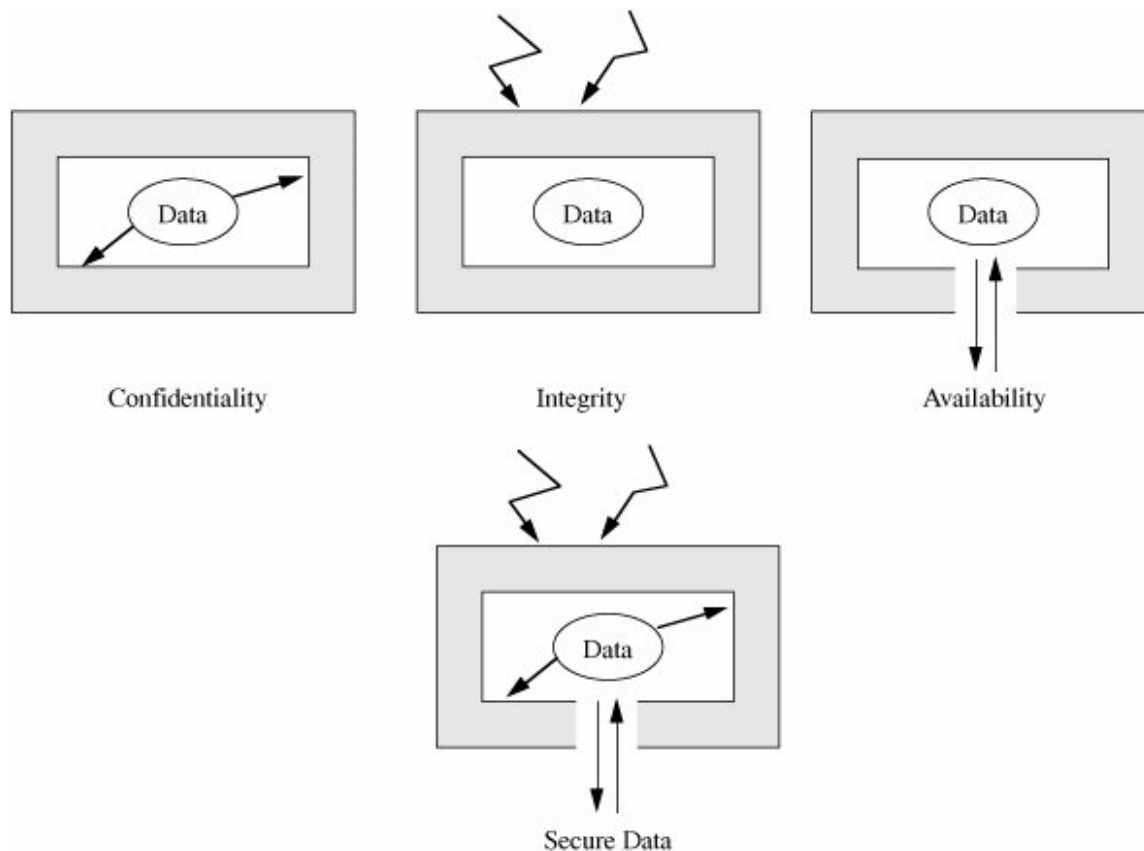


Figure 1.5. Security of Data.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Data confidentiality

Data confidentiality is about protecting data against unintentional, unlawful, or unauthorized access, disclosure, or theft.

Confidentiality has to do with the privacy of information, including authorizations to view, share, and use it. Information with low confidentiality concerns may be considered "public" or otherwise not threatening if exposed beyond its intended audience. Information with high confidentiality

concerns is considered secret and must be kept confidential to prevent identity theft, compromise of accounts and systems, legal or reputational damage, and other severe consequences.

Examples of data with high confidentiality concerns include:

- Social Security numbers, which must remain confidential to prevent identity theft.
- Passwords, which must remain confidential to protect systems and accounts.
- Consider the following when managing data confidentiality:
 - To whom data can be disclosed
 - Whether laws, regulations, or contracts require data to remain confidential
 - Whether data may only be used or released under certain conditions
 - Whether data is sensitive by nature and would have a negative impact if disclosed
 - Whether data would be valuable to those who aren't permitted to have it (e.g., hackers)

Guidelines for data confidentiality

When managing data confidentiality, follow these guidelines:

Encrypt sensitive files.

Encryption is a process that renders data unreadable to anyone except those who have the appropriate password or key. By encrypting sensitive files (by using file passwords, for example), you can protect them from being read or used by those who are not entitled to do either.

Manage data access.

Controlling confidentiality is, in large part, about controlling who has access to data. Ensuring that access is only authorized and granted to those who have a "need to know" goes a long way in limiting unnecessary exposure. Users should also authenticate their access with strong passwords and, where practical, two-factor authentication. Periodically review access lists and promptly revoke access when it is no longer necessary.

Physically secure devices and paper documents.

Controlling access to data includes controlling access of all kinds, both digital and physical. Protect devices and paper documents from misuse or theft by storing them in locked areas. Never leave devices or sensitive documents unattended in public locations.

Securely dispose of data, devices, and paper records.

When data is no longer necessary for University-related purposes, it must be disposed of appropriately.

Sensitive data, such as Social Security numbers, must be securely erased to ensure that it cannot be recovered and misused.

Devices that were used for University-related purposes or that were otherwise used to store sensitive information should be destroyed or securely erased to ensure that their previous contents cannot be recovered and misused.

Paper documents containing sensitive information should be shredded rather than dumped into trash or recycling bins.

Manage data acquisition.

When collecting sensitive data, be conscious of how much data is actually needed and carefully consider privacy and confidentiality in the acquisition process. Avoid acquiring sensitive data unless absolutely necessary; one of the best ways to reduce confidentiality risk is to reduce the amount of sensitive data being collected in the first place.

Manage data utilization.

Confidentiality risk can be further reduced by using sensitive data only as approved and as necessary. Misusing sensitive data violates the privacy and confidentiality of that data and of the individuals or groups the data represents.

Manage devices.

Computer management is a broad topic that includes many essential security practices. By protecting devices, you can also protect the data they contain. Follow basic cybersecurity hygiene by using anti-virus software, routinely patching software, whitelisting applications, using device passcodes, suspending inactive sessions, enabling firewalls, and using whole-disk encryption.

Data integrity

Data integrity is the assurance that digital information is uncorrupted and can only be accessed or modified by those authorized to do so. Integrity involves maintaining the consistency, accuracy and trustworthiness of data over its entire lifecycle.

To maintain integrity, data must not be changed in transit and steps must be taken to ensure that data cannot be altered by an unauthorized person or program. Such measures include implementing user access controls and version control to prevent erroneous changes or accidental deletion by authorized users. Other measures include the use of checksums and cryptographic checksums to verify integrity. Network administration measures to ensure data integrity include documenting system administration procedures, parameters, and maintenance activities, and creating disaster recovery plans for occurrences such as power outages, server failure or security attacks. Should data become corrupted, backups or redundancies must be available to restore the affected data to its correct state.

Network

Network security is any activity designed to protect the usability and integrity of your network and data. It includes both hardware and software technologies. Effective network security manages access to the network. It targets a variety of threats and stops them from entering or spreading on your network.

Access control

Access control is a security technique that regulates who or what can view or use resources in a computing environment. It is a fundamental concept in security that minimizes risk to the business or organization.

There are two types of access control: physical and logical. Physical access control limits access to campuses, buildings, rooms, and physical IT assets. Logical access control limits connections to computer networks, system files, and data.

To secure a facility, organizations use electronic access control systems that rely on user credentials, access card readers, auditing and reports to track employee access to restricted business locations and proprietary areas, such as data centers. Some of these systems incorporate access control panels to restrict entry to rooms and buildings as well as alarms and lockdown capabilities to prevent unauthorized access or operations.

Access control systems perform identification authentication and authorization of users and entities by evaluating required login credentials that can include passwords, personal identification numbers (PINs), biometric scans, security tokens or other authentication factors. Multifactor authentication, which requires two or more authentication factors, is often an important part of the layered defense to protect access control systems.

1.4. Computer Criminals

Computer criminals are people who are caught and convicted of computer crimes such as breaking into computers or computer networks. Computer crime can be broadly defined as criminal activity involving information technology infrastructure, including illegal access (unauthorized access), illegal interception (by technical means of non-public transmissions of computer data to, from or within a computer system), data interference (unauthorized damaging, deletion, deterioration, alteration or suppression of computer data), systems interference (interfering with the functioning of a computer system by inputting, transmitting, damaging, deleting, deteriorating, altering or suppressing computer data), misuse of devices, forgery (or identity theft) and electronic fraud

Amateur

People who are accidentally accessed to unauthorized resources and execution of unauthorized operations. They don't harm to the regular users. The amateurs are the "normal" people who exploit the apparent security flaws to gain an advantage. So is a worker in an office, who can simply read the mail from other users

Crackers or Malicious Hackers

The crackers have more knowledge than the amateurs. They see often a challenge to break into a system and most of them have the attitude that there is no real victim. They use the World Wide Web, email, forums, etc. to get the newest information about insecure systems. "There is no common profile or motivation to attackers [crackers]."

Career Criminals

The career criminals are real experts who started commonly as computer professionals. They break into the systems to get some important data and sell them. This is often their main income

Terrorists

There is a strong connection between computers and terrorism. We see terrorists in three cases using the computers:

Targets of attack: Denial-of-service attacks and website defacement are common with every political organization, because they draw attention to the cause and add unnecessary negative publicity to the subject of the attack.

Propaganda vehicles: Efficient, quick, and inexpensive ways of getting a message to several people are website, website logs and e-mail listing.

Methods of attack: Computers ought to be used to conduct offensive attacks

1.5. Methods of Defense

In the near future, computer crime is likely to continue. For this purpose, safeguards for confidentiality, honesty and availability need to be closely monitored. These controls can often deter or minimize attacks; other, less effective mechanisms can not but warn us that safety has been breached, whether it continues or has occurred.

Harm occurs when a weakness is attacked. So we can neutralize the threat, close the vulnerability, or both, to protect against damage. The risk is called the probability of harm. In various ways, we can deal with harm. We should try

- **Prevent it:** block the attack
- **Deter-it:** make the attack harder or more expensive
- **Deflect it:** make yourself less attractive to an attacker
- **Detect it:** notice that attack is occurring (or has occurred)
- **Recover from it:** mitigate the effects of the attack

More than one may of course be achieved immediately. We could try to avoid intrusion, for instance. However, if we don't prevent them, we might add a detectors device to you of a

forthcoming attack. In case of a successful intrusion, we will have incident response procedures in place to assist recovery.

Controls

In order to consider controls or counter measures to prevent the manipulation of the vulnerabilities of a computer system, we begin by considering conventional ways to improve physical protection. In the medieval ages castles and fortifications were built to protect the inhabitants and valuable property. The fortress may have had one or more protection features, like

- a strong door for repelling invaders
- Strong walls to prevent the throwing or projecting of objects
- a moat, for access control
- Arrow slits, allowing archers to fire at enemies
- Inhabitants may lean from their roof and spill out hot or vile liquids on attackers crenellations
- a drawbridge to restrict allowed persons' access
- Patrons must ensure that approved persons and products are permitted to enter

In the way we defend our homes and offices, we are using a multi-stakeholder approach. We can add extra door locks and a burglar alarm, strengthened windows and even a nosy neighbor to keep an eye on our precious things. In any case, we choose one or more ways to deter an attacker or intruder and our selection not only depends on the importance of our protection, but on the attempt an attacker or intruder may spend to get in.

The protection of computers has similar features. We have at our disposal a lot of reviews. Many of them are easier to use or incorporate than others. Others are less expensive to use or implement than others. And some individuals are easier to bypass than others. Figure 1-6 demonstrates how the controls we use to protect our precious resources. We use one or more checks according to what we protect, how security costs relate to the risk of failure and how likely we think the intruders can do what they want.

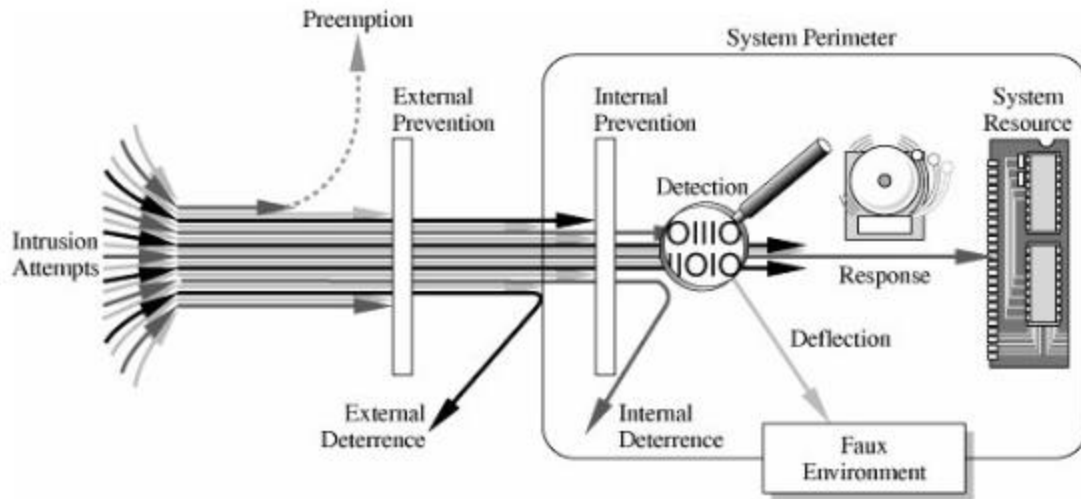


Figure 1-6. Multiple Controls.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Encryption

Coding is the most effective method for computer security. The value of an interception and the possibility of a change or manufacture are almost annulled through the transition of data to make it unreadable for the external observer.

Data protection is provided by encryption. In addition, encryption can be used for privacy as data which can not usually be read can not be modified. In addition, encryption is essential in protocols that are accepted sequences of measures to perform certain tasks. Such protocols ensure that services are available. Therefore, encryption is at the foundation of computer security strategies for these three objectives.

Encryption is a crucial computer security device, but its value should not be overestimated. Users will recognize that not all information security issues have been solved in encryption. In addition, encryption can not have any effect in terms of security or degrade the performance of the complete system if it is not used properly. So it is important to know the circumstances where encryption is useful and to use it efficiently.

. Software controls

Program itself is the second connection in computer security. To prevent external attacks, programs must be safe enough. They must also be developed and maintained to make sure that the systems are accurate. Program controls include the following kinds of things:

The following forms of program controls include:

- **Development controls**, which are standards for the design, coding, testing and maintenance of a program.
- **Operating system controls**, which are operating system limitations to protect each user against all other users.
- **Internal program controls** that enforcing safety restrictions, such as access restrictions to software controls in the database management

Tools including hardware components, encryption or the collection of information can be used by software controls. Computer controls typically directly impact users, and they are also the first aspects of computer security. As they affect the interaction of users with a computer program, device controls must be built carefully. The easiness of use and the power of software controls

Hardware Controls

Most hardware systems were designed to help computer security. Computer protection is also competing objectives. These devices vary from hardware encryption implementations to blocks that restrict access to theft to devices for users' identity verification.

Policies

The introduction of hardware or software features to handle such computing device controls as mentioned above. Additional controls are policy issues. In addition, some of the simplest tests, such as regular password changes, can be carried out with a tremendous effect at virtually no expense.

The security of computers includes legal and ethical inspections. The law is developing gradually and computer technology unexpectedly came into being. While it is important and beneficial to have legal defense, it is not as effective in this field as it is in more common and long-time crimes.

Computer ethics is equally unclear, not that computer people are ethically unethical, but that societies, and in particular the computer community, have not taken formal ethical behavior standards. Some organizations try to develop computer professional codes of ethics. While these are significant, the programming world and the general public need to understand what kind of behaviors are wrong until ethical codes are generally accepted and thus successful.

Physical Controls

Some of the easiest, fastest and cheapest controls are physical controls. Security checks include door locks, security guards, software and data backups and physical site plans which reduce the risk of natural disasters. Sometimes, when looking for more advanced approaches, we forget fundamental approaches

Effectiveness of Controls

Effectiveness of controls It is not sufficient just to have controls when properly used. Find several factors that can boost the efficiency of controls.

Awareness of Problem

Those using controls must be sure they need safety. Security is required. In other words, only when they realize that protection is required in a given situation are people readily compliant with safety standards. Nonetheless, many users may not know the need for safety, particularly if a group had recently carried out a computer task with a lack of protection or obvious safety.

Likelihood of Use

Naturally, no control is effective unless it is used. When people block the door open, the lock on a computer room door is not pleasant. Code clerks used obsolete codes during World War II because they had mastered them already and were able to easily decipher messages. Unfortunately, some of these codes were already broken on the other side and they could easily decode the messages.

Principle of Effectiveness. To be accurate, checks must be used. Effective, user-friendly and relevant.

This principle implies that computer security checks should be sufficiently efficient, with regard to time, memory, human activity or other resources used to prevent any serious effect on a task being protected by the application of the control. To order not to restrict valid accesses, controls should be limited.

Overlapping Controls

A single vulnerability can be resolved by several different controls as we have seen Fortress or Home security. In order to protect a microcomputer application for example, we can use a combination of controls for program access to data; physical access to micro-computers and storage media; and even file locking to control access to processing programs.

Periodic Review

The useful monitoring behaviors are permanent. Periodic analysis. The opposition is only duplicated by the fact that the security professionals find a way to defend assets from certain forms of attacks. Therefore, the efficacy of a regulation is an ongoing operation.

Controls are rarely, if ever, entirely effective. Functions of controls, inadequate controls, or controls, for example, have been bypassed or abused. Therefore, we use overlap controls, also known as layered protection, because one test balances another's failure. In other cases, controls complement each other well. But two controls are not necessarily better than one, and in some cases two can also be worse than one. This takes us to a different defense concept.

Principle of Weakest Link: Security can be the weakest link only. It can lead to a security failure if either firewall or operating system supplies power or someone prepares, implements and handles controls.

1.6 Review Question

1. Distinguish among vulnerability, threat, and control.
2. List at least three kinds of harm a company could experience from electronic espionage or unauthorized viewing of confidential company materials.
3. List at least three kinds of damage a company could suffer when the integrity of a program or company data is compromised.
4. Describe two examples of vulnerabilities in automobiles for which auto manufacturers have instituted controls. Tell why you think these controls are effective, somewhat effective, or ineffective
5. Preserving confidentiality, integrity, and availability of data is a restatement of the concern over interruption, interception, modification, and fabrication. How do the first three concepts relate to the last four? That is, is any of the four equivalent to one or more of the three? Is one of the three encompassed by one or more of the four?
6. Do you currently use any computer security control measures? If so, what? Against what attacks are you trying to protect?

1.7 References

1. Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger - RAND Corporation Publisher: Prentice Hall
2. Cryptography and Network Security - Principles and Practice fifth edition Stallings William Publisher: Pearson
3. Cryptography And Network Security 3rd Edition behrouz a forouzan and debdeep mukhopadhyay 3/E Publisher: McGraw Hill Education
4. Cryptography and Network Security, 3e Atul Kahate Publisher: McGraw Hill

Chapter 2. Elementary Cryptography

2.0 Introduction of Cryptography

2.1. Terminology and Background

2.2. Substitution Ciphers

1.2.1 Vulnerabilities, Threats, Attacks, and Controls

2.3. Transpositions (Permutations)

2.4. Making "Good" Encryption Algorithms

2.5 The Data Encryption Standard

2.6. The AES Encryption Algorithm

2.7. Public Key Encryption

2.8. The Uses of Encryption

2.9 Review Question

2.8 References

1.0 Introduction of Cryptography

Cryptography is process of converting ordinary plain text into unintelligible text and vice-versa. In cryptography, there are many strongest tools for controlling against many different kinds of security threats. The tools can convert data that cannot be read, modified, or fabricated easily.

Cryptography is based on higher mathematics, it requires in field of group and field theory, computational complexity, real analysis, probability and statistics. Perhaps it is not necessary to understand the underlying mathematics to be able to use cryptography.

The chapter begins by focusing at what encryption does and how it works. Using two basic encryption methods we implement the core concepts of encryption: substitution and transposition. Next, we will discuss how they can be improved to enhance and extend security. We also discuss how encryption can be compromised because poor or failed encryption offers only the illusion of security. We examine techniques used to crack the security mechanism and reveal the original text. Today, DES, AES and RSA are three very common algorithms. We analyze these and other algorithms in some depth to see if protocols and mechanisms can be used to perform certain computational tasks, such as signing of documents, detecting modifications, and exchanging sensitive data. We analyze these and other algorithms in some depth to see if protocols and mechanisms can be used to perform certain computational tasks, such as signing of documents, detecting modifications, and exchanging sensitive data. We analyze these and other algorithms in some depth to see if protocols and mechanisms can be used to perform certain computational tasks, such as signing of documents, detecting modifications, and exchanging sensitive data.

2.1. Terminology and Background

Consider the following steps for sending messages from a **sender**, S, to a **recipient**, R. If S send the message to T and entrust to T, then who will delivers the message to R, T could become the **transmission medium**. If an outsider, O, wants to access the message (in terms read, change, or even destroy the content of the message), Then we call O is an **interceptor** or **intruder**. If S wants to transmit the message at any time through the T, then message becomes vulnerable to exploitation, and O might try to access the content of the message in any of the following ways:

- The outsider will block the message, before reaching the message to R, hence it affect the availability of the message.
- The Intruder will intercept the message, by reading content of message or listening to the message, therefore its affects the confidentiality of the message.
- The intruder tries to modify the content of message, by seizing the message and or change the message in some other way, thereby affecting the integrity of message.
- The intruder try to fabricate an authentic-looking message, arranging for it to be delivered as if it came from S, thereby also affecting the integrity of the message.

In previous chapter 1, we have seen there are four security failures cause by a message's vulnerabilities. So the encryption is a technique that can address all these problems. Encryption, probably the most fundamental building block of secure computing, is a means of maintaining secure data in an insecure environment. (It is not the only building block, however.) In this book, we study encryption as a security technique, and we see how it is used in protecting programs, databases, networks, and electronic communications.

Terminology

Encryption is the process in cryptography that converts the plain text in to unreadable text (encrypted format); **decryption** is the reverse process in cryptography, that covert the encrypted text back into original format. The terms **encode** and **decode** or **encipher** and **decipher** are used instead of encrypt and decrypt i.e. we say that the word **encode**, **encrypt**, or **encipher** the original content of message to hide. The words **decode**, **decrypt**, or **decipher** it to reveal the content original message. A system used for the process of encryption and decryption is called a **cryptosystem**

The original content of a message is known as **plaintext**, whereas the encrypted text format is called **ciphertext**. This relationship is shown in Figure 2.1. For more convenient way for describing the relation, we denote a plaintext message as letter P as a it contains a sequence of individual characters $P = \langle p_1, p_2, \dots, p_n \rangle$. Similarly, we denote a ciphertext is written as letter C $= \langle c_1, c_2, \dots, c_m \rangle$. consider an example , the plaintext message written as "I want a code" can be denoted as the message of string $\langle I, w, a, n, t, , a, , c, o, d, e, \rangle$. It can be transformed into ciphertext $\langle c_1, c_2, \dots, c_{13} \rangle$, and the encryption algorithm tells us how the transformation is done.



Figure 2.1. Encryption.

Above Image taken from the book **Security in Computing, Fourth Edition** By **Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger**

We describe the formal notation for transformations between plaintext and ciphertext. For example, we write $C = E(P)$ and $P = D(C)$, where C denote as ciphertext, E denote as the encryption rule, P denote as the plaintext, and D is denote for a decryption rule. We want a cryptosystem to be $P = D(E(P))$. In other words, we want to be able to convert the plaintext message to ciphertext message in order to protect from an intruder, but we also retrieve the get the original message from the ciphertext from which intended receiver able to read the message.

Encryption Algorithms

The cryptosystem defines sets of rule for how to encrypt the plaintext and how to decrypt the ciphertext message. The process of define a rules for encryption and decryption are called as **algorithms**, the main component in the algorithm is key, which denoted as K , from which we get resulted ciphertext from the plaintext, the algorithm, and the key value. As we note dependence $C = E(K, P)$. Whereas E content is a set of encryption algorithms, and here we use the key K selects for one specific algorithm from the set.

The algorithm which use the same key for encryption and decryption keys, is called a **symmetric** encryption, the notation for the both the process with the key is $P = D(K, E(K, P))$. The algorithms uses the different key for encryption and different decryption is called, **asymmetric** encryption, Here a decryption key K_D , inverts the encryption of key K_E so that $P = D(K_D, E(K_E, P))$.

The difference between symmetric and asymmetric encryption is shown in **Figure 2.2**.

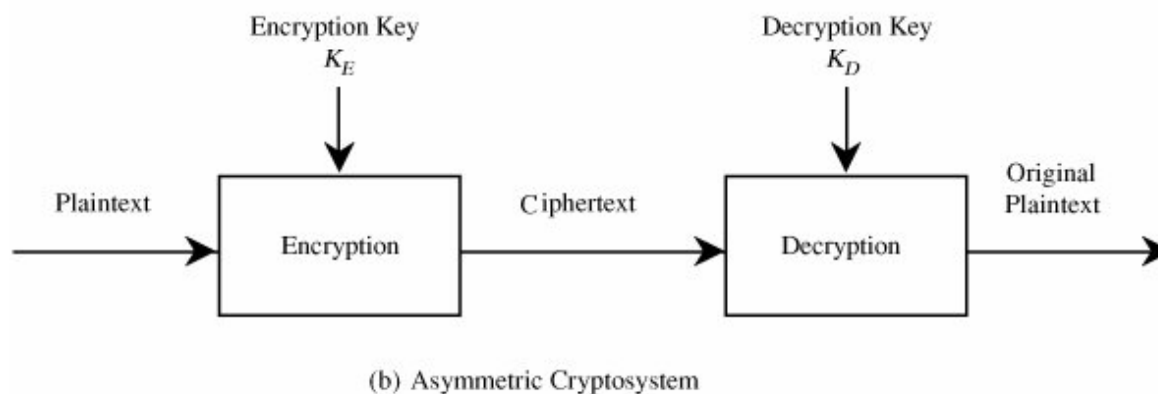
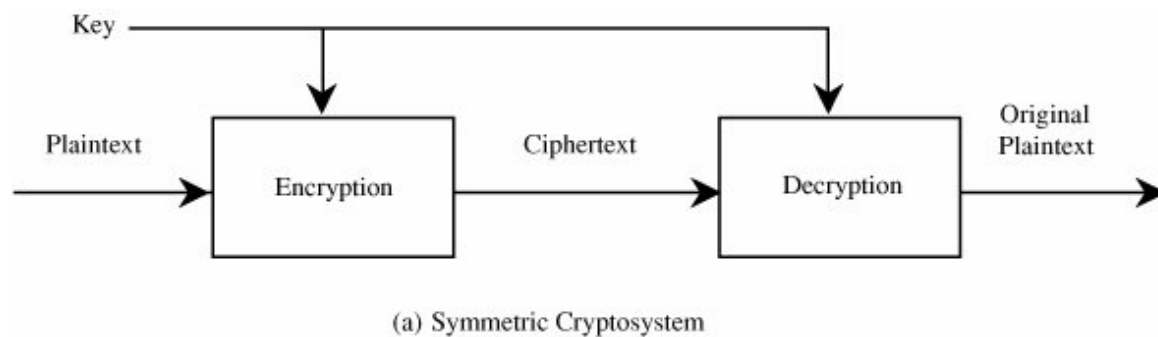


Figure 2.2. Encryption with Keys.

Above Image taken from the book **Security in Computing, Fourth Edition** By **Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger**

An encryption scheme which does not require a key is called a **keyless cipher**

The word **cryptography** means to write the hidden content by the practice of using encryption to conceal text. A **cryptanalyst is a person which** studies the process encryption and encrypted messages, in order to find the plain content from the hidden content

Both a cryptographer and a cryptanalyst may try an attempt to translate hidden or coded text back to its original form. Here the two terminology is different, i.e. a cryptographer will work on behalf of a legitimate user i.e. sender or receiver, whereas a cryptanalyst will work on behalf of an unauthorized interceptor.

Cryptology is the research in which we use to study of encryption and decryption process; which includes both cryptography and cryptanalysis.

Cryptanalysis

The main aim of cryptanalyst's is to **break** an encryption process. That is, the cryptanalyst try to find and guess by using the original meaning of a ciphertext message. The intruder hope to determine which decryption algorithm will match the encrypting algorithm so the encoded messages will be broken to get plaintext. For instance, suppose two countries fighting with each other and second country will send the encoded message to own army headquarter ,the first country has intercepted encrypted messages of the second country. Cryptanalysts of the first country will decipher a particular message of the second country message so that the first country will anticipate the movements and resources of the second. If the first country have better decryption algorithm; then the first country can easily break the encryption of all messages sent by the second country.

Thus, a cryptanalyst can attempt to do any or all of six different things:

- It can break a single plain text message
- It can recognize patterns in encrypted messages, so cryptanalyst will able to break subsequent messages by a straightforward decryption algorithm
- Cryptanalyst will try to guess some meaningful information without having broken the encryption text by noticing an unusual frequency of communication or determine whether the communication was short or long for breaking the encrypted text.
- It will try to find the actual key applied in the algorithm, to break encrypted messages easily to get the readable message.
- It will try finding weaknesses in the implementation of algorithm that use of encrypting the plaintext.
- It will try to find general weaknesses in an encryption algorithm, without failing to intercept any plaintext message.

Breakable Encryption

An encryption algorithm is known as breakable when a cryptanalyst can determine the algorithm and given an ample amount of time and space for breaking the encrypting algorithms.. However, theoretically an algorithm can be broken may not be impractical to try to break an algorithm. Consider an example let a 25-character message can be expressed in just uppercase letters, so given cipher scheme may have 26^{25} (approximately 10^{35}) possible ways for decryption, so the task is to select the right one out of the 26^{25} . If your computer could perform computation of message on the order of 10^{10} operations per second, finding this decipherment would require on the order of 10^{16} seconds, so it would roughly take a 10^{11} years for breaking the encrypted text. In this case, we theoretically we could create the solution of decipherment by determining the deciphering algorithm by examining all possibilities way. But still we have to ignore as infeasible with current technology.

Representing Characters

In computer system, we want to study different ways any character representation, whether it is written as ASCII characters, binary data, object code, or a control stream. In order who the character are encrypted consider an example, we start with an encryption of messages written in the standard 26-letter English alphabet, A through Z.

LETTER CODE	A 0	B 1	C 2	D 3	E 4	F 5	G 6	H 7	I 8	J 9	K 10	L 11		
LETTER CODE	M 12	N 13	O 14	P 15	Q 16	R 17	S 18	T 19	U 20	V 21	W 22	X 23	Y 24	Z 25

In above, the letter A is represented by a zero, B by a one, and so on. This representation allows performing an arithmetic operation on the "letters" of a message. Then we can also perform addition and subtraction on letters by adding and subtracting the corresponding code numbers. With the representation of Expression on letter such as $A + 4 = E$ or $L - 1 = K$. Arithmetic operation can be performed in a alphabetic order where the alphabetic table order were circular. In other words, addition wraps around from one end of the table to the other so that $Y + 4 = C$. Thus, every result of an arithmetic operation is between 0 and 25.

There are many types of encryption. We look two different forms of encryption: **substitutions**, where one letter is exchanged for another, and **transpositions**, where the order of the letters is rearranged. The main aims of two forms for studying with the different concept of encryption and decryption, we will learn some of the terminology and methods of cryptanalysis, and to study some of the weaknesses to which encryption is prone.

2.2. Substitution Ciphers

Substitution ciphers is a technique in which characters from the plaintext are simply substituted (replaced in a specific manner) with another set of characters, we get a results in the form of ciphertext. This technique as called as a **monoalphabetic cipher** or **simple substitution**.

The Caesar Cipher

The Caesar Cipher, also called as a shift cipher, it is one of the oldest and simplest technique for encryption of a message. It is a type of substitution cipher technique, where each letter in the original message is replaced with a letter corresponding to a certain number of letters shifted up or down in the alphabet.

For an example, the Caesar Cipher encryption of a full message, using a left shift of 3.

Plaintext: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Ciphertext: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Advantages and Disadvantages of the Caesar Cipher

Advantages include:

- It is one of the simplest methods to use in cryptography and can provide less security to the information
- It only use a short key in the entire process
- One of the best methods to use if the system cannot use any complicated coding techniques requires few computing resources
- One of the easiest methods to use in cryptography and can provide minimum security to the information

Disadvantages of using a Caesar cipher include:

- It has Simple structure usage
- It can only provide minimum security to the information
- Frequency of the letter pattern provides a big clue in deciphering the entire message

Cryptanalysis of the Caesar Cipher

The Caesar cipher can be cracked with computer devices in milliseconds, arguably the simplest keyboard cipher. As there are only 25 key choices available (every alphabet shift possible), we are only attempting to decrypt the ciphertext with every key This solution type is called 'brute force' and only available for very simple ciphers..

An Example

Our ciphertext is the following:

YMJHFJXFWHNUMJWNXTSJTKYMJJFWQNJXYPSTBSFSIXNRUQJXYHNUMJWX

To find out what the original was, we try decrypting it with each of the 25 possible keys, calculating the fitness for each trial decryption:

Key	plaintext
-----	-----------

1	XLIGEIWEVGMTLIVMWSRISJXLIIEVPM...
2	WKHFDHVDUFLSKHULVRQHRIWKHHDUOL...
3	VJGECGUCTEKRJGTKUQPGQHVJGGCTNK...
4	UIFDBFTBSDJQIFSJTPOFPGUIFFBSMJ...
5	THECAESARCIPHERISONEOFTHEEARLI...
6	SGDBZDRZQBHOGDQHRNMDNESGDDZQKH...
7	RFCAYCQYPAGNFCPGQMLCMDRFCCYPJG...
8	QEBZXPXOZFMEOFLKBLCQEBBXOIF...
9	PDAYWAOWNYELDANEOKJAKBPDAAWNHE...
10	OCZXVZNVMXDKCZMDNJIZJAOCZZVMGD...
11	NBYWUYMULWCJBYLCMIHYIZNBYYULFC...
12	MAXVTXLTKVBIAXKBLHGXYMAXXTKEB...
13	LZWUSWKSJUAHZWJAKGFWGXLZWWSJDA...
14	KYVTRVJRITZGYVIZJFEVFWKYVVRICZ...
15	JXUSQUIQHSYFXUHYIEDUEVJXUUQHBY...
16	IWTRPHTHPRXEWGTGXHDCTDUIWTTPGAX...
17	HVSQOSGOFQWDVSFWGCBSCTHVSSOFZW...
18	GURPNRFNEPVCUREVFBARBSGURNEYV...
19	FTQOMQEMDOUBTQDUEAZQARFTQQMDXU...
20	ESPNLPDLCNTASPCTDZYPZQESPLCWT...
21	DROMKOCKBMSZROBSCYXOYPDROOKBVS...
22	CQNLJNBALRYQNARBXWNXOCQNNJAUR...
23	BPMKIMAIZKQXPMZQAWVMWNBPMIMIZTQ...
24	AOLJHLZHYJPWOLYPZVULVMAOLLHYSP...
25	ZNKIGKYGXIOVNXOYUTKULZNNKKGXRO...

The art of breaking codes and ciphers is called as cryptanalysis. Perhaps the simplest cipher to crack is the Caesar cipher. Given that the shift has to be 1 to 25 numbers (0 or 26 will lead to an unchanged plaintext), we can only try each possibility to see the text is readable. When you know what a part of the ciphertext is or you can guess a part, then you can find the key instantly.

If this is not feasible, the frequency distribution of letters in the cipher text should be determined in a more systematic manner. This includes the number of times a letter appears. The text has a rather distinct distribution in natural English which can be used for crack codes.

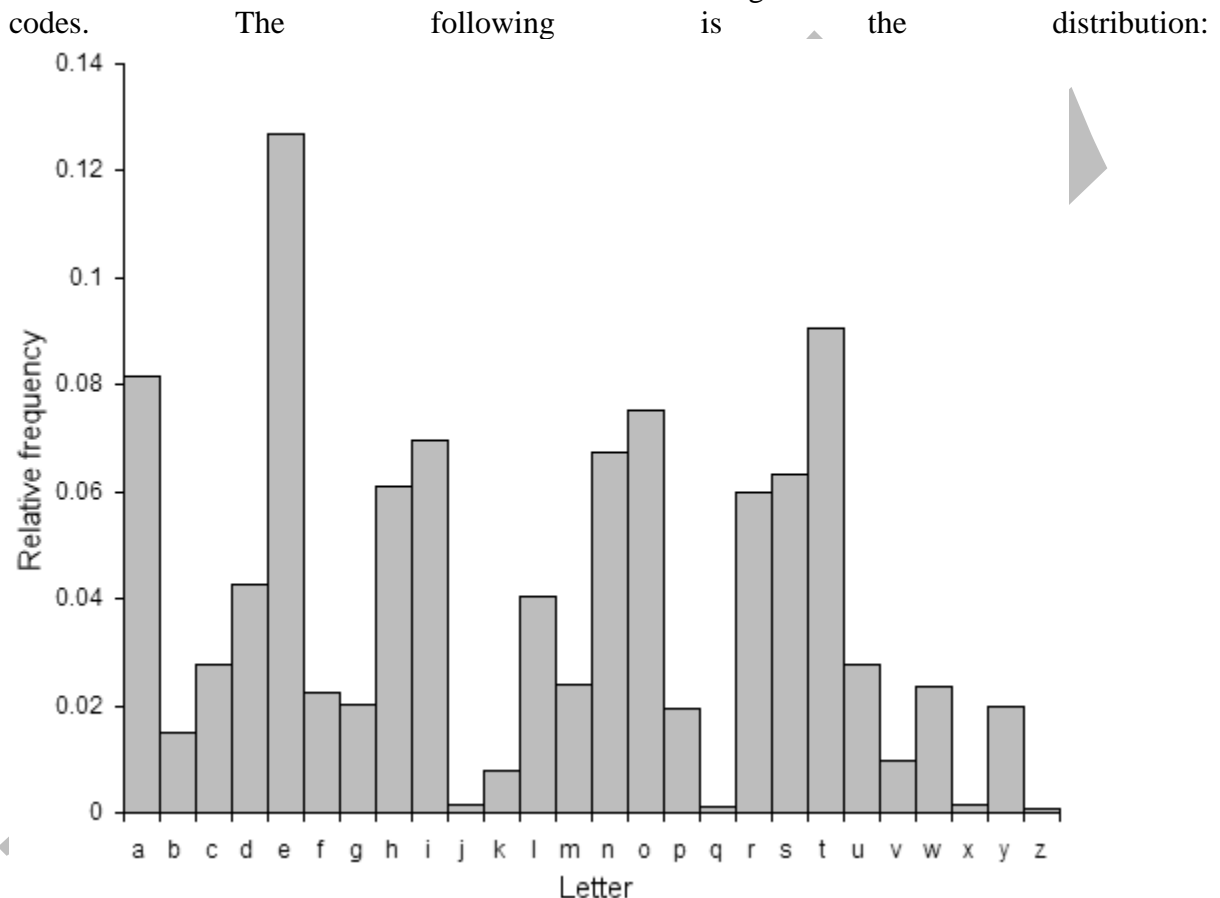


Figure 2.3 Frequency of Letter Occurrence

This means that the letter **e** is the most common, and appears almost 13% of the time, whereas **z** appears far less than 1 percent of time. Application of the Caesar cipher does not change these letter frequencies, it merely shifts them along a bit (for a shift of 1, the most frequent ciphertext letter becomes **f**). A cryptanalyst just has to find the shift that causes the ciphertext frequencies to match up closely with the natural English frequencies, then decrypt the text using that shift. This method can be used to easily break Caesar ciphers by hand.

One -Time Pad

The One-Time Pad, or OTP is an encryption technique in which each character of the plaintext is combined along with a character from a random **key stream**. Originally described in 1882 by banker Frank Miller (USA), it was re-invented in 1917 by Gilbert Vernam and Joseph Mauborgne. When applied correctly, the OTP provides a truly unbreakable cipher. It is named after the sheets of paper (pads) on which the key stream was usually printed. It also exists as *One Time Tape* (OTT).

Theory

The theory behind the OTP is that the encryption key has at least the same length as the actual message (i.e. the plaintext) and consists of truly random numbers. Each letter of the plaintext is 'added' to one element from the OTP using modulo-addition. This results in a cipher text that has no relation with the plaintext when the key is unknown. At the receiving end, the same OTP is used to retrieve the original plaintext.

For this to work, the following rules are mandatory:

- The OTP should consist of truly random characters (noise).
- The OTP (i.e. the key) should have the same length as the plaintext (or longer).
- Only two copies of the OTP should exist.
- The OTP should be used only once.
- Both copies of the OTP are destroyed immediately after use.

The Vernam Cipher

The Vernam cipher is a type of one-time pad devised by Gilbert Vernam for AT&T. The Vernam cipher is immune to most cryptanalytic attacks. The basic encryption involves an arbitrarily long nonrepeating sequence of numbers that are combined with the plaintext.

Algorithm of Vernam cipher:

For a string of m numbers, a string of m random numbers is generated using a key r which is "large prime number". Here the term "large" is in a sense that it should have as many bits as the message to be transmitted has.

Encrypted output $E(i) = (x(i) + k(i)) \% 26$

$x(i)$ = Number at the i th position in input string

$k(i)$ = Corresponding random number generated

Hence ' m ' random numbers + ' m ' meaningful numbers give rise to set of m numbers which form the encrypted message.

Decrypted output $D(i) = (x(i) - k(i)) \% 26$

Example of Vernam Cipher

- Here, we combine the key and the message using modular addition.

- The numerical values of corresponding message and key letters are added together, modulo 26.
- If key material begins with "XMCKL" and the message is "HELLO", then the coding would be

OTP Encryption Example

H	E	L	L	O	MESSAGE
7(H)	4(E)	11(L)	11(L)	14(O)	MESSAGE
+23(X)	12(M)	2(C)	10(K)	11(L)	KEY
=30	16	13	21	25	MESSAGEE +KEY
=4(E)	16(Q)	13(N)	21(V)	25(Z)	MESSAGEE +KEY (MOD 26)
E	Q	N	V	Z	➔ CIPHERTEXT

If a number is larger than 25, then the remainder after subtraction of 26 is taken in modular arithmetic fashion .This simply means that if your computations "go past" Z, you start again at A. The ciphertext to be sent to Bob is thus "EQNVZ". Bob uses the matching key page and the same process, but in reverse, to obtain the plaintext. Here the key is *subtracted* from the ciphertext, again using modular arithmetic.

OTP Decryption

E	Q	N	V	Z	CIPHERTEXT
4(E)	16(Q)	13(N)	21(V)	25(Z)	CIPHERTEXT
-23(X)	12(M)	2(C)	10(K)	11(L)	KEY
-19	4	11	11	14	CIPHERTEXT -KEY
7(H)	4(E)	11(L)	11(L)	14(O)	CIPHERTEXT -KEY (MOD 26)
H	E	L	L	O	➔ MESSAGE

NB: If a number is negative then 26 is added to make the number positive

OTP

Cryptanalysis

- Suppose Eve intercepts Alice's ciphertext: "EQNVZ". If Eve had infinite computing power, she would quickly find that the key "XMCKL" would produce the plaintext "HELLO", but she would also find that the key "TQURI" would produce the plaintext "LATER"

E	Q	N	V	Z	CIPHERTEXT
4(E)	16(Q)	13(N)	21(V)	25(Z)	CIPHERTEXT
-19(T)	16(Q)	20(U)	17(R)	8(I)	POSSIBLE KEY
-15	0	-7	4	17	CIPHERTEXT -KEY
11(L)	0(A)	17(T)	4(E)	17(R)	CIPHERTEXT -KEY (MOD 26)

It is possible to "decrypt" out of the ciphertext any message whatsoever with the same number of characters, simply by using a different key, and there is no information in the ciphertext which will allow Eve to choose among the various possible readings of the ciphertext. Thus, OTP coined, the "Perfect Cipher"

Book Ciphers

A book cipher uses a large piece of text to encode a secret message. Without the key (the piece of text) it is very difficult to decrypt the secret message.

To implement a book cipher, each word in the secret message would be replaced with a number which represents the same word in the book. For example, if the word "attack" appeared in the book as word number 713, then "attack" would be replaced with this number. The result would be an encoded message that looked something like this.

713 23 245 45 124 1269 586 443 8 234

To decipher the message you simply count the number of words in the book and write down each one.

Vigenere Cipher

In a Caesar Cipher every letter in the alphabet is moved along several locations; for instance, in a shift 3 Caesar cipher, A will become D, B will become E and so on. The Vigenere cipher consists of multiple Caesar cipher with different change values in sequence

A table of alphabets called a table recta, Vigenere square, or Vigenère table may be used to encipher them. The alphabet consists of 26 times a row, each of which has been cyclically moved to the left compared to the previous alphabet, which equates to the 26 possible ciphers of Caesar. The cipher uses a different alphabet from one row at different points in the encryption method. The alphabet used depends on a repeated keyword at every point.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 2.4 Vigenère square or Vigenère Table

Above Image taken from the book **Security in Computing, Fourth Edition** By **Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger**

For example, suppose that the plaintext to be encrypted is:

ATTACKATDAWN

The sender of the message selects and repeats a key word until it corresponds with the length of the plaintext, such as the keyword "LEMON":

LEMONLEMONLE

The intersection of the plaintext letter and the keyword letter is encoded in each letter. For eg, the first letter of the plaintext, A, is coded by the first letter of the main alphabet of row L. This is accomplished by analyzing the letter in row L and column A, namely L, of the Vigenere square. The second letter of the key will likewise be used in the second letter of the plaintext. The letter of rows E and column T is X. The remainder of the plaintext is equally encrypted:

Plaintext: ATTACKATDAWN

Key: LEMONLEMONLE

Ciphertext: LXFOPVEFRNHR

Decryption is done with a ciphertext letter position in a table row, followed by a column label in which it is shown as a plaintext. In row L, for instance, the ciphertext L appears in the first letter in column A. By looking at X in row E, the second letter is decrypted and is given in the column T as plaintext letter.

Transposition Cipher

A transposition cipher is a cipher rearranged by some predetermined form, which re-arranges the order of the letter in the cipher (encoded text). The aim of the cryptography with transposition is to diffusion, spreading the information widespread from the message or from the key through the ciphertext. Transpositions attempt to break patterns. Since a transposition represents a rearrangement of a message's symbols, it is also known as a permutation.

Columnar Transpositions

It is another type of cipher, which rearranges the order of the alphabets in the plaintext to create the ciphertext. The existing alphabets for plaintext are not substituted.

One example is the 'simple columnar transposition' cipher with a certain amount of alphabetic width. The plaintext is written horizontally. The ciphertext is then read vertically.

For instance, the plaintext is "modern statue in white house," and the secret random key chosen is "five." This text is arranged in a horizontal table with the column number equal to the key value. The following is the corresponding text.

m	o	d	e	r
r	s	t	a	t
u	e	i	n	w
h	i	t	e	h
o	u	s	e	

The ciphertext is obtained from the first to the last column by reading the column vertically downwards. 'mruho oseiu dtitse anee rtwh' is the ciphertext.

The recipient sets the same table for decryption. This is equal key to the number of the columns. The number of rows is obtained through a division by key value in the total number of the ciphertext alphabets and the rounds of the quotient to the next integer value. The receiver then writes the received ciphertext vertically down and from left to right column. To obtain the text, he reads horizontally left to right and from top to bottom row.

Monogram, Bigram and Trigram frequency counts

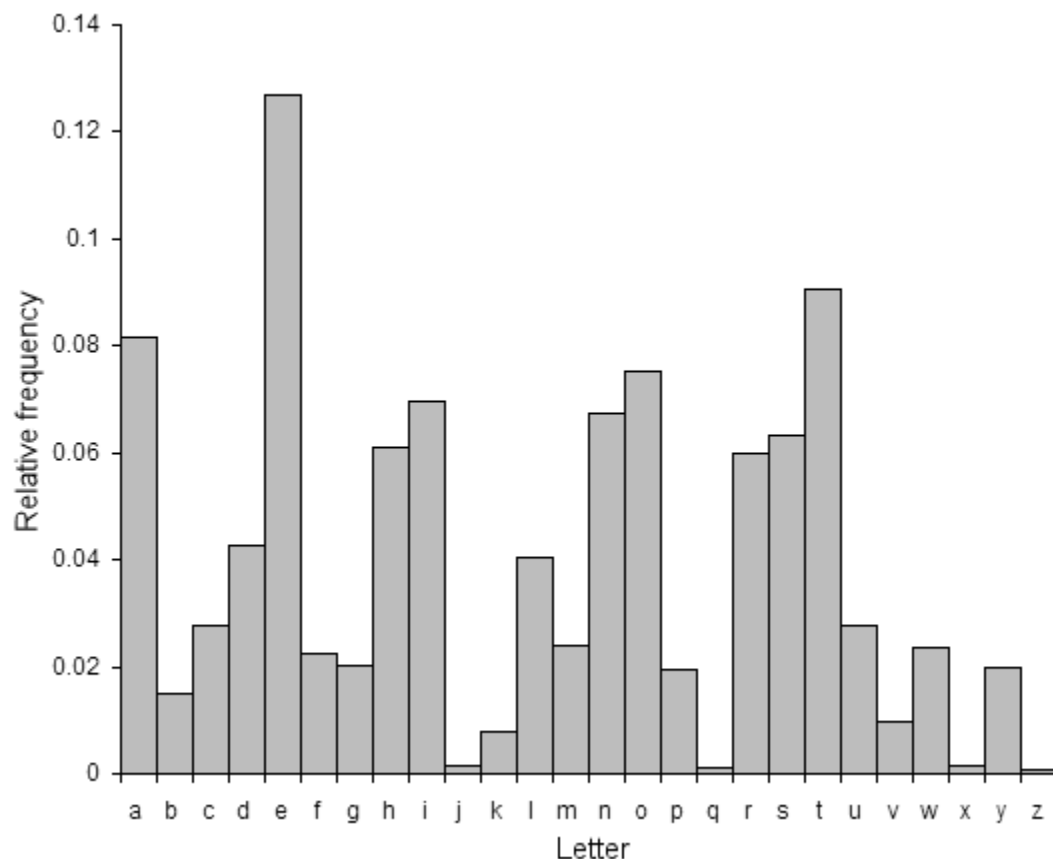
Introduction to Frequency Analysis

Frequency analysis is the process of counting the number of occurrences of various characters in ciphertext in the expectation that information will also be used to crack ciphers. Frequency analysis not only involves the single character but may also calculate how frequently pairs of characters appear in the text (bigrams) often known as digraphs. Trigram frequency counts measure the occurrence of 3 letter combinations.

We should focus on text characterization in relation to solving polygraphic cipher (e.g. playfair) while thinking about the bigrams and trigrams. The difference is that text characteristics depend on all possible two character combinations, as we want to learn as many bigrams as we can. We don't permit the bigrams to overlap when you crack the playfair.

Monogram Counts

In cipher type substitution, such as the caesar cipher, substitution chipers, polybius square, etc. monogram frequency counts are most effective. It works because the normal English text follows a very particular distribution of frequency which is not masked by substitution chipers. It looks like the distribution:

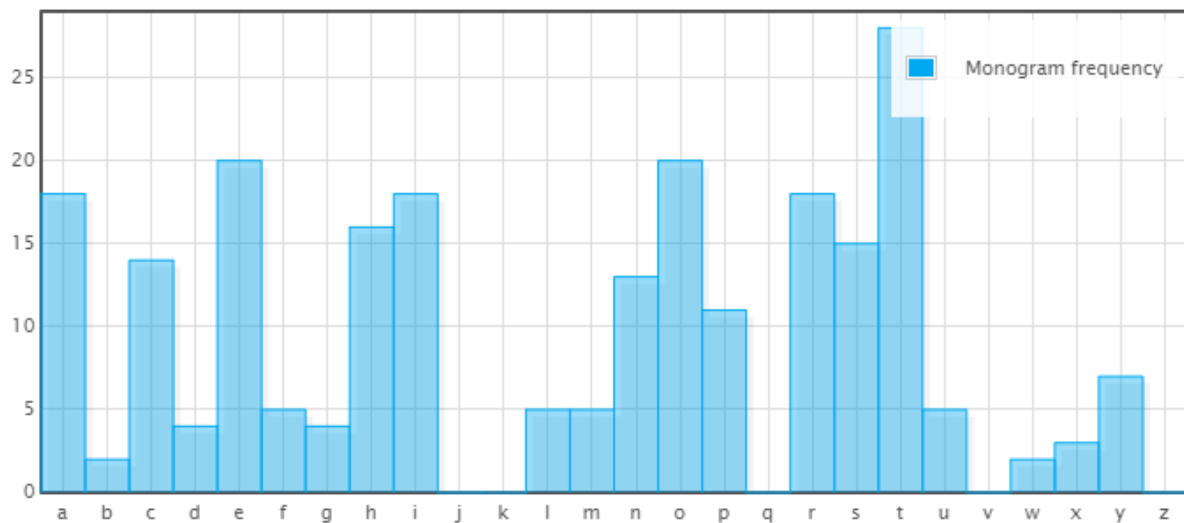


See an Example substitution cipher cryptanalysis on applications of frequency counts for solving substitution ciphers.

Consider a below text

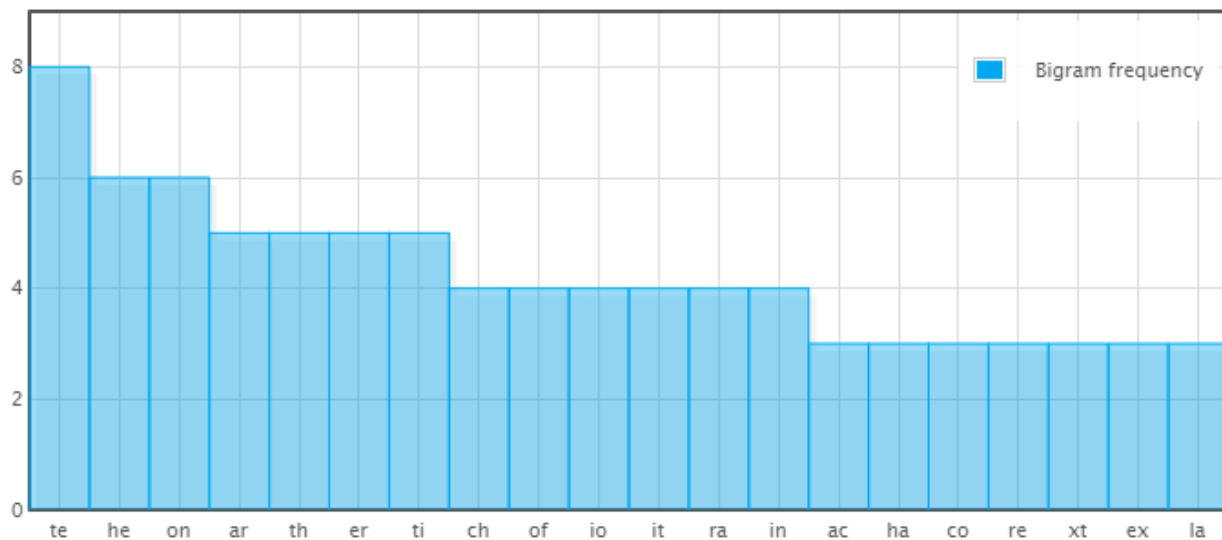
“In cryptography , a transposition cipher is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext.”

In Monogram diagram, Occurrence of letters



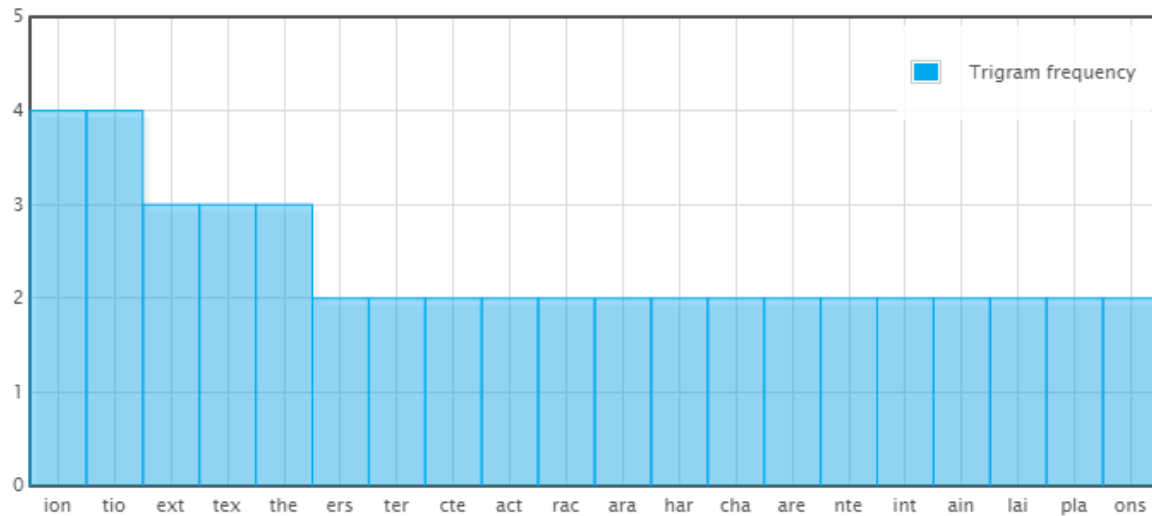
Bigram Counts

Bigrams counts the same concept as monogram counts; however, the bigram counts count the frequency of the pairs of characters rather than count the occurrences of the single characters.



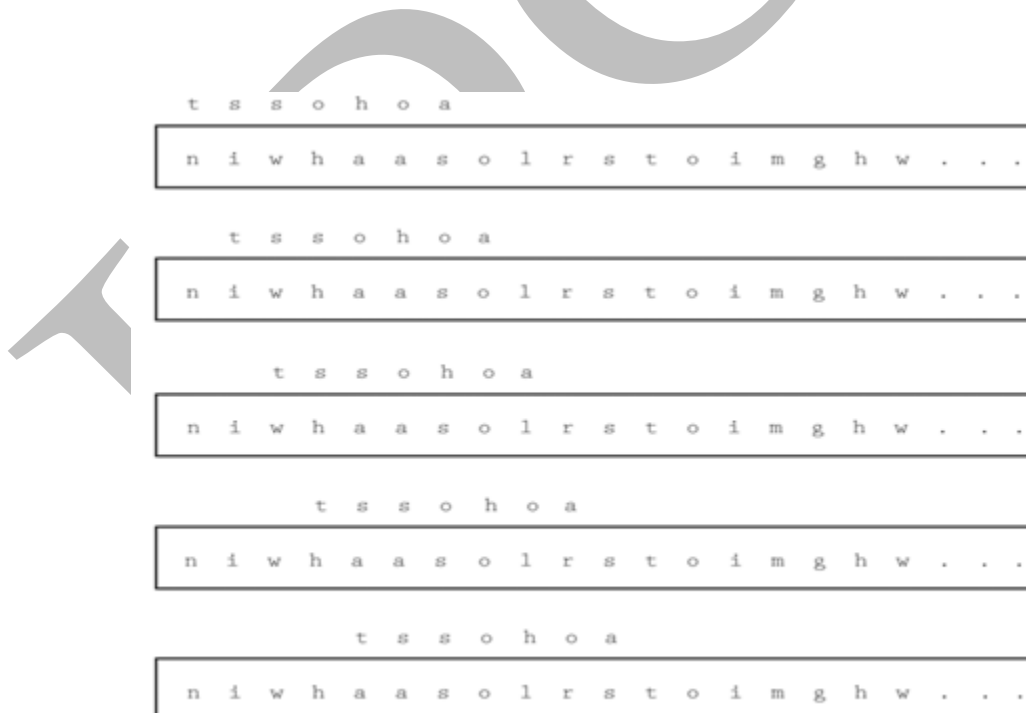
Trigram Counts

Just as bigrams count the pair frequency of characters, trigrams count the three-character frequency.



Cryptanalysis by Digram Analysis

Assume the block being compared is seven characters. The first comparison is c1 through c8, c2 through c9, ..., c7 through c14. Then we try eight characters away, so we can match the window of comparison shifts and c1 with c9, c2 to c10 and then continuing. Two questions are asked for each window location. Firstly, common digrams appear and secondly, most digrams seem reasonable.



Combinations of Approaches

- Substitution and transposition can be considered as encryption building blocks.
- The product cipher is defined as a combination of two ciphers.
- Product ciphers, as in $E_2(E_1(P, k_1), k_2)$, are usually carried out one by one.

2.4. Making "Good" Encryption Algorithms

- What makes an algorithm for secure encryption?
- What does it mean for a cipher to be "good"?
- The meaning of good depends on the intended use of the cipher
- cipher used by military personnel in the field has specific needs to be used in a secure facility with significant computer support, as compared with one to be used in a secure network
- In this section we look at the different characteristics of cipher more closely

Shannon's Characteristics of "Good" Ciphers

Claude Shannon suggested several characteristics in 1949, which defined a good cipher.

1. The required level of secrecy would decide the quantity of labor for encryption and decryption.
2. The key set and the algorithm enciphering should be without complicity
3. The implementation of process should be as simple as possible.
4. Ciphering errors should not spread and cause additional information in the message to be corrupted.
5. The enciphered text does not surpass the original message text.

Properties of "Trustworthy" Encryption Systems

Commercial consumers need to follow many criteria when choosing an encryption algorithm. Therefore, if we say that encryption is "commercial grade," or "trustworthy," we mean that it meets these requirements: Not only are good cryptographic algorithms invented but they derive from solid principles.

It was analyzed and found to be sound by professional experts. Even the best cryptographic experts will only think of as many attacks as possible and developers would become too convinced of the strength of their own algorithms. Therefore, it is important for key external experts to check.

As a new algorithm becomes more popular, people are continuing to study both their mathematical foundations and how they build on those foundations. Although a long duration of successfully used and evaluated algorithms are not guaranteed, faults in many algorithms are identified fairly soon after they have been released.

The world of commercial enterprises is famous for three Algorithms: DES (data encryption standard), RSA (Rivest Shamir Adelman named after inventor) and AES, (advanced encryption

standard). The algorithms for DES and RSA (as well as others) meet our commercial encryption criteria; AES, which is quite a recent technology, follows the first two and starts to be accomplished.

2.5 The Data Encryption Standard

IBM set up a computer-based cryptography research project headed by Horst Feistel in the late 1960's. The project ended in 1971 when the LUCIFER algorithm had been developed. LUCIFER is a Feistel block cipher operating in 64-bit blocks using a 128-bit key size.

IBM, led by Walter Tuchman and Carl Meyer, is trying to develop a marketed commercial encryption product which can be ideally implemented on one chip due to the promising results produced by the LUCIFER project. It included not only IBM researchers but consultants outside the NSA. The outcome of this effort was a simplified LUCIFER version, which was more cryptanalytic but had a 56 bits key size to fit on a single chip.

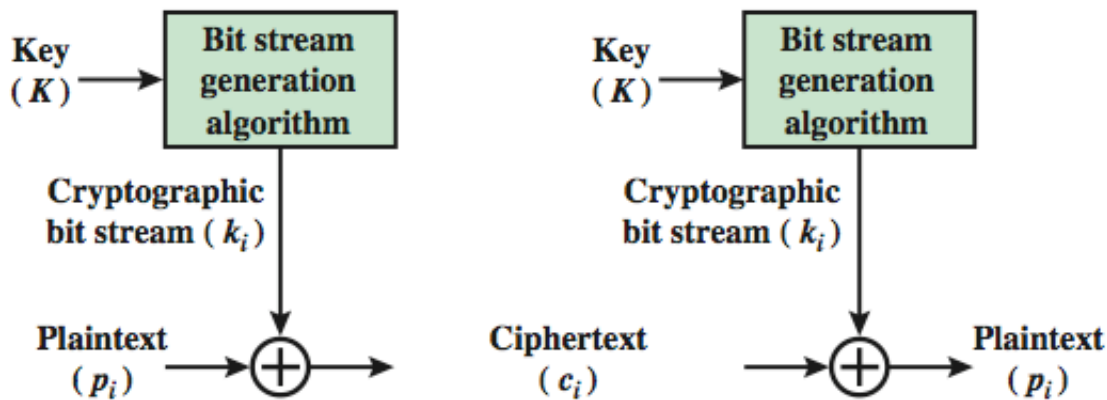
A national cipher standard specification was published by the National Bureau of Standards (NBS) in 1973. The amended LUCIFER was presented by IBM. The Data Encryption Standard was introduced in 1977 as it was by far the best possible algorithm.

A system developed for the US government, the Data Encryption Standard (DES) was intended for use by the general public.

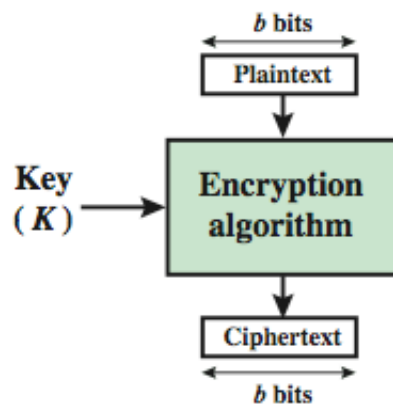
Block vs Stream Ciphers

Block ciphers operate a variety of bits on one word / block at a time. All such bits must be accessible before the processing of the block is possible. Stream cipher works on a certain bit or byte of the message; hence it is interpreted as a "stream." Currently, block ciphers are better evaluated. A block cipher tends to have an even broader variety of applications in which the plaintext block is handled as an entire packet and used in order to generate a same length ciphertext message. A 64 or 128-bit block size is usually used. Like a stream cipher, the two users share a symmetrical encryption key. One stream cipher is used to encrypt one bit or one

byte of digital data stream at a time.



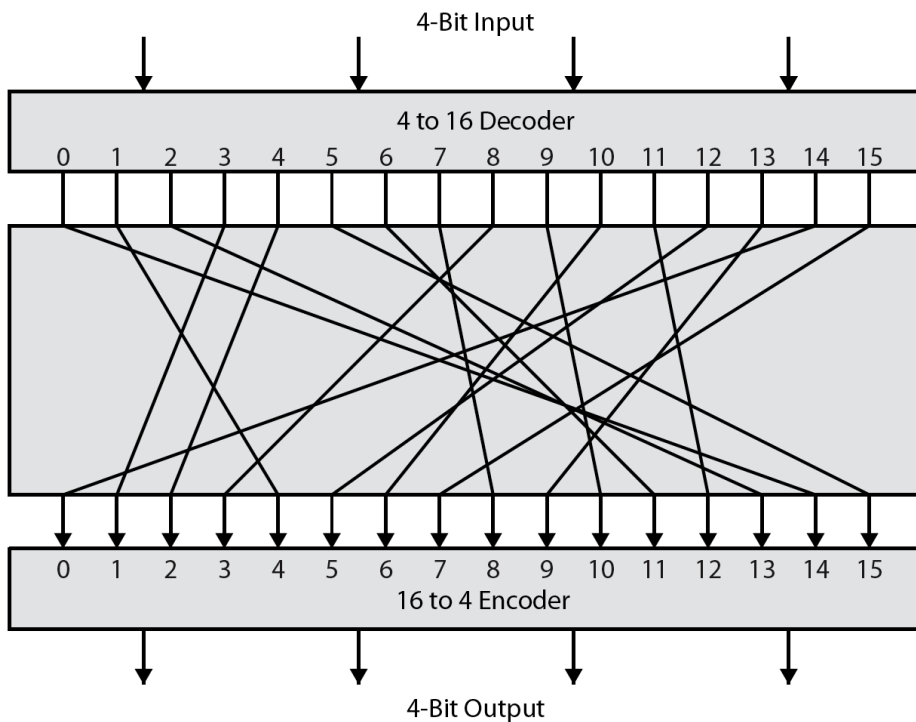
(a) Stream Cipher Using Algorithmic Bit Stream Generator



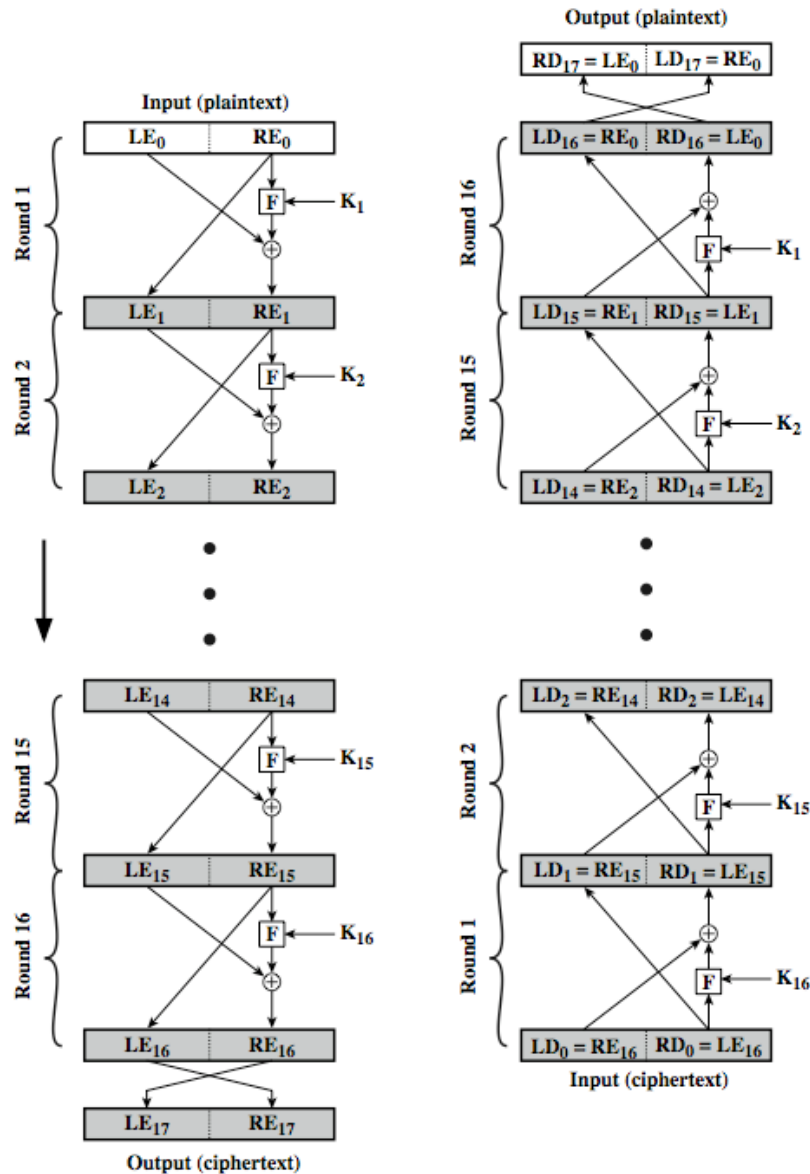
(b) Block Cipher

In current usage, the most symmetric block encryption algorithms are based on a model known as a Feistel block cipher. A block cipher operates on a n bits plaintext block to produce a n bits ciphertext block. However, from an implementation and efficiency point of view, an arbitrary reversible substitution cipher for a broad block size is not realistic. The size of a key is normally $n \times 2n$ for the n -bit general block substitution cipher. The key size of a 64-bit block is $64 \times 264 = 270 = 1021$ bit for the optimal duration for a statistical attack. In view of these problems, Feistel

points out that an approximation to the ideal block cipher is needed.



Feistel references a general n -bit substitution as a ideal block cipher as it allows for as many encryption mappings as possible, from the plaintext to the ciphertext block. A 4-bit input generates one of 16 probable input states which is translated to one out of 16 probable output states via the substitution cipher, each with 4 ciphertext bits. As shown in the Figure, the encryption and decryption mappings can be specified by tabulation. The demonstrates a small four bit substitution to show that every possible input can be mapped to any output arbitrarily – which is why it is increasingly complex.



- The exact implementation of a Feistel network depends on the selection of the parameters and design features below:
- block size - Improve security and slow ciphers by increasing size •
- key size - increasing size enhances security, makes it difficult to find exhaustive key but can slow down cipher.
- number of rounds - rising numbers enhance security but slow down cipher
- subkey generation algorithm - More complexity will make analyzes more complicated, but it's slowing down cipher process
- round function - More complexity can make analyzes more challenging, but slows cipher
- fast software en/decryption - more recent concern for practical use
- ease of analysis - for easier validation & testing of strength

Claude Shannon and Substitution-Permutation Ciphers

- In 1949, Claude Shannon presented the concept of the substitution-permutation (S-P) paper creation of the existing block cipher chains.

- The two primitive cryptography operations seen before S-P networks are based on:
 - *substitution* (S-box)
 - *permutation* (P-box)
- provide *confusion* & *diffusion* of message & key

Confusion and Diffusion

- • The statistic properties of the original message must be absolutely unknown by cipher
- a one-time pad does this
- Shannon proposed in more concrete terms that S & P elements be combined to obtain:
- **diffusion** – Dissipates the plaintext statistical structure over ciphertext bulk
- **confusion** – Claude Shannon used the terms diffusion and confusion to identify the two basic building blocks for any cryptographic system, which allows for as complex ciphertext and the key relations as possible. The concern of Shannon was to circumvent a statistical analysis dependent cryptanalysis. Each block cipher involves converting a plaintext block into a ciphertext block, which depends on the key. The diffusion mechanism seeks to complex the statistical relationship between the plaintext and the ciphertext in order to counteract efforts to ascertain the key. Confusion aims, again, to tackle efforts to detect the key, to make the relation between ciphertext statistics and the encryption key values as complex as possible. Diffusion and confusion are so effective in obtaining the nature of a block cipher's desired attributes that it is the foundation of modern cipher design.

Overview of the DES Algorithm

Data Encryption Standard (DES) is the most widely used private keyblock cipher. This was adopted as Federal Information Processing Standard 46 (FIPS PUB 46) in 1977 by the National Bureau of Standards. In 64-bit blocks, DES encrypts data with a 56-bit key. The DES is widely used. It was also very controversial for its security.

The DES algorithm combines two basic encryption components: substitution and transposition with an intricate and complex one. The strength of the algorithm is derived from repeated applications, one over the other, of these two techniques for a total of 16 cycles. Until now, researchers in the general public have struggled to notice the sheer complexity of mapping the single bit through 16 iterations of substitutions and transpositions.

The algorithm starts with 64-bit blocks of plaintext encryption. The key is 64 bits long but can be 56 bits as well. If the security of the old key is not known, the user is free to change a key at will. (The extra 8 bits are often used as check digits and do not impact normal implementation encryption.

The algorithm uses the two technologies defined by Shannon to mask the information: confusion and diffusion. In other words, the algorithm does two things. Make sure the output bits do not have a direct association with the input bits and spread the effect of one plaintext bit over another in the ciphertext. Substitution gives rise to confusion and transposition creates diffusion. Usually, plaintext is influenced by a number of substitution cycles followed by a permutation. Iterative substitutes and permutations are done. The total scheme for DES encryption is shown in the figure, which contains 64-bit data and key input.

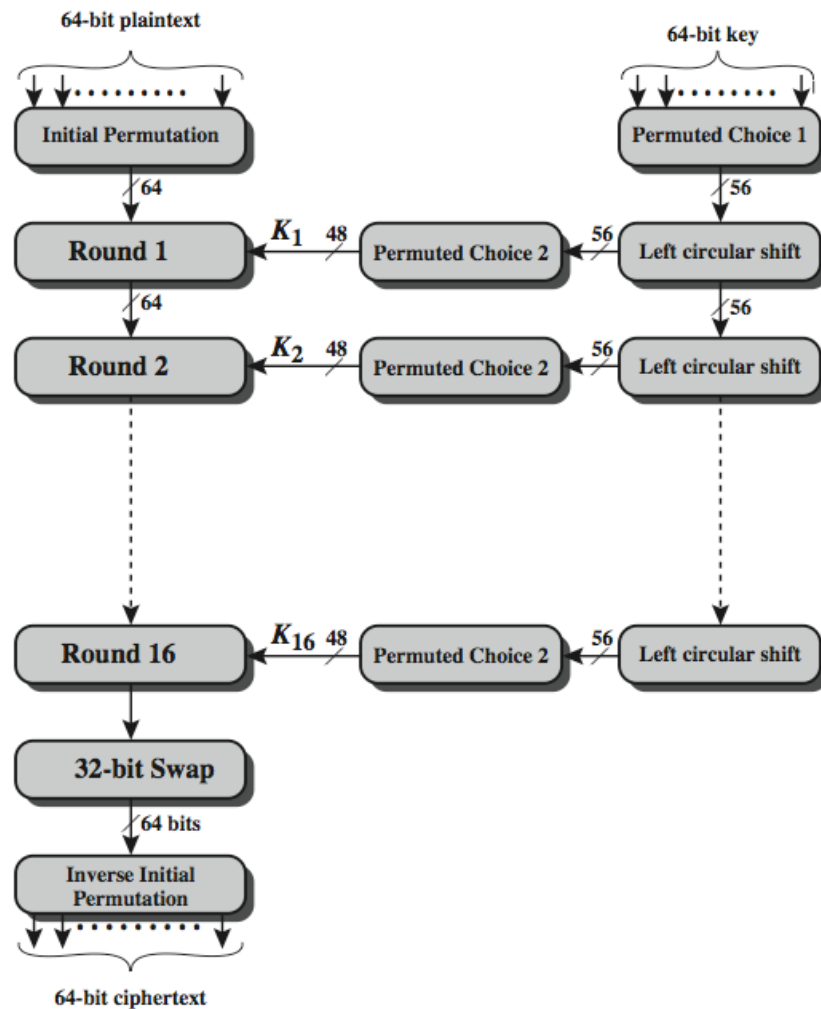
The left side shows the basic process for enciphering a 64-bit data block which consists of:

- An initial permutation (IP) for the input 64-bit block to be shuffled

- 16 rounds of a complex round function with substitutions & permutations
- Final permutation is the opposite of IP

The right side shows the 56-bit key and includes:

- Initial permutation key (PC1), selecting 56 bits in two 28-bit halves of the 64-bit input.
- 16 stages in order to generate the 48-bit subkey with the left circular shift and the two 28-bit halves permutation



Initial Permutation (IP)

- first step of the data computation
- IP reorders the input data bits
- even bits to LH half, odd bits to RH half
- quite regular in structure (easy in hardware)
- no cryptographic value

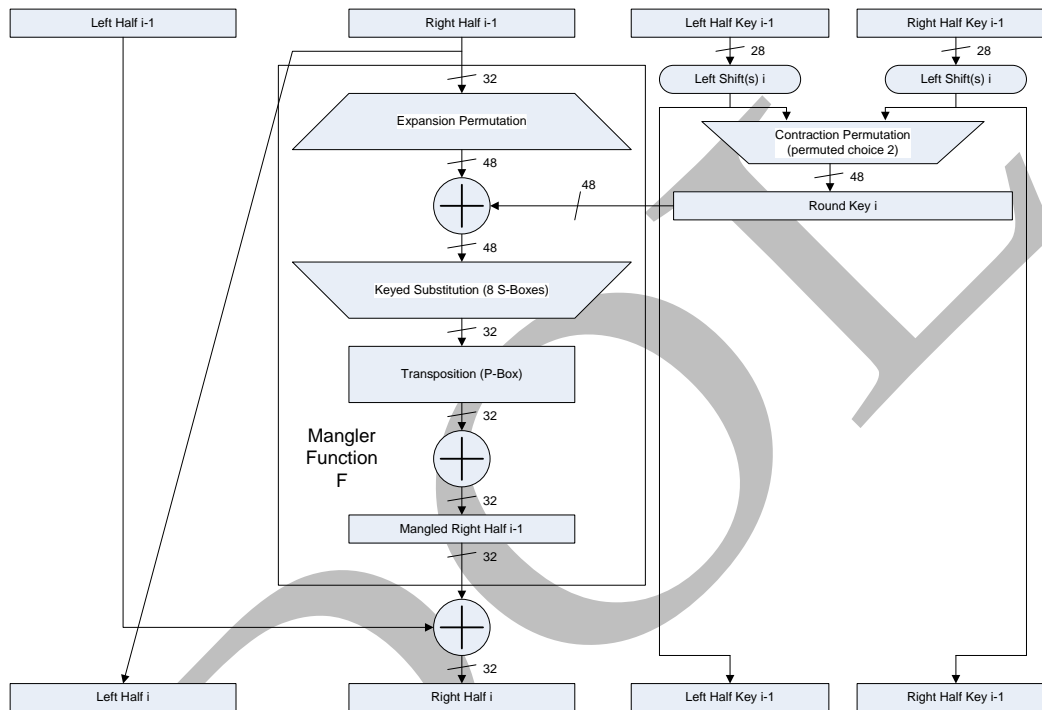
DES Round Structure

- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

- F takes 32-bit R half and 48-bit subkey:
 - expands R to 48-bits using perm E
 - adds to subkey using XOR
 - passes through 8 S-boxes to get 32-bit result
 - finally permutes using 32-bit perm P



DES Key Schedule

- forms subkeys used in each round
 - Initial key permutation (PC1), selecting 56-bit in two 28-bit halves 16 stages consisting of:
 - rotating one or two places separately for each half, depending on the key rotation schedule K
 - to select 24-bits for use in round F function from each half and to permute PC2.
- note practical use issues in hardware vs software

DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again using subkeys in reverse order (SK16 ... SK1)
 - IP undoes final FP step of encryption
 - 1st round with SK16 undoes 16th encrypt round

-
- 16th round with SK1 undoes 1st encrypt round
- then final FP undoes initial encryption IP
- thus recovering original data value

Avalanche Effect

Any encryption algorithm has the desirable property of allowing major cipher text changes by a small change in either the plaintext or in the key .. A change of the plaintext or a bit of the key in particular should lead to changes in many pieces of the ciphertext. If the change is small, it can reduce the plaintext or the key space to be searched. If it is small, DES has a powerful avalanche.

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- brute force search looks hard
- recent advances have shown is possible
 - in 1997 on Internet in a few months
 - in 1998 on dedicated hardware (EFF) in a few days
 - in 1999 above combined in 22hrs!
- still must be able to recognize plaintext
- must now consider alternatives to DES

DES Design Criteria

- as reported by Coppersmith in [COPP94]
- 7 criteria for S-boxes provide for
 - non-linearity
 - resistance to differential cryptanalysis
 - good confusion
- 3 criteria for permutation P provide for
 - increased diffusion

Double DES

Some researchers consider using double encryption for greater security in order to cope with the discomfort. The double encryption works like this. Take two keys, k_1 and k_2 , and produce two encryptions: $E(k_2, E(k_1, m))$ on top of one key. This method will make it more difficult to crack encryption, just as it is easier to pick two locks than one.

Triple DES

This triple DES process is defined as $C = E(k_3, E(k_2, E(k_1, m)))$. In other words, one key use for encryption, a second key is use for decryption and a third key used for encryption. This procedure provides the strength of a 112-bit key (because the DES attack defeats one of the three keys).

A little variant of triple DES, also referred to as triple DES in some cases, is $C = E(k_1, D(k_2, E(k_{1am}))$). This is, one key is encrypted, the second key is decrypted and the first key is encrypted again.

2.6 The AES Encryption Algorithm

The AES Contest

The new encryption system was developed by NIST in January 1997. NIST made several important restrictions, like the call for candidates from which DES was selected. The algorithms were required

- Unclassified
- publicly disclosed
- Available worldwide royalty free
- Symmetric block cipher algorithms, for blocks of 128 bits
- Usable with key sizes of 128, 192, and 256 bits

In August 1998, of the submitted algorithms, fifteen were chosen, while the candidate area was reduced by five in August 1999. The five were closely checked by the public and private sector. The selection was made not only based on security, but also on the cost-effectiveness or operating efficiency of software and easy implementation. Two Dutch cryptographers given the winning algorithm, Rijndael.

Origins

- clear a replacement for DES was needed
 - have theoretical attacks that can break it
 - have demonstrated exhaustive key search attacks
- can use Triple-DES – but slow, has small blocks
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug-99
- Rijndael was selected as the AES in Oct-2000
- issued as FIPS PUB 197 standard in Nov-2001

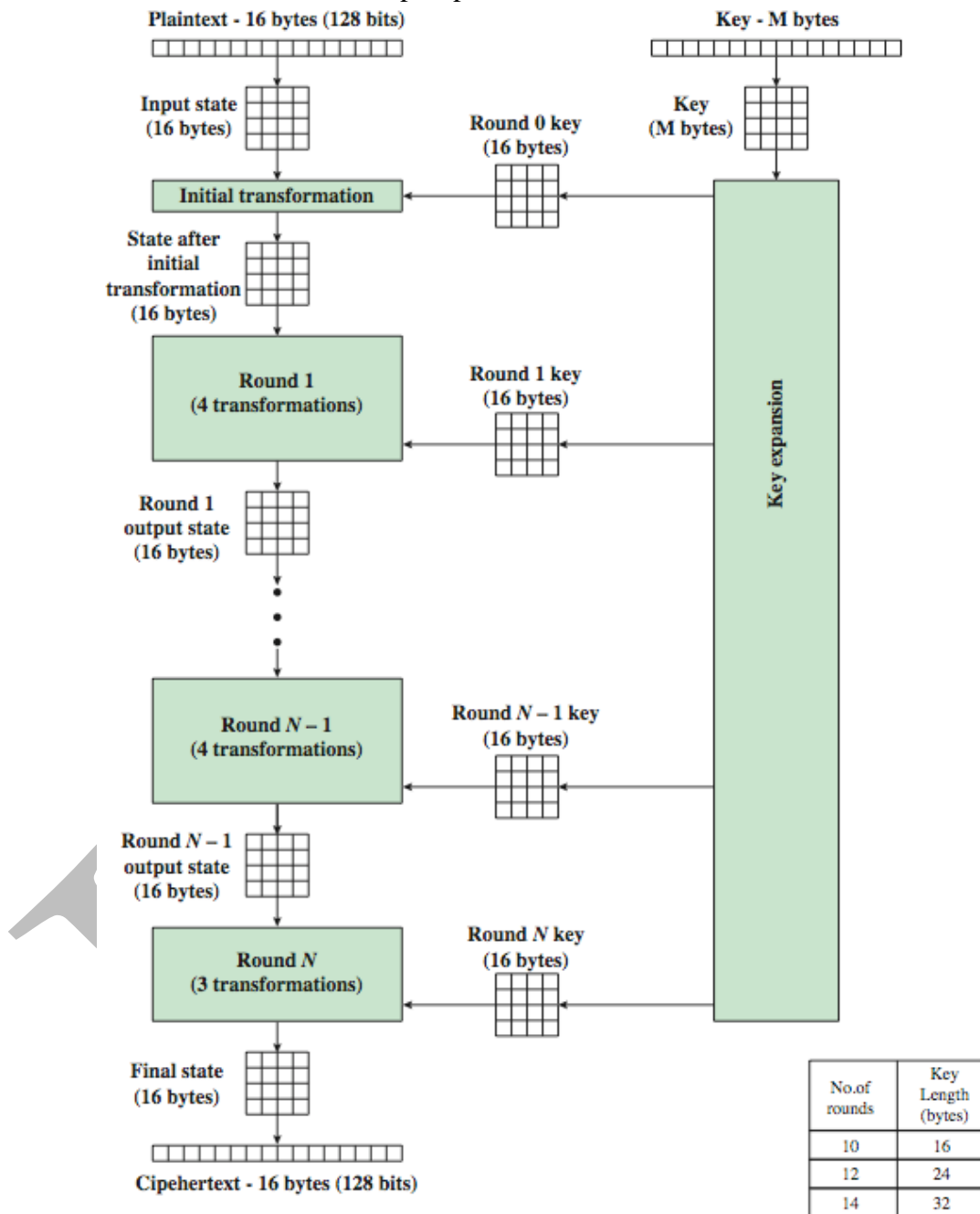
The AES Cipher - Rijndael

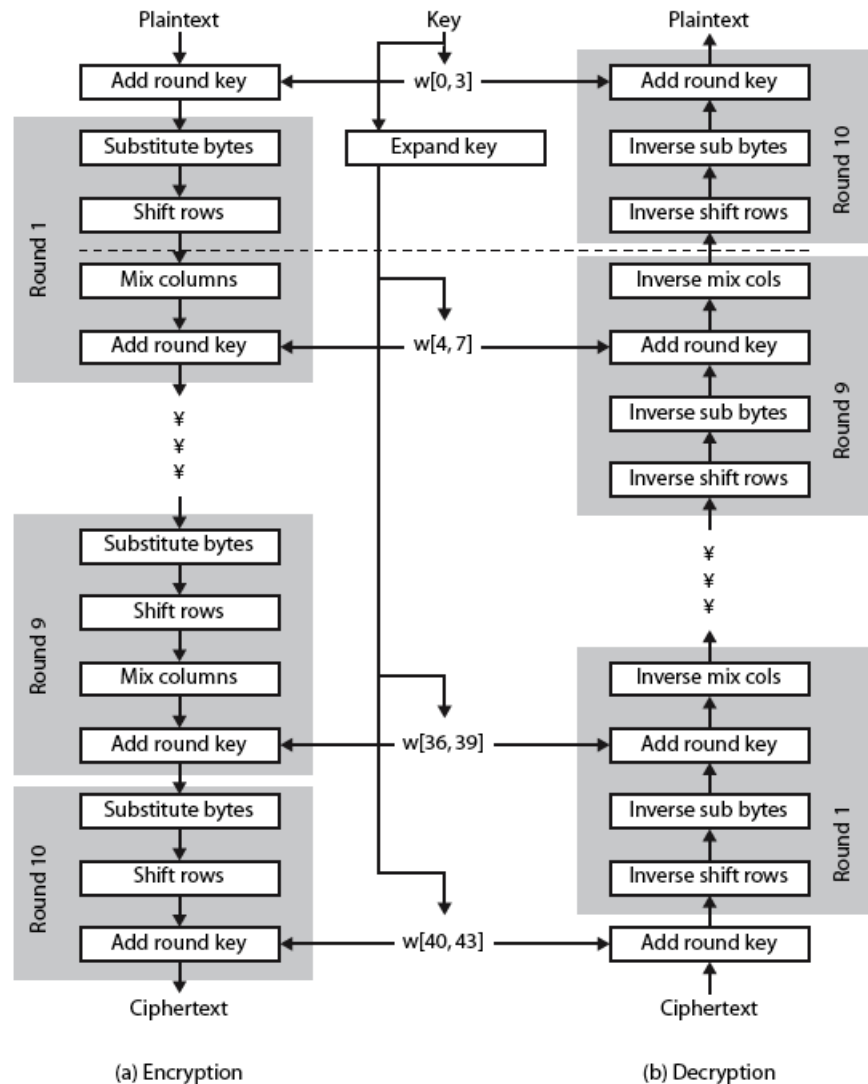
- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **feistel** cipher
 - processes data as block of 4 columns of 4 bytes
 - operates on entire data block in every round
- designed to be:
 - resistant against known attacks
 - speed and code compactness on many CPUs
 - design simplicity

AES Structure

- data block of 4 columns of 4 bytes is state
- key is expanded to array of words
- has 9/11/13 rounds in which state undergoes:
 - byte substitution (1 S-box used on every byte)
 - shift rows (permute bytes between groups/columns)
 - mix columns (subs using matrix multiply of groups)

- add round key (XOR state with key material)
- view as alternating XOR key & scramble data bytes
- initial XOR key material & incomplete last round
- with fast XOR & table lookup implementation





1. an **iterative** rather than **feistel** cipher
2. key expanded into array of 32-bit words
 1. four words form round key in each round
3. 4 different stages are used as shown
4. has a simple structure
5. only AddRoundKey uses key
6. AddRoundKey a form of Vernam cipher
7. each stage is easily reversible
8. decryption uses keys in reverse order
9. decryption does recover plaintext
10. final round has only 3 stages

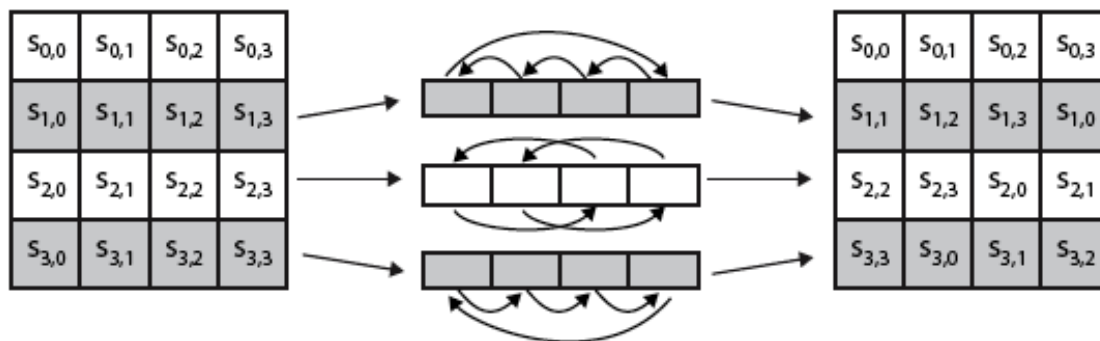
Substitute Bytes

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values

- each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
 - eg. byte {95} is replaced by byte in row 9 column 5
 - which has value {2A}
- S-box constructed using defined transformation of values in GF(28)
- designed to be resistant to all known attacks

Shift Rows

- a circular byte shift in each each
 - 1st row is unchanged
 - 2nd row does 1 byte circular shift to left
 - 3rd row does 2 byte circular shift to left
 - 4th row does 3 byte circular shift to left
- decrypt inverts using shifts to right
- since state is processed by columns, this step permutes bytes between the columns



87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

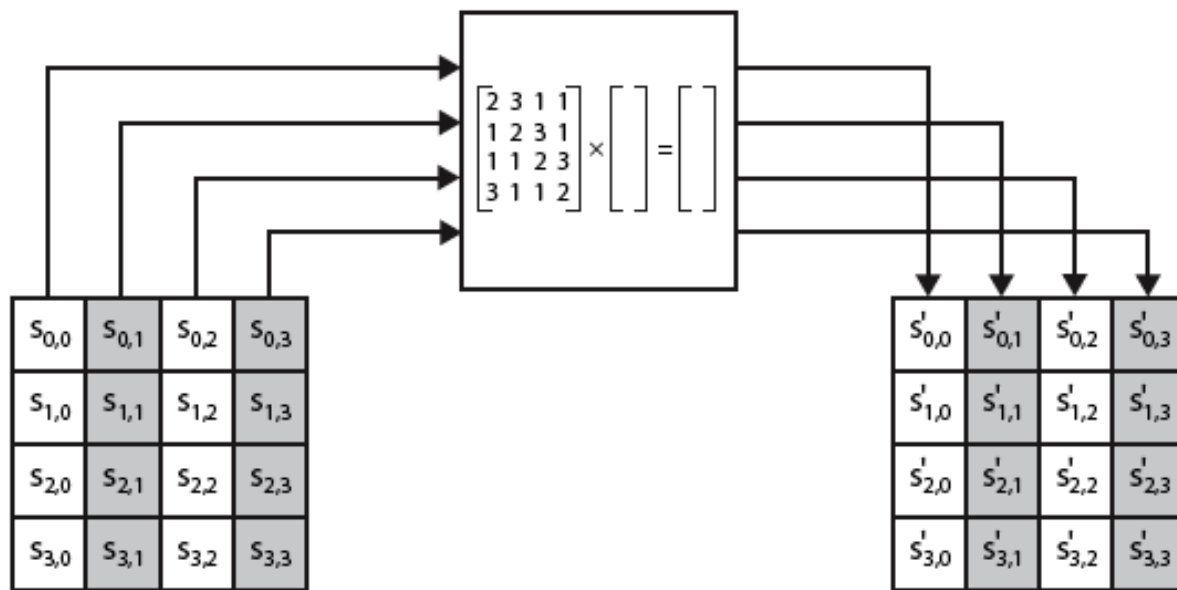
➔

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Mix Columns

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in GF(28) using prime poly $m(x) = x^8 + x^4 + x^3 + x + 1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}
 \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}
 =
 \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$



87	F2	4D	97		47	40	A3	4C
6E	4C	90	EC		37	D4	70	9F
46	E7	4A	C3		94	E4	3A	42
A6	8C	D8	95		ED	A5	A6	BC

$$(\{02\} \cdot \{87\}) \oplus (\{03\} \cdot \{6E\}) \oplus \{46\} \oplus \{A6\} = \{47\}$$

$$\{87\} \oplus (\{02\} \cdot \{6E\}) \oplus (\{03\} \cdot \{46\}) \oplus \{A6\} = \{37\}$$

$$\{87\} \oplus \{6E\} \oplus (\{02\} \cdot \{46\}) \oplus (\{03\} \cdot \{A6\}) = \{94\}$$

$$(\{03\} \cdot \{87\}) \oplus \{6E\} \oplus \{46\} \oplus (\{02\} \cdot \{A6\}) = \{ED\}$$

- can express each col as 4 equations
 - to derive each new byte in col
- decryption requires use of inverse matrix
 - with larger coefficients, hence a little harder
- have an alternate characterisation
 - each column a 4-term polynomial
 - with coefficients in GF(28)
 - and polynomials multiplied modulo (x4+1)

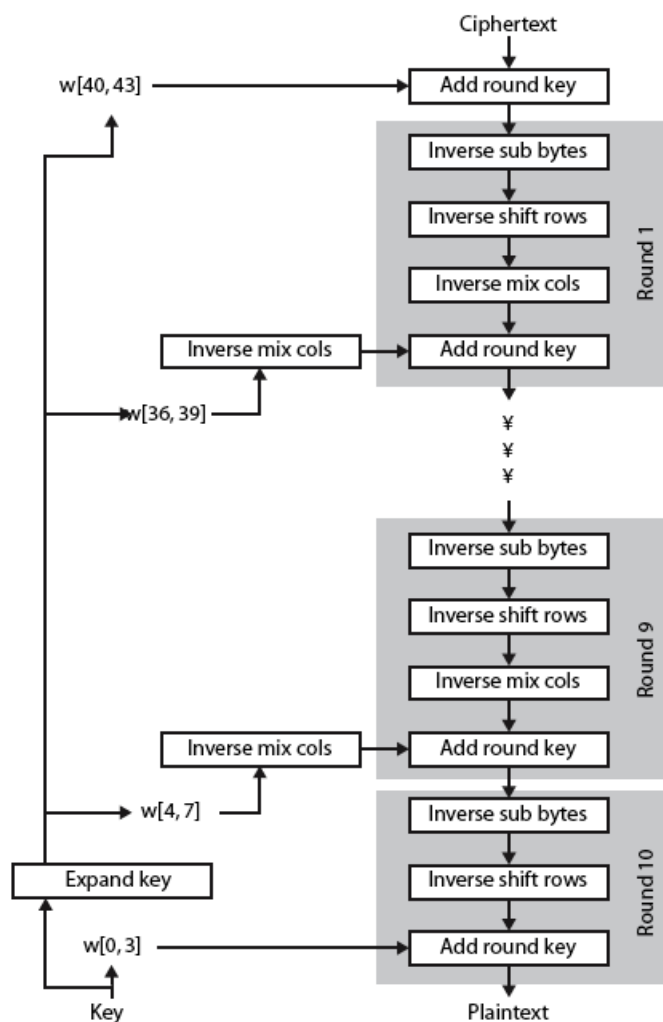
- coefficients based on linear code with maximal distance between code words

Add Round Key

- XOR state with 128-bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption identical
 - since XOR own inverse, with reversed keys
- designed to be as simple as possible
 - a form of Vernam cipher on expanded key
 - requires other stages for complexity / security

AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- but can define an equivalent inverse cipher with steps as for encryption
 - but using inverses of each step
 - with a different key schedule
- works since result is unchanged when
 - swap byte substitution & shift rows
 - swap mix columns & add (tweaked) round key



Comparison of DES and AES

	DES	AES
Date	1976	1999
Block size	64 bits	128 bits
Key length	56 bits (effective length)	128,192,256 bits (and possibly more)
Encryption primitives	Substitution, permutation	Substitution, shift, bit mixing
Cryptographic primitives	Confusion, diffusion	Confusion, diffusion
Design	Open	Open
Design rationale	Closed	Open
Selection process	Secret	Secret, but accepted open public comment
Source	IBM, enhanced by NSA	Independent Dutch cryptographers

Public Key Encryption

Diffie and Hellman introduced a new type of encryption system in 1976. Each user has a public key encryption system, which must not be kept secret. Although the public nature of the key does not compromise the secrecy of the system, although this is contra-intuitive. The basis for public key encryption is, instead, to disclose the key but to keep decryption technique secret. This objective can be achieved by using two key public key cryptosystems: one for encryption and the other for decryption.

Asymmetric or public key encryption systems use two keys, a public key and a private key. Unfortunately, a few people call a symmetric or secret key system a “private key “system. This terminology is confusing. We do not use it in this book, but you should be aware that you might encounter the terminology in other readings.

Every user has two keys in a public key or an asymmetrical encryption system: a public key and a private key. The user should freely publish the public key as the encryption and decryption process only takes place by half each key. The keys act as inverse keys, which is, that one key undoes the encryption provided by other .

Let k_{PRIV} be a private key for a user and let k_{PUB} be the appropriate public key. Instead, by using a Public Key, the encrypted plaintext is decrypted by using a private key; the relationship is written as: $P = D(k_{\text{PRIV}}, E(k_{\text{PUB}}, P))$ In short, a user will decrypt with a privately kept key what someone has encrypted with a public Key. In addition, we have this relationship with other primary public encryption algorithms, like RSA:

$$P = D(k_{\text{PUB}}, E(k_{\text{PRIV}}, P))$$

In other terms, the user can encrypt a message using a private key, and only the corresponding public key will expose the message. Both properties inform us that private and public keys can be used as they want. The decryption function D can be used in particular to decrypt any statement before we encrypt. We often talk about decryption before encryption with traditional

encryption technique. But the statement makes sense with public keys, where it only means first private transformation take place and then public transformation.

We found that symmetric encryption presents a significant problem because a single user has to store and track number of keys. Only two keys per user are provided with public keys: a private and a public. Let's see how much the number of keys required makes this difference. Suppose we have three users, B, C, and D, who will give user A and each other secure messages. Since every user pair requires a key, every user will need three keys; for example, A would need a key for B, a key for C and a key for D. However, each B, C, and D will encrypt messages for A using public key A using public key encryption. If B is using the public key A to encrypt the message, C will not decrypt it, given that C is aware of the public A key to encrypt it. For instance, twice applying A's public key will not decrypt the message. So, the number of keys required in the public-key system is fairly small. (We presume, of course, that A's private-key remains secret)

Comparing Secret Key and Public Key Encryption:

	Secret Key (Symmetric)	Public Key (Asymmetric)
Number of keys	1	2
Protection of key	Must be kept secret	One key must be kept secret; the other can be freely exposed
Best uses	Cryptographic workhorse; secrecy and integrity of data single characters to blocks of data, messages, files	Key exchange, authentication
Key	Must be out-of-band	Public key can be used to distribute other keys
Distribution Speed	Fast	Slow; typically, 10,000 times slower than the secret key

2.7. Public Key Encryption

We have seen the encryption algorithms so far from the assumption that it is easy to scrap (because the sender can literally crypt a note) and that the receiver can decrypt the bote, but it does not have an attacker. This useful view of plaintext change to ciphertext . The role of encryption function should also be examined. We noticed that the useful keys can be used to prevent the intruder, but we presumed that the key must stay the key in order to work. In this field, we use methods in order to ensure that the primary is the general population. However, the meaning is still secure. We also focus on the open-key system RSA algorithm which is obviously a favored approach for commercial encryption.

A new form of encryption scheme was proposed by Diffie and Hellman in 1976. Include a primary public encryption technique, each individual has a key that is not magical formula. In reality the dynamics of the main variable do not negotiate the confidentiality of the system, though contrary to the definition. Instead, essential public encryption will be focused on the

disclosure of the data, but on keeping the decryption technique confidential. Public key cryptosystems accomplish this goal by using two tips: one to encrypt and the other to decrypt.

Motivation

Why would it be attractive to deliver the key people? Every couple of users needs a separate key with the standard symmetrical key techniques. With general population key techniques, however, someone who uses a single primary public may submit a hidden message to an end user, and information is shielded from an interceptor's study. Why are we not investigating why that is so.

Note that an n-user technique typically involves $n*(n-1)/2$ keys, and every customer needs to track a key for any other end user he or she wants to speak to. The number of users rises extremely quickly, as seen in Figure 2.10. The amount of keys increases. It is a challenge to define and spread these secrets. Seriously, the protection for the tips already sent is typically retained, since customers can not expect to store several key.

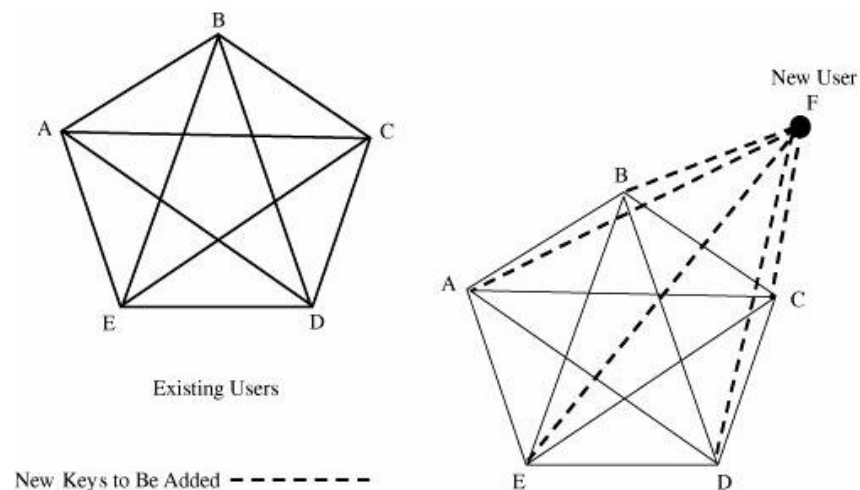


Figure 2.10. Key Proliferation.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Characteristics

Using a public key technique we can reduce the problem of crucial proliferation. Every individual has two secret keys with common major or asymmetric encryption technique: a common key and an exclusive key. An individual can publish the general public key openly as each key is encrypted and decrypted by only 1/2 of the procedure. The keys run as opposite, therefore the encryption given by the other key factor is inverted on an important basis.

To see how k_{PRIV} can be treated as a primary private user and allow k_{PUB} function to act as an adequate primary public. After this, the program of the exclusive key will decrypt encrypted plaintext using the public key;

$$P = D(k_{\text{PRIV}}, E(k_{\text{PUB}}, P))$$

In other words, a customer can decipher what another person encrypted with the related public with an exclusive key. In addition, we have this partnership with certain open main encryption algorithms which integrate RSA:

$$P = D(k_{\text{PUB}}, E(k_{\text{PRIV}}, P))$$

An user can encrypt a message with an exclusive key and the message can only be shared with the specific primary public. The two houses reveal that public and private tips could be used for each purchase. Specifically, any statement may be put with the decryption functionality D so that we can decrypt before we encrypt. We never think about decrypting before encrypting with standard encryption. The definition is, however, realistic with public secrets where it ultimately requires first and foremost private and general public transition.

We know that the large number of keys that each user has to store and observe can be a major problem with symmetric encryption. Only two tips per person are relevant with public tips: one person and one exclusive. Let us find out what the difference in the number of keys that this produces. Suppose that we have three users, B, C, and D, which must be supported by protected text messages to User A. Every consumer would like three different tips, because each specific customer pair wants a major one; for instance, A would like an integral for B, an integral for C and an integral for D. However, both B, C, and D can encrypt information for A by using A's public primary by using public key element encryption. If B has encrypted a message with an open major of A, C can not decrypt it, given the fact that C knew it was encrypted with an A's population key. , for example, with the use of A 's public key element twice, information will normally not be decrypted. So the number of keys needed in a public key system in general is certainly fairly small. (We presume, offline, that A's exclusive key remains hidden knowledge.)The characteristics of the secret key and public key algorithms are compared in Table 2.5.

Table 2.5. Comparing Secret Key and Public Key Encryption.

	Secret Key (Symmetric)	Public Key (Asymmetric)
Number of keys	1	2
Protection of key	Must be kept secret	One key must be kept secret; the other can be freely exposed

Best uses	Cryptographic workhorse; secrecy and integrity of data single characters to blocks of data, messages, files	Key exchange, authentication
Key distribution	Must be out-of-band	Public key can be used to distribute other keys
Speed	Fast	Slow; typically, 10,000 times slower than secret key

RivestShamirAdelman Encryption

A public key method is the **RivestShamirAdelman(RSA)** cryptosystem. This algorithm was unveiled in 1978, which was based on an underlying tough problem and named following the three inventors. RSA is the key subject of detailed cryptanalysis and there are yet to be any significant flaws. Although the quantity of the study can not guarantee the security of the system, after a while we have no problem finding our self-confidence in the technique.

RSA depends on where mathematical scientists research the attributes of the number theory including, for instance, prime variables, . The RSA encryption algorithm incorporates quantitative theory results and the number of problems in deciding the prime components of the verified number. The RSA algorithm also works with arithmetic mod n, as a few of the other algorithms we have learned.

For decryption and encryption, both secrets found in RSA, d and e are used. In reality they are compatible: either may be chosen, for the major public, however one owning is preferred. For convenience, we contact the major encryption e plus the key element d for decryption. The tips can also be used in any order due to the nature of the RSA algorithm:

$$P = E(D(P)) = D(E(P))$$

(The two complementary capacities of E and D can be considered, which all "undoes" another.)

Any P-block in plaintext is usually encoded as $P^e \bmod n$. Because exponentiation is performed mod n, it is definitely difficult for P_e to identify encrypted plaintext. The primary d decryption,

however, is carefully chosen to $(P^e)d \bmod n = P$. Therefore, the legitimate recipient who can conveniently compute $d(P^e)d \bmod n = P$ and recover P and never have to factor P^e .

The encryption algorithm is based on the fundamental problem of factoring good sized quantities. To date, no one has found a simple or shortcut to point large numbers into a finite set known as a discipline .. Boneh criticized all of the known cryptanalytic disorders of RSA in extremely technical yet excellent papers and concluded that nothing is significant. Given the factorization issue can be open for quite a while, a good basis for a secure cryptosystem is considered by most cryptographers.

2.8. The Uses of Encryption

Encryption algorithms are not the solution for all everyone's encryption desires. Although encryption implements protected channels of communication, it is often used for specific tasks. In addition, symmetrical and asymmetrical encryption usually capitalizes on each one's best top features.

Public key algorithms are of only assistance for specific tasks because they are very tricky. An encryption of public key element may take 10,000 times as long as the root modular exponentiation relies on multiplication and division. This is inherent in the process of being more slow than the bit functions (addition, exclusive Or, substitution, and shifting), the symmetric algorithms that occur. As a consequence, symmetric encryption may be "workhorse" by the cryptographers, and general public encryption key is generally reserved for limited uses where slow operation does not present a permanent problem.

Let us be more vigilant with four encryption software: hash features, key exchange, electronic digital signatures and certificates.

Cryptographic Hash Functions

Encryption is often used for secrecy; we usually encrypt it in a manner that will not know its contents or maybe its presence by anyone, even a privileged crowd. However, integrity is in some cases a more serious problem than secrecy. Nevertheless, integrity is in some cases a much more critical issue than secrecy. Of one, it is crucial to realize with a doc recovery system, with legal documents, that the backup is precisely what was found. Likewise, the need for the right exchange of information will overcome confidentiality concerns by a very reliable mode of communication. Look at how integrity is given by encryption.

Sun and rain or the different parts of the file are not linked at all in most data files. That is, every byte or tad or character in the record can be independent from each other. The lack of agreement ensures that modifying one attribute will harm the integrity of the file, but a certain improvement can always not be accepted.

Anything we intend to do somehow should be locked, or secured by the document to help us know that everything is modified if the seal is broken. This principle can be compared to the medieval use of painted seals; if the polish was damaged, the receiver would realize that someone was destroying the seal and was reading at the document. Likewise, cryptography can

be used to lock a document and encase it to make any modification apparent. The following is the case. Another way of seal distribution is to measure a cryptographic function, also called a hash or checksum or digest of messages with the file.

The hash has unique characteristics. For example, a function that is easy to see but difficult to compute defines such encryptions. Think of the cube function, $y = x^3$, for an example. You can measure x^3 with pencil and paper or use the calculator. It is not too complicated for you. The reverse function $\sqrt[3]{y}$, however, is much more difficult to calculate. And the function $y = x^2$ has no reciprocal functionality as two possibilities for \sqrt{y} : $+x$ and $-x$ are valid. Such capabilities are usually called one-way features, which can be calculated more easily than their inversions.

Inside an encryption algorithm a one-way feature may be useful. Both objects in the file must be decided by the event, and every change to a successful single specific object changes the checksum consequence. The price of the checksum is stored with the file. Then the checksum will be recomputed any time the file is definitely accessed or used. If the computed checksum is sufficient for the retained value, the document is unchanged.

A cryptographic function such as the DES or AES is especially appropriate for closing criteria, because an outside entity does not know the main element and thus can never change the stored benefit to match the data that are updated. Algorithms that are even easier than DES or AES may be used with low-threat applications. For eg, by using an OR special to merge the encrypted earlier block with all the encryption of the present, the chaining method links each stop with the price of the previous block (and thus to all earlier blocks) for stop encryption strategies.

MD4, MD5 (in which MD means Message Digest) and SHA / SHS (Secure Hash Algorithm or common) are the trustworthy cryptographic hash functionality. Ron Rivest and RSA Laboratories developed the MD4/5 algorithms. MD5 may be an upgraded MD4 variant. Both condense a documents of any size into a 128-bit break down. SHA / SHS is identical to MD4 and MD5 it digests a 160-bit.

Cryptanalysis episodes on SHA, MD4, and MD5 have been announced by Wang et al. For SHA, two concerns can be found that produce exactly the same hash process in approximately 263 phases; there are very strong 280 methods that your 160-bit hash function may anticipate, and incredibly easy for an sufficiently financed attacker. While SHA is not useless (the attacker will compile and evaluate several ciphertext examples), this strike will require the use of very long digest and extended tips. NIST has thoroughly reviewed the episode and has suggested countermeasures.

Key Exchange

Suppose that you will include someone you don't know and don't find out about a protected subject. This example is more common than you could think. For instance, you may send your income tax back to the government. However, you do not know who gets the information. You do not need the information to be covered. You may also use your online browser to connect to a purchasing web site, e-mail business people (encrypted) or ask for two hosts to determine a safe route. Each of these situations will depend on the ability to swap a type of encryption in such a way that nobody else can stop it. It is difficult and necessary to discuss two previously unknown meetings which share cryptographic secrets.

Cryptography that is vital to the public might help. As asymmetric secrets are present, one-half of this pair can be revealed, and the other half can not be reduced. To see how, imagine the desire for a distributed symmetric key by S and R (our well-known sender and device). Also imagine that both S and R contain the public secrets for the standard algorithm of encryption; for any private and general public tips for the S and R, please contact kPRIV-S, kPUB-S, kPRIV-R, and kPUB-R. The easy way of selecting any symmetrical essential K and delivering $E(kPRIV-S, K)$ to R is generally S. Then R needs S's big public, eliminates encryption and gets K. Unfortunately, every eavesdropper that can get S's public key can also get K.

Enable S to add $E(kPUB-R, K)$ to R instead. Only R will decode K then simply. However, R has no assurance that K came from S.

There's a stronger alternative, however. We can consider this change with regard to lockboxes and tips. The answer is S: $E(kPUB-R, E(kPRIV-S, K))$. If S decides to actually submit something to R (such as a selection of credit cards or a number of medical details), then the exchange works like this. S inside a lockbox places the protected data that can only be opened with S's common key. This is surely then put in the second lock box, which can only be opened with R's personal key. R may then take advantage of his exclusive main to open the outer box (something that he can do) and use S's public to open the inner box (showing that the deal was born from S). The basic protocol simply bundles up the protected data in two offers: the first unwrapped only by the S public key and the next unwrapped only by the exclusive R key. This cycle is shown in Figure 2.11..

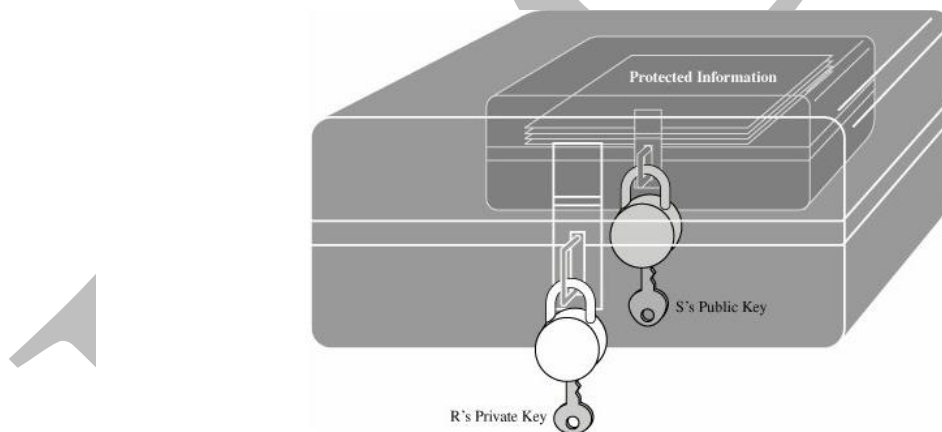


Figure 2.11. The Idea Behind Key Exchange.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The so-called DiffieHellman key exchange protocol can provide another solution that does not necessarily require pre-shared group keys. S and R use a very simple arithmetic to change a secret key for this process. It recognizes both a field dimension n and an initial number g ; it can express these numbers in the apparent. Each believes the number, state, s and r is top secret. S is transmitted by R g^s and R by S g^r . Following the S $(g^r)^s$ and R $(g^s)^r$ equations that will be the same, $g^{rs} = g^{sr}$ transforms into their shared secret keys.

Digital Signatures

Another issue that usually involves a normal human: a purchase to move money from one person to another. Simply put, you want the ability to send electronically as a computerized take. We know how to handle this transaction in the traditional paper function: a check is indeed a concrete item which permits the monetary transaction.

The signature in the checkout confirms authenticity as only the reasonable signator can (presumably) create the signature.

With regard to an suspected forgery, an approved individual may be expected to determine authenticity.

If a test is reviewed, it is cancelled to prevent it from being repeated.

The look of the paper is immutable. Or practically all sorts of changes are often observed.

Transaction organization by check is subject to the prescription forms of tangible objects. But for computer transactions tangible items can not be found. Thus, it takes a different model to allow repayments by computer. Why should we not take into consideration the preconditions for this kind of case, both from the standpoint of every bank and from the standpoint of your user? Suppose Ramesh transmits her bank a note authorizing it to copy Rs 10000 to Suresh. Ramesh's bank or investment firm must be able to verify and confirm that the communication was really from Ramesh if it should denounce the message shortly after it is sent out. The lender would also like to note that Ramesh's message is certainly absolutely that this is not modified. Ramesh also needs to ensure that all of these communications can not be fabricated by its loan provider. Both functions want to ensure that the message is fresh not just to reuse an old message, and that the message does not alter when sending it. This approach is complicated by the use of electronic alerts rather than documents.

Meanwhile we have forms to get the task done. A digital signature banking system is a standard protocol with much the same effect as a genuine signature: only the sender may do this, but other people can easily recognize as a sender. A real signature can be used to confirm the contract to a note by an electronic signature.

Properties

In two prime circumstances, A digital signature bank should connect: it should be imperfect. If individual P signs subject M with trademark $[M, S(P, M)]$, it is impossible to create the match for anyone else $[M, S(P, M)]$.

It ought to be true. If a person R gets the match $[M, S(P, M)]$ allegedly from P, R can be sure that the signature is from P. This trademark can have only been created by P and the trademark is firmly attached to M.

These two requirements are the main obstacles in laptop transactions shown in Figure 2.12. For agreements completed with the support of digital signatures, two more properties that are also attracted by parallels with the paper based environment will also be desirable:

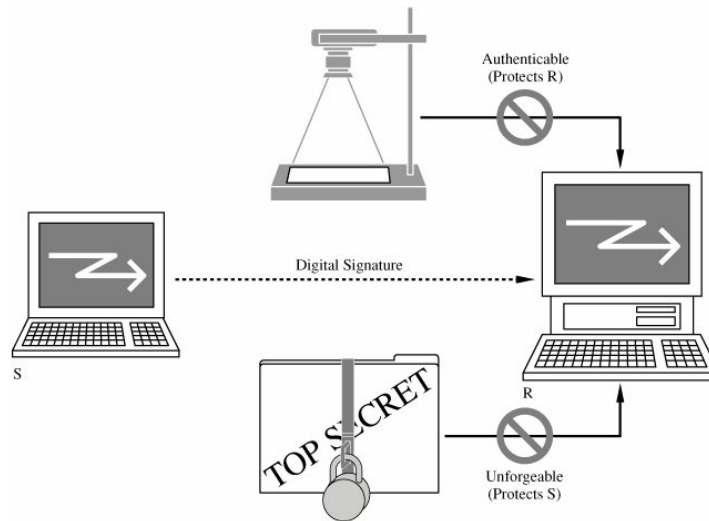


Figure 2.12. Requirements for a Digital Signature.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

It's not changed. M can not be formed by S, R, or an interceptor when it is transmitted.

It can not be reused. R will immediately recognize an earlier message once again.

We first offer a device that meets the basic 2 criteria to test how digital signatures operate. To satisfy various other conditions, we add to the procedure.

Major public encryption schemes usually are best suited for digital electronic signatures. How should we not presume that the general public key U is shown in $E(M, KU)$ and that each key shift in U is written as $D(M, KU)$? For very simple notations. We can think of E as privacy changes (as only U can decrypt it), and D as authenticity changes (as only U can develop it). However, remember that with some asymmetric algorithms such as RSA, D and E, each message can be commutative.

$$D(E(M,),) = M = E(D(M,),)$$

S uses authenticity changes to create $D(M, KS)$, if S is intended to mail M to R. S transfers $D(M, KS)$ to R in this situation. R can decode the concept to modify S, $E(D(M, KS), KS) = M$ with a general public key. Since S can simply make an impression message in $E(, KS)$, it goes without saying that S is the result of the communication. This assessment meets the credibility criterion.

R will save you $D(M, KS)$. If soon after S alleges that this message is actually a forgery, R can only contain M and $D(M, KS)$. Since $M(KS)$, with general public key change to S but, is certainly changed to M, everyone can confirm that $D(M, KS)$, definitely $D(M, KS)$ should be from S. In this case $D(M, KS)$. This power meets the unforgivable requirement.

Some methods for using digital signature banks are additional; some use symmetrical encryption, while others use asymmetric methods. The following approach demonstrates the manner in which the protocol will address unique unforgiveness and authenticity criteria. In addition, $E(M, KR)$, as seen in Figure 2.13, can be used to provide secrecy.

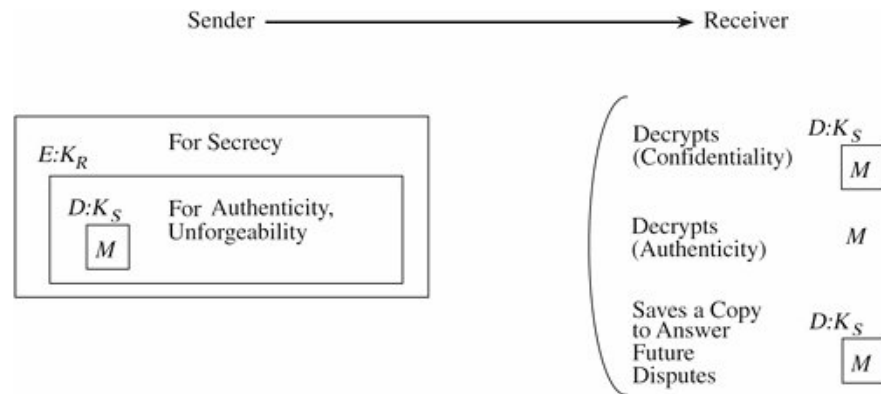


Figure 2.13. Use of Two Keys in Asymmetric Digital Signature.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Certificates

As individuals, we trust our daily experiences with people on a regular basis. By realizing their voices, conversations, or writing, we identify men and women. We use an association to list trust on certain occasions. For example, if someone who is an unknown telephone us and we note that "I represent the federal neighborhood ..." or "I'm contacting behalf of the the charity ..." or that "I'm calling in the university/hospital/police about your mom/father/son/daughter/brother/sister...", we that choose to believe the caller even if it isn't known to us. Regarding the essence of the decision, we may continue to consider or even request independent proof of the caller 's affiliation. for instance has the range of the affiliate from the call directory to the party. If we have small trust in us, we can even do something about excluding outsiders, such as: 'I'm going to send you the right to your charity instead of giving you my billing number.' Or maybe we will find out more through callers such as: ' What color coat did you use?' or "Who's the president of one's corporation?" For each interaction we have what we can call a 'trust threshold;' In commercial interactions, too, this level prevails. If Acorn Production Company supplies Big Steel Corporation with a purchase of 10,000 sheets of stainless steel, delivered within a week and covered ten times, they trust in abundance. The purchase shall be printed in Acorn form, which is agreed by somebody defined as the procurement agent Helene Smudge. Perhaps before receiving cash from Acorn Big Metal can begin to organize steel. In order to decide whether to dispatch purchases without payment at first, Big Metal will review Acorn's credit history. If uncertain, Big Metal might call Acorn and

discuss the purchasing portion with Ms. Smudge. But even more likely, Big Steel will already ship the goods without figuring out who Ms Smudge is, if she is the customer, whether she can actually invest in such a huge order and whether the trademark is actually hers. Such a transaction also occurs by fax, because Big Steel does not have a genuine data file signature. In this case, trust is dependent on the whole look of authenticity (similar to the printed, signed type), on details from outside (similar to a credit file), and on urgent procedure (the demand of Acorn to send metallic quickly).

In order to succeed in electronic contact, we should build similar methods to create trust without fulfilling two meetings. The capacity to include a person or something that can at least confirm the lifetime and integrity of 1 or both people may be a standard thread within our personal and company interactions. The officials, the Business Chamber or the Senior Business Bureau guarantees the caller 's authenticity. Acorn implicitly states that Mrs Smudge is definitely her assistant in the buying department by handing the decision on to she. Somehow, the telephone company guarantees that a celebration is authentic by listing it in the directory. This idea of "building on" by means of a third party could be a basis for relying on commercial adjustment, when two events have no idea. There can be many businesses in a large company, various departments in each department, different workplaces in each department and more than a few task communities in each of them (with variations in names, levels and completeness). Although the best executive may not be familiar with the label or look each worker in the company, the leader of the group understands all the duty team members. This hierarchy can become the foundation for trust in the entire organisation.

There may be a large organization with various divisions, a few departments for every segment, various tasks for each team, and more than a few tasks (with changes in titles, ranks, and level of completeness of hierarchy) for each job. The leader in the business group has learned all the users of the service group, the leader knows all the task group leaders and so forth by brand or vision, perhaps not all workers within the business. The whole organization can become the foundation of trust in this hierarchy.

To see how two different people accomplish themselves, assume: Ann and Andrew. For the same company as Ann, Andrew says it does. Ann wants to make an independent search. She realizes that the Costs and Betty are two task team leaders, Anne performs for Invoice and Andrew for Betty, who is precisely the same project (guided by Camilla). Such information give Ann and Andrew a basis on which to trust each other's I d (the organizational connections can be found in Figure 2.14). The test string can be something like this:Ann asks Charge who Andrew is definitely.

- Charge either asks Betty if he is aware of her immediately or or even, asks Camilla.
- Camilla asks Betty.
- Betty replies that Andrew performs for her.
- Camilla tells Charge.
- Bill conveys to Ann.

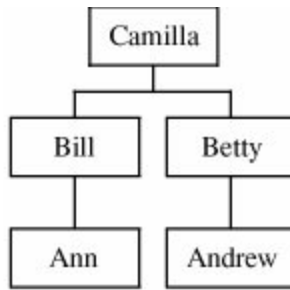


Figure 2.14. Organization in Hypothetical Company.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

If Andrew is in another activity group, it is possible that a standard point must be increased in the organization tree.

For cryptographic key elements exchange we can work with a similar method found in Figure 2-15. If Andrew and Ann want to connect in this way, Andrew should provide Betty, who goes to Camilla or Bill, who gives her to Ann, with his important public. But it's not the way this collection can work in real life. The key should be followed by a note that states it is actually from Andrew, which varies from a yellow paper to a 947 ID statement submission. In the future, Betty would also apply to an application 632a Transmittal of Individuality if an application 947 could be used, Camilla should bind another 632a and Costs would link to the last one, as seen in Figure 2 15. The 632a range will basically mean, "I am Betty and I acquired this key along with the associated identity confirmation of a specific person I know to be Andrew," "I'm Camilla and I got this key plus the attached identity declaration and the associated transmittal of id personally from the person I understand to become Betty" etc. When Ann gets the main element, she may criticize the data string and conclude with an inexpensive guarantee that Andrew actually created the main element. This process is a way to get authenticated popular tips, a connection between an essential and a trustworthy identity.

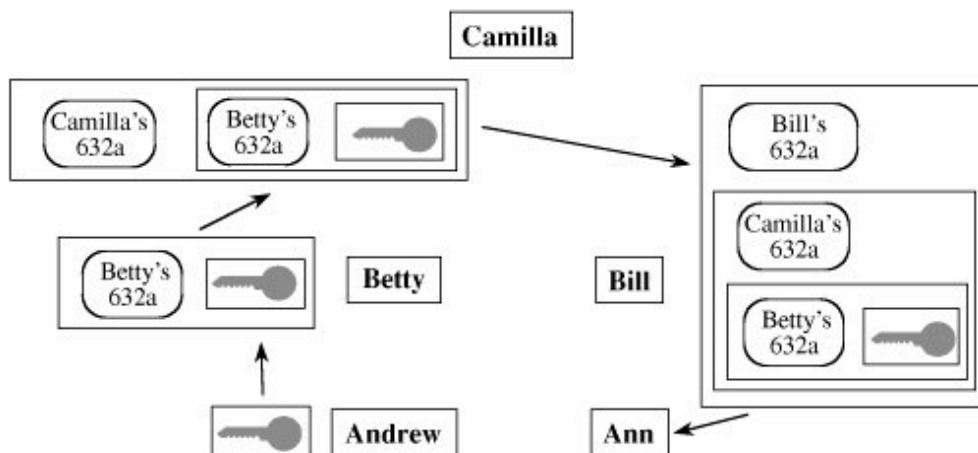


Figure 2.15. Andrew Passes a Key to Ann.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

This model works in a business, as two employees are always often employed, even though both employees come in various divisions so that the individual is chairman. The process goes even further if, if Andrew and Ann wish to speak, Ann, Invoice, Camilla, Betty and Andrew have to be obtained. Where Betty usually goes on a trip or costs are unhealthy, then the procedure fails. Moreover, if the president sees no significant job to be done, it can not function correctly, since form 632a can be handled everyday.

Andrew can demand his full range of 632a styles from the President right down to him in order to resolve the original complications. Then Andrew can give anybody in the business who wants his key a duplicate of the entire set. Andrew starts at the very top and extracts his entire chain from the bottom up to the common level. It becomes these signatures every time you find your superiors, so they don't have to be reached if you really want to hand over your authenticated public key.

The second question is to reverse the process. We start at the very top rather than start from the bottom (with task participants) and seek the top in the tree (leader). Andrew also provides a pre-authenticated public key for unrestricted potential use. Assume, as Figure 2.16 indicates, that the enlarged construction in our hypothetical company displays the President along with other rates.

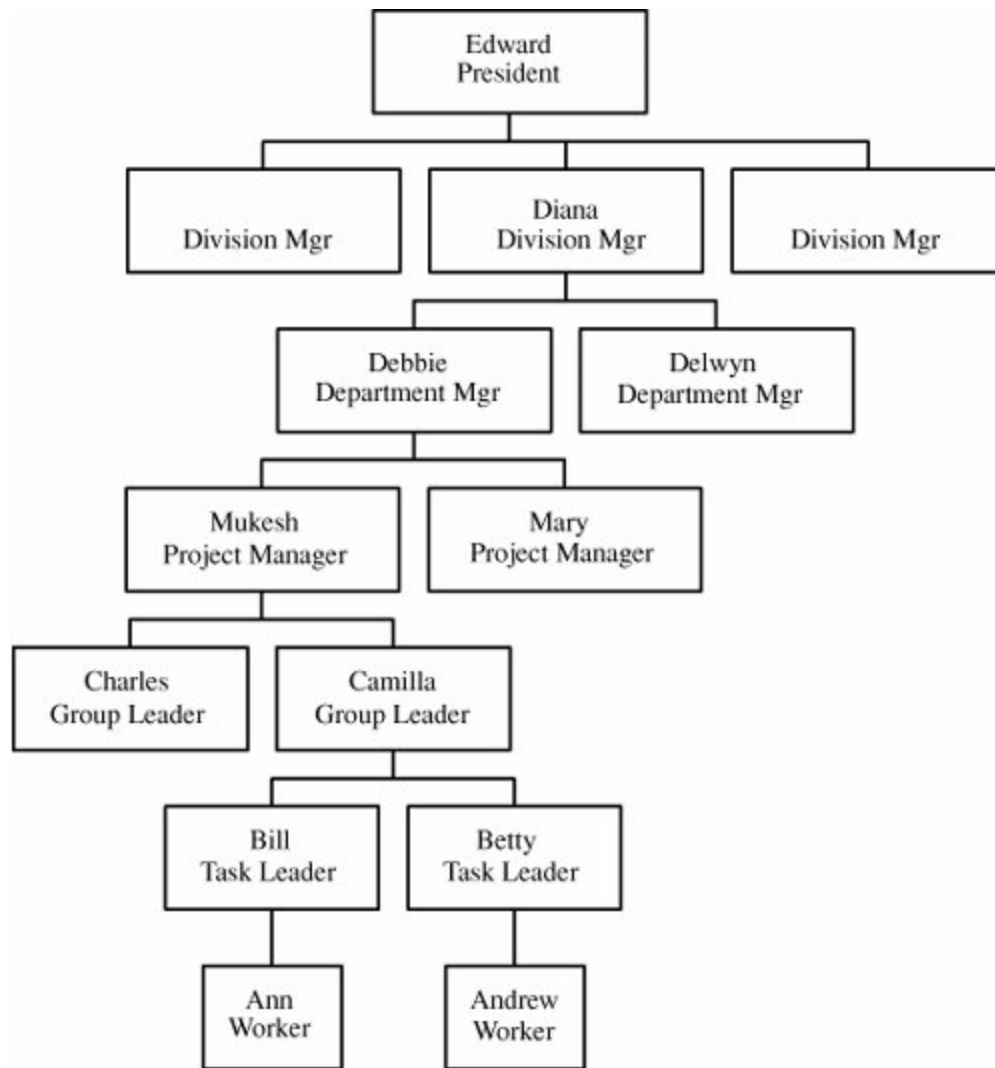


Figure 2.16. Expanded Corporate Structure.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The President writes a letter to every division manager saying "I'm Edward, the Chief Executive of the division, I verify the identity the Diana, a division director I understand from a professional point of view, and I trust in Diana to verify the identities of his subordinates". Andrew receives a packet of characters, every letter linked by name to another, from the President via his Process Group chief. If each employee in the firm receives such a package, two employees who want to share authenticated tips need to compare and contrast only the packet of the other; both packages should have Edward at the very least, or some other high professional, and they will sooner or later deviate. For example, Andrew and Ann were able to analyze their chains, determine they were the same using Camilla, and monitor the underlying components.

Andrew recognizes that Alice's chain is certainly traditional via Camilla, because it can not be separated from his string, and Ann has learnt exactly the same thing. Everyone knows that each other chain is correct because it uses brands and signatures unbroken.

Certificates to Authenticate an Identity

Electronically, this protocol can be more easily expressed than written. With paper it is important to guard against fabrications, avoid changing the section of a string and ensure that the public key element in the bottom is the string. With electronic digital signatures and hash features the whole lot can be carried out electronically. The concept of the use of a digital certificate with a list of authenticators is generated by Kohnfelder which is expanded in Merkle 's articles.

A public important identity and user identification are bound in one certificate and signed by someone referred to as a certificate expert, who attests to its binding accuracy. The company may build a certificate design in the following manner within our example. Edward selects a pair of public key features, publishes the general public part where anyone can understand it and retains the private side. -- section manager, like Diana, leads to her public leader pair, puts both her individuality and the public key into a file, and sends the message firmly to Edward. Edward symptoms it by creating a hash value and encrypting the information with his private key item. Edward affirms by putting his signature on the concept that the general public key (Diana's) and the identity (like Diana's) are exactly the same person. The document is called a Certificate of Diana.

The majority of administrators in Diana's department build messages that use their suggestions, Diana signals and hashes through and everybody's results. She also adds to each of them a copy of the Edward badge. But anyone can confirm the license of a manager by starting with Edward's common primary population, deciphering the license of Diana for public control (and personality) and using the primary public of Diana to decipher the qualifications of the manager. Figure 2.17 illustrates how Diana and some of her technical Delwyn certificates are made. The strategy goes down to Ann and Andrew across the hierarchy. As shown in Figure 2.18, the license from Andrew is indeed his own individual certificate and all his certificates are inside the

president's line for all of those above.

To create Diana's certificate:

Diana creates and delivers to Edward:

Name: Diana Position: Division Manager Public key: 17EF83CA ...

Edward adds:

Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4
---	---------------------

Edward signs with his private key:

Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4
---	---------------------

Which is Diana's certificate.

To create Delwyn's certificate:

Delwyn creates and delivers to Diana:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...
--

Diana adds:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA
--	---------------------

Diana signs with her private key:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA
--	---------------------

And appends her certificate:







Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA
Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4

Which is Delwyn's certificate.

Figure 2.17. Signed Certificates.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Key to encryptions

-  Encrypted under Betty's private key
-  Encrypted under Camilla's private key
-  Encrypted under Mukesh's private key
-  Encrypted under Delwyn's private key
-  Encrypted under Diana's private key
-  Encrypted under Edward's private key

Name: Andrew Position: Worker Public key: 7013F82A ...	hash value 60206
Name: Betty Position: Task Leader Public key: 2468ACE0 ...	hash value 00002
Name: Camilla Position: Group Leader Public key: 44082CCA ...	hash value 12346
Name: Mukesh Position: Project Manager Public key: 47F0F008 ...	hash value 16802
Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA
Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4

Figure 2.18. Chain of Certificates.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Trust With out a Single Hierarchy

Certificates based on management structure were provided in our illustrations. Nonetheless, it is not essential to have or even follow this type of structure to use a certificate for authentication. Whoever considers that an authority is acceptable, can indicate a certificate. You do not actually contact the president or chancellor if you want to decide if a person is accredited by a university or college, but rather go straight to the documentation office or registrar. You may ask the workplace or the film director about recruitment to test someone's position. And in order to check if someone resides at a certain address, you can consult any public record office.

Often a person is selected to test if the document or a specific person is authentic. For example, a notary frequently shows that the (written) signature of your document is valid. Some companies have a stability officer to ensure that an employee has sufficient protections to read a report or attend a meeting. In most businesses, each website or flower location includes a separate employee office; for this position the staff agent guarantees the employees on this Web site. These officers or office buildings may credibly claim to the people inside their competence certificates. Organic hierarchies in modern society are present, and can be used to confirm certificates of the same hierarchies.

The only problem with a hierarchy can be the highest level of the need for trust. The entire authentication string is certainly secure because every document contains the main item that, apart from the top, decrypts another certificate. Within a company, the person at the very top can really be reasonably believed. However, if certificates in electronics are respected, citizens will be able to securely alternate certificates around companies , organizations and countries.

The Internet actually is an vast group of intercompany, intergovernmental and overseas (besides intra-company, intra-organized and intranational) networks. It is not a part of any government, nor is it a private company. It is actually governed by the Web Society board. THE WEB Society has electricity simply because its citizens , public authorities and companies which make up the web decide to connect. However, there's no "core" online. Various firms, such as C&W HKT, SecureNet and Certiposte will be root-certification bodies of governments, meaning each of them really is the highest professional who has certificates for symptoms. These organizations are also responsible for root-certification of governmental authorities. There are several sources, mostly organized around national limits, rather than just a root and the top.

In this section we have addressed many approaches to key element distribution ranging from direct modification to distribution through a central distribution service to accredited progressive distribution. It addresses the main Community infrastructures, offers a more comprehensive overview of the notions of qualifications and qualification bodies of government. However, no matter what path to mainstreaming is taken, each has its advantages and disadvantages. Please notice that the following are points concerning every relevant delivery protocol:

- What are the organizational constraints? For example, does a permanently accessible facility such as the main distribution middle need the standard protocol?
- What are the confidence requirements? Who should be able to do it correctly and what entities?
- What is the faulty safeguard? Can an external personify the entities and weaken security within the standard protocol? Can anyone join in without diagnosis the normal protocol fraud?
- How protocol can be efficient? A standard protocol that requires several steps to create a primary encryption that is commonly used is very important; a very different one is that many long-term measures are taken for the one-time use.
- How simple would the protocol be to implement? Please notice that sophistication might not be the same as manual usage in machine execution..

2.9 Review Question

- 1.What would definitely create unbreakable encryption? What are the characteristics of an encryption that would not break down?
2. Is substitution a permutation on the plaintext icons actually necessary? Why or you're going to want to?
3. Explain why a high level of security could be obtained by the product of two fairly simple ciphers, such as substitution and transposition.

4. How do you check ciphertexts quickly to say whether they are the product of a simple substitution?
5. How can you check a bit of ciphertext quickly to tell you if the transposition was probably the result?
6. suggest a way to get a very long, inconsistent sequence of numbers. The sender and user must be able to easily access your resource, but it is not instantly obvious to the outside party and not transmitted from sender to receiver.
7. If the speed of the standard personal computer (for residence or light workplace use) is constant, estimate the amount of time required to break a DES encryption by checking all 256 feasible keys. To Create a similar estimate to get a 128-bit AES key.

8. Record three applications when a stream cipher will be desirable. Are programs for stop ciphers more frequent? Why or you will want to? Why do you consider this is legitimate?

9. Are DES and AES stream or block ciphers?

1. Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger - RAND Corporation Publisher: Prentice Hall

2. Cryptography and Network Security - Principles and Practice fifth edition Stallings William Publisher: Pearson

3. Cryptography And Network Security 3rd Edition behrouz a forouzan and debdeep mukhopadhyay 3/E Publisher: McGraw Hill Education

4. Cryptography and Network Security, 3e Atul Kahate Publisher: McGraw Hill

Chapter 3. Program Security

3.1. Secure Programs

3.2. Nonmalicious Program Errors

3.3. Viruses and Other Malicious Code

3.4. Targeted Malicious Code

3.5. Controls Against Program Threats

3.6 Review Question

3.7 References

100%

3.1. Secure Programs

We all know that security implies a sense of trust that this program ensures anticipated confidentiality, integrity and availability. Consider what we are indicating when we speak of an application being "secure." As, how can we see and decide the security of a software component or code portion? Naturally, this problem is close to the task of assessing the standard of software in general. One reasonable way to assess security or quality is to ask most people to identify the software functions that contribute to its overall protection. We will possibly receive multiple answers from different people, however. This is the difference, because who analyzes the program decides the value of the software. Another individual may decide, for instance, that the code is secure because security checks would take too long to crack. And someone else will decide the code is secure if it worked with zero obvious failure for a period of time. Nonetheless, a individual can decide to make code insecure by any possible failure to comply with security requirements.

The paradigm in 'penetrate and patch' was used in early work in computer security in which analysts looked for and corrected errors. Often the "tiger staff" of high quality is assembled to test the reliability of a device by actually failing. The control was considered to be a "evidence" of safety; similarly, the device resisted attacks, it was sound and stable in consideration, sadly, evidence has often developed into a counterexample, which has exposed not just one but a number of essential security issues. The issue was, on the other hand, that the computer was rapidly "patched" to fix or restore security. The patch attempts, on the other hand, were essentially futile, making the device less stable and sound than reliable, as fresh failures are often implemented.

There are approximately four explanations.

1. A small focus on the fault itself and not on its context was urged by the pressures to fix a certain problem. The analysts have especially found the immediate cause and not the underlying style or requirements.
2. Often in addition to the immediate section of the fault the fault experienced obvious unwanted consequences.
3. Settling a problem sometimes triggered a mistake in another place, or the fix fixed it in just one place, not in other places.

4. The fault could hardly be corrected appropriately because of this machine functionality or overall performance.

The insufficiency of penetrate and patching has led experts to look for an easier way to trust that code meets their safety requirements. This is a good way to equate the demands with the behaviour. This could allow us to look at programs and decide if they operate according to their expectations or users to be aware of application protection. It may be improper software activity that triggers a system vulnerability. We consider such an unexpected activity an application security flaw.

Flaws in program security can easily be triggered by any software fault. This is all from confusion about plans to a one-character mistake in coding and even typing. The defects can be caused by bugs in a code component or by a failure by the distributed interface of various programs or software elements to communicate. The security defects may expose a code designed or coded intentionally to be malicious or even just constructed slopingly or misguided. So it is useful to separate software imperfections into two logical groups: unintended human error and intentionally caused malicious errors.

Types of Flaws

We may identify categories that differentiate one problem from another to help us understand the problems and their prevention or correction. Landwehr et al., for example, provide a taxonomy of program failures that break them into deliberate and unintentional failures at first. Intentional flaws are further divided into malicious and no malicious ones.

In the taxonomy, the inadvertent flaws fall into six categories:

- **Domain errors:** controlled access to data
- **serialization and aliasing:** program flow order
- **insufficient identification and authentication :** basis for authorization
- **boundary condition violation:** fault in the 1st or last case
- **inacceptable validation error** (incomplete or inconsistent) : permission checks
- **other exploitable logic errors**

3.2. NON MALICIOUS PROGRAM ERRORS

As a human being, programmers and different other developers create many mistakes, most of them unintended and unmalicious. Many such errors cause malfunctions of the program but do not normally lead to more severe security vulnerabilities. Nonetheless, several types of errors have for many years been affected by security professionals and programmers and no reason can be thought of as having gone. In this section, we discuss three typical types of errors that have allowed many recent security violations. We explain any type, why security is really essential and in particular how this can be prevented or mitigated more effectively.

Buffer Overflows

A buffer overflow is a measurement equivalent that seeks to spill out two liters of water in one liter of pitcher. And what a mess these mistakes made in coding!

Definition

A buffer is a space where data can be stored. There is a memory buffer. Due to a small memory, the capacity of a buffer is small. For that purpose, the programmer must specify the maximum size of the buffer so that the compiler can set this amount of space apart in many programming languages.

See an example for how buffer overflows can occur. The statement is in the C language program

```
char sample[10];
```

the compiler has 10 bytes in store for this buffer, one byte for each of the ten array components, the sample[0] for sample[9]. We are running the statement right now:

```
sample[10] = 'A';
```

and subscript beyond the limitations, so there's a problem with us that might be, usually not between 0 and 9. In order to define the problem, the compiler gets the nicest end result (from a security perspective), and tags the error during build. However, we are unable to recognize the issue if the declaration were

```
test[i] = 'A';
```

until i was set to an oversized subscript at the time of execution. It would be helpful if the program produces a message warning that your subscription is not limited during implementation. Sadly, buffer sizes in certain dialects need not be predefined so that an out-of-bound error is not detected. In addition, the code had to verify that any subscript takes time and time when it is executed and the assets are caught up in a fairly uncommon problem. In addition, if the compiler has been attentive to the buffer statement and uses it, the same problem is caused by indications that an effective limit is not fairly set. Consequently, some compilers produce no code for the exceeding boundaries evaluation. Why do we not take a closer look at this problem? The potential overflow only causes a major problem in most cases. It is important to note. The problem depends on what happens next to the test array. Suppose, as a image, the following are filled out with Avis A and the incorrect comparison uses B for each of the ten components of the test array:

```
for (i=0; i <=9; i++)
```

```
test[i] = 'A';
```

```
test[10] = 'B'
```

All systems and data components have memory space, space with the operating system , various other code and resident routines during execution. There are almost probably definitely four instances where the 'B' is to be decided. If the extra personality is overflowing into the user's data space, it merely overwrites the existing variable value (or it can be created as yet) and may affect

the result of the program, but it does not affect additional information or programming .

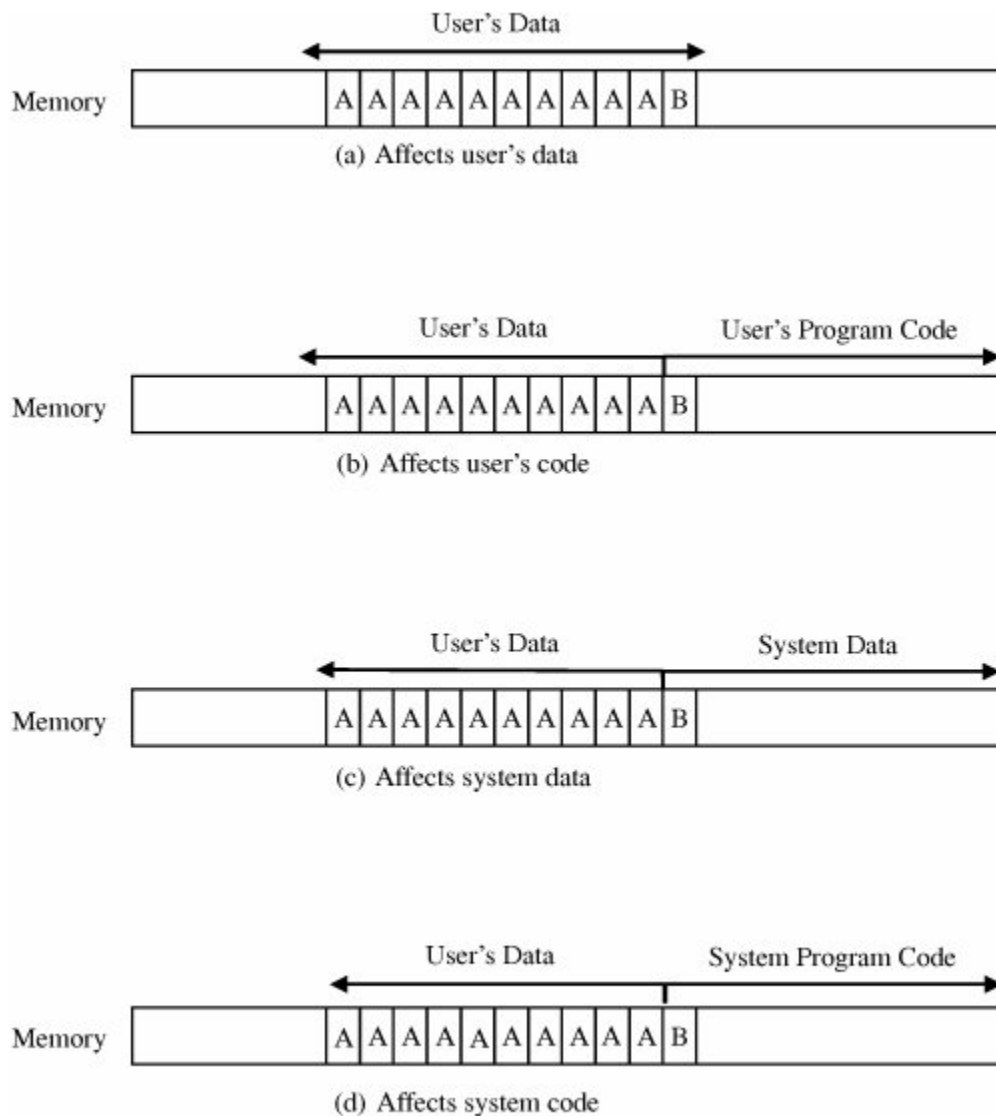


Figure 3.1. Places Where a Buffer Can Overflow.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

3.3 VIRUS AND OTHER MALICIOUS CODE

Programmes are never security risks themselves. The programs only work on data and take action until it results in knowledge and state changes. Most of the work that is done in a program is not seen by users, so that they do not understand any malicious behavior. For starters, when did you find anything the previous time? Do you have any idea of the form a store of a document?

Would you consider it if you know that a text files somewhere on a disk? Can you say if the program does something other than its planned contact with you? When you make a document, which files are changed by a word processor? These questions are not answered by most users. Since computer data are typically not directly accessible by users, however, malicious people may render programs a means of accessing and modifying data and other programs.

Let us look at the possible consequences of malicious code, then analyze in depth different forms of programs for data interception or alteration.

Why Worry About Malicious Code?

In particular the systems, None of us likes the unexpected. Malicious code acts unanticipatedly for the malicious purpose of the programmer. The malicious code we think lures in our system, all or a few of the programs that people run or perhaps a disgusting part of another program that somehow links with a (good) program..

Malicious Code Can Do Much (Harm)

Malicious code can do something, such as write a note on a computer screen, interrupt a running program, create a sound and delete a stashed text. Or malicious code does nothing at this time; it may lie inactive before any occurrence leads to code. or malicious code does nothing at this time. The trigger can include a time or particular date, a timeframe (e.g., 30 minutes later), a meeting (e.g. if a program was running), a condition (to show, when conversation takes place over a modem), a count (one example is, anything takes place fifth), a combination of such types or a random situation. In reality, malicious code every time performs different things or typically nothing runs under the authority of the user with malicious code. Malicious code can therefore influence anything the user can experience. Users regularly track their program code and documents and are capable of reading, writing, modifying, adding and deleting them. And they have to. But malicious code does the same, with no permission or even awareness from the user.

Malicious Code Has Been around a Long Time

The famous literature and the press seem to underline the consequences of malicious code as if the trend is quite new. That's not the case. The discovery of viruses often credited Cohen to a phenomenon that has been recognized long before, but Cohen actually gave a name to it.

In 1984 ,Thompson, for example, defined code that can be transferred by a compiler during his Turing Award lecture, "Reflections on Trusting Trust." In this speech he refers to the Multic security report, an earlier study by the Air Force. Currently,

At least 1970 have been cited for virus behaviour. Ware's 1970 research (which was published publicly in 1979 and Anderson's 1970 U.S. Air Flow Force planning research (which also Schell refers to) still describes risks, flaws and system security defaults, especially those of a intent. The news about malicious code is the number of unique instances and copies have been appeared.

There is still malicious code in around and its consequences are more common. It is crucial that we know how and what it looks like, to prevent harm to it or, at least, to mediate its effects. How can a system control malicious code? How can a system accommodate it? How is malicious code transmitted? How is it recognizable? How can it be found? How does it stop? How can this be avoided? In the following section, we will address these problems.

Kinds of Malicious Code

Malicious code or a malicious program is the common term for unintended or unwanted effects of a significant harm-caused agent in programs or in code parts. This description prevents errors, but can have a significant negative impact as well. This description also ruled out the fact that two positive systems sometimes combine to create a negative conflict. In general malicious code will work with all the predictability of a two-year - old boy: we usually know what two-year - olds would do. We might even know what a particular two-year - old sometimes does amazing capacity in other unexpected situations.

The operator is the program creator or the person responsible for its distribution. By this concept, the majority of software tests, reviews and tests defects are not regarded as malicious code because we regard them as unexpected. However, when you look at this section, accidental problems will certainly lead to irreparable response to intentional malignancy; in all situations, a reasonable reason will lead to a catastrophe.

You were possibly affected at any point by a virus either because your computer was contaminated by one or because you were unable to get access to an infected machine.

A virus is a program that can pass the malicious code by modification to various non-malicious programs.

The word virus is written because the infected software, which includes many safe files by adding themselves to the software, by decimating them or living with them in combination with them, is like a biological virus. We can not believe that yesterday a clean system is still good as infections are tricky. Furthermore, it is possible to change a good program to incorporate a duplicate file of the infection program, so that the good, tainted program itself begins to infect the other programmes. The infection usually spreads geometrically, overwhelms a complete computer system and spreads to all the other computer systems associated with that infection.

Either a virus may be temporary or resident. A transient virus has a true life that is dependent on your host; the infection happens as your associated program operates and finishes when your attached program ends. A resident virus has been detected in memory; at that time it can be stable or be introduced as an independent system even though its unified system is terminated. (In the meantime, the transient virus will spread its infection to a diverse program.)

A Trojan horse is a malignant code with a second, obvious malignant effect, e.g. a Trojan horse, despite its main effect,

A logical bomb is a form of destructive code "explores" or disappears if a predefined condition occurs. A time bomb is a logical bomb, the trigger of which is a date or a period.

A trapdoor or back door is a component in a system from which someone else can get to the program with unique benefits rather than a self apparent direct call. An automated bank employee software can allow for the process, for example, of logging all transactions on that computer, any person entering 990099 on the keyboard. For this model, the trapdoor could be intentional for help purposes, or any reports of wrongdoing could be a wrong path for the implementer to crash.

A worm is a program that spreads duplicates of itself through a system. The key difference between a worm and virus lies in the fact that a worm spreads via systems and that a virus can be spread through any medium. In addition, the worm propagates duplicates of itself as an independent program, and the virus spreads duplicates as a program which joins or installs in various programs.

White et al. also describe the rabbit as an unbounded virus or worm in order to dilute certain computer resources. In a rabbit, for example, duplicates can be created and stored on the hard disk with the intent to fill the hard disk fully.

The term 'virus' is sometimes used to refer to any malicious code. Furthermore, a third type of issue can be combined with at least two types of destructive code. For example, if the virus spreading code cause a chance after a time frame has passed, a virus may become a period bomb.

The sorts of malevolent code are condensed in Table 3-1

TABLE 3-1 Types of Malicious Code

Code	Characteristics
Virus	Attaches itself to program and propagates copies of itself to other programs
Trojan horse	Contains unexpected, additional functionality
Logic bomb	Triggers action when the condition occurs
Time bomb	Triggers action when the specified time occurs
Trapdoor	Allows unauthorized access to functionality
Worm	Propagates copies of itself through a network
Rabbit	Replicates itself without limit to exhaust resources

Since "virus" is the common name for any and all kinds of malicious code, and since thin lines exist between entirely different malicious code types, in the following discussion we will not be too restrictive. However, we wish to display malicious code spread, but it is allowed, and what will result. A pestilence may be a convenient term for mobile malicious code, so we prefer almost exclusively to use the word 'virus' in the following sections. The generated points also refer to various malicious code forms.

How Viruses Attach

Nothing can do or harm someone by a written duplicate of a virus. There isn't anything even executable virus code on the disk. What triggers a virus to replicate? In order to execute and propagate the destructive function of a virus, it must be triggered by itself. Luckily for virus writers, but sadly there are also ways to ensure programs are operating on a working machine for us.

Remember, for example, the SETUP program you are using on your computer. It can call dozens or hundreds of other programs, some on the internet, some on computers and others on memory. In case any of the programs has a virus, it is possible to release the virus code. Let us see how. Let us see how. Assume that the virus code in the distribution media is a program, such as a CD; when executed, the virus could install itself on a permanent storage medium (typically, a hard disk). Human interaction is required to get the cycle started; a person places the virus in the distribution medium; maybe another person begins operating the program that the virus is certainly attached to. (The execution can take place without human intervention, but it is certainly triggered by the use of a date, or a certain amount of time after the execution is done.) After that there is no human interference; the virus can spread on its own.

An attachment to an email address is a more common form of triggering the virus. The writer tries to convince the recipient (the receiver of the email message) to open the attachment during this attack. The disabled virus will do its job when the virus attachment is definitely opened. Modern e-mail managers automatically open attachments when a recipient opens the e-mail size to "help" the recipient (victim). The virus can be executable, but other file types are equally harmful, embedded in executable attachments. Objects as graphics or picture files, for example, may contain code that an user needs to execute, so that they can be virus transmitting agents. It is usually easier to compel users to open documents on their own rather than automatically; programming does not have the consent of a user if it is necessary to execute measures which are essential to health. Simple to use often overrides protection, however. So programs like browsers, email dealers and spectators sometimes open files "helpfully" without asking the user first.

Appended Viruses

A computer virus binds itself to the program; the virus is activated while this program is running. The system is normally straightforward for this kind of association.

In the simplest case, before the first execution instruction a virus incorporates a copy of itself into the executable program. All virus instructions take place first. Use the last virus instructions to test what is typically the first program instruction. Instead, control flows normally. In Figure 3.4, this scenario is proven.

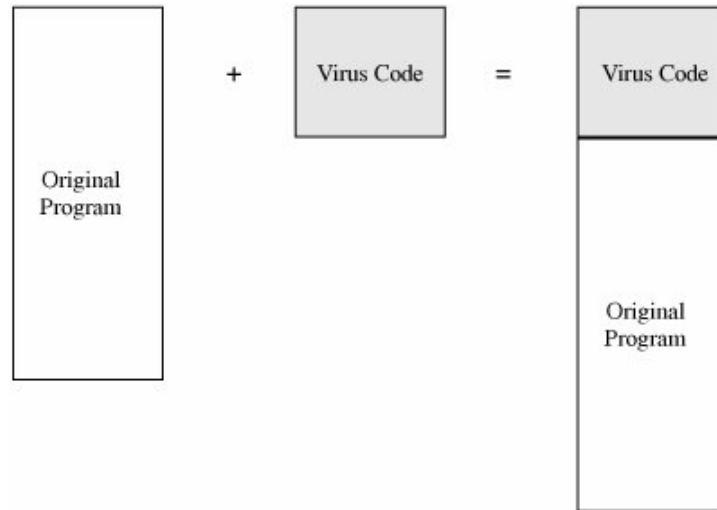


Figure 3.4. Virus Appended to a Program.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

These are basic and usually successful forms of attachments. The writer of the virus code doesn't know much about the virus, and often the attached software behaves as a carrier of the virus. After switching to the original program, the virus works. Usually, if the original program still does what it used to, an person is not aware of the virus. That is how the bulk of viruses is bind.

A virus that runs the first program can be a substitute to the attachment, so it can be controlled before and after it has been executed. For example, a virus writer may want to avoid detection. The presence of the virus on the disk is shown by its file name or the amount of space on the disk is affected by its sizes. The writer of the virus will arrange to add the virus to the program which compiles the files list on the disk. If the virus reverts after the listing program produces the list, but before the shown or printed list, the virus may delete the entry from the list and counterfeit the number of spaces to avoid it. Figure 3.5 indicates an surrounding virus.

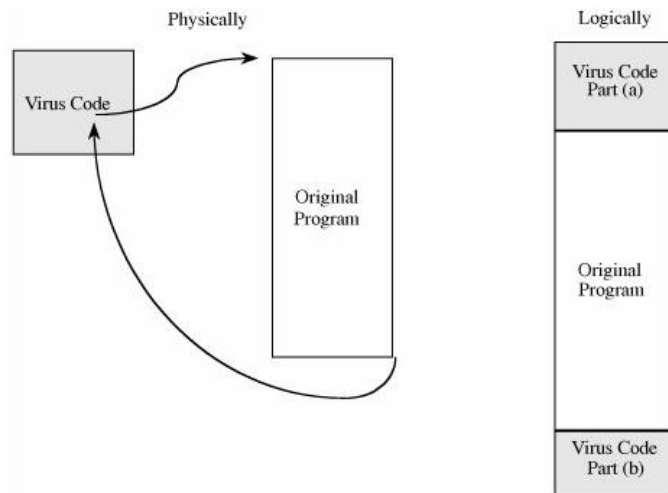


Figure 3-5. Virus Surrounding a Program.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

A third event is when a part of the target is substituted by the virus, inserted into the target's original code. This has been shown in Figure 3.6. The infection author must obviously be aware of the exact layout of the first program to figure out what bits of code a virus might contain.

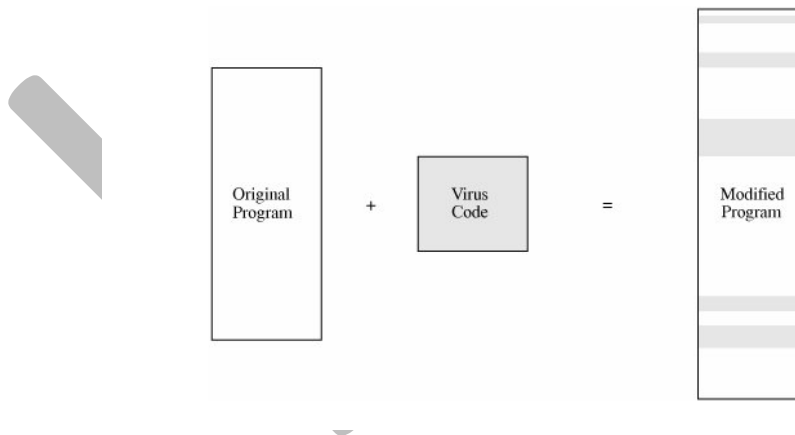


Figure 3.6. Virus Integrated into a Program.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Finally, the virus will replace the entire target, representing either the effect of the target or disregarding the usual effect of the target and getting only the effect of the virus. The customer is to see the failure of the original program for this case.

Document Viruses

Currently, a most common type of virus is a Document virus that executes inside a completely structured document, e.g. a written text, a database, a presentation slide, an picture or a tablet.

The documents contain the two elements of data (words or numbers) and control (for example formulations, formatting controls, links) which are exceptionally structured. The commands are a rich piece of programming, including macro, variables and methods, access to directories, and even calls for systems. Any highlights of the programming language are used by the text virus author for malignant activities.

The standard client typically considers only the content of the document (the data or material), so the virus writer mainly includes the virus in the system control section, as in the integrated virus program..

How Viruses Gain Control

Instead of Target (T), virus (V) should be renamed. The virus must be either T, or "I am T," or T must be taken away from the path of the virus. It will succeed in saying "Call me instead of T." The increasingly barefaced virus will effectively say "conjure me [you fool]." In reality, the virus must either be T.

The virus will agree that T's name is most suitable for the traditional system by supplanting (or adding) T's code into a document structure. The virus will overwrite T (in general, the T duplicate is removed, for example). On the other hand, the virus will adjust the pointers in the log table so that the virus is located in the T file system, instead of T. Both have been shown in Figure 3.7.

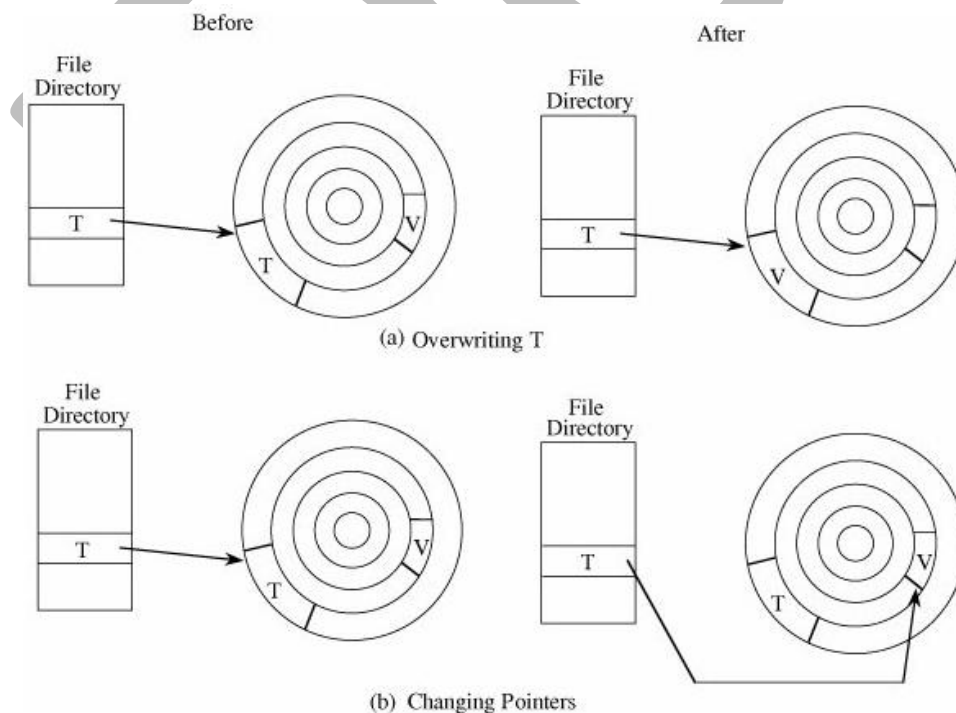


Figure 3.7. Virus Completely Replacing a Program.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The virus will override T by modifying the sequence T invoked to now name the virus V; this name can be used to remove portions of the resident operating system, for example by shifting pointers to certain inhabitant parts of the operating system files for different forms of breaks.

Homes for Viruses

- The author of the virus will detect these features of a virus:
- It's hard to recognize.
- It is not removed or disabled effectively.
- Contamination is widely spread.
- You can reinfect your home program or the other one.
- It's hard to do anything else.
- It's a free machine and an independent operating system.

Perhaps a few viruses satisfy each of these requirements. When choosing what the virus will do and where it will live, the virus writer looks at these goals.

Just a few years ago, the virus writer had been checked to write code which would be executed more than once to replicate the virus. At the moment, one execution is enough to guarantee the distribution around the files. Many viruses are spread by email and both courses are used. In the main scenario, a writer sends a new email to all the address book locations of the victim. These new messages contain a duplicate of the virus that is usually created by this virus. The message is always a succinct, accurate, unspecific message that would allow a new recipient (the principal recipient) to open a link with a partner. In a second case, the virus writer will leave the tamed document for the unfortunate to unintentionally delete the file in advance. For example, it might read "I think that you can appreciate that picture from our vacation." If the effect of the virus is not immediately apparent, the victim may unintentionally move the spread record to numerous unfortunate victims.

Give us an opportunity to take a closer look at the viral residence problem.

One-Time Execution

Most of the viruses actually only perform once, spreading and damaging their infection. As an email connection to a document virus, a virus periodically occurs. It is only done by opening the file at one time.

Boot Sector Viruses

The alleged boot sector virus is a rare instance of a virus attachment, but once truly popular. When a Computer is started, control is started by the firmware to decide which components of the computer hardware are accessible, check and exchange controls with an operating system. A certain hardware platform can run a wide variety of the operating system so, but after the equipment test, the operating system isn't coded in the firmware.

The OS code is stored on the hard disk. This replicate is known as the load bootstrap (often bootstraps), provided that the machine metaphorically runs through its memory through the bootstraps. Code replicates an operating system from a hard drive to a memory and shares power over it. The firmware transfers its power by reading a fixed region of bytes from the hard drive (called the boot sector) to a fixed memory location and hoping to enter that address (which eventually contains the first bootstrap loader instructions). The bootstrap loader must read the rest of the operating system from the hard disk into memory at that point. The client only embeds a hard disk with the current operating system and a bootstrap loader in order to run an alternative operating system. The loader gets and runs another operating system at the point where the customer restarts this new hard drive. For PCs, workstations and large central computers this similar program is used.

Hardware designers save plenty of room for the bootstrap loader to take into account adjustments, development and vulnerabilities. However, as the loader is larger, the equipment generation block "chaining" in which each block in the bootstrap is connected (includes the disk area) to the next blocks, the boot sector on a PC is much less than 512 bytes. This chain makes massive bootstraps, but also rationalizes the existence of a virus. The virus writer just splits the chain at any time, inserts a pointer into the execution code and re-connects the chain following the infection. Figure 3.8 indicates this situation.

Understanding the Resident virus

For anyone affiliated with the network, viruses are a colossal threat. Such dangerous programs often show themselves and take place without the knowledge of the unfortunate victim. The effect of a virus usually goes from stopping your Computer from showing up to fully removing most of your significant records. In general, it will spread to different machines in which you communicate, allowing it to influence the entire network. Given the drastic outcome, a virus is not something on your Computer you need

What is a memory Resident virus?

A memory Resident virus is one of the best known kinds of PC infection. It works by injecting malicious code into your PC 's memory, tainting your new program, and any others. To do this, the resident virus has to develop a method for memory-giving, which means that it has to find something to lock away. It must also create a procedure to start tainting various records by initiating an inhabitant code.

A resident virus may use different methods to spread the disease. One of the most ignored strategies includes the capacity-implementing TSR (Terminate-Stay-Resident). Although this technique is the best to call for disease, an infection scanner can also effectively characterize it. The management of MBCs (memory control blocks) is an increasingly desired system. Ultimately, a virus must be applied to clear obstacles so the occupant's code is dispatched. When a virus is altered for every program to be executed, for example, it must be snapped into interfering with the stacking and executing capacities allocated.

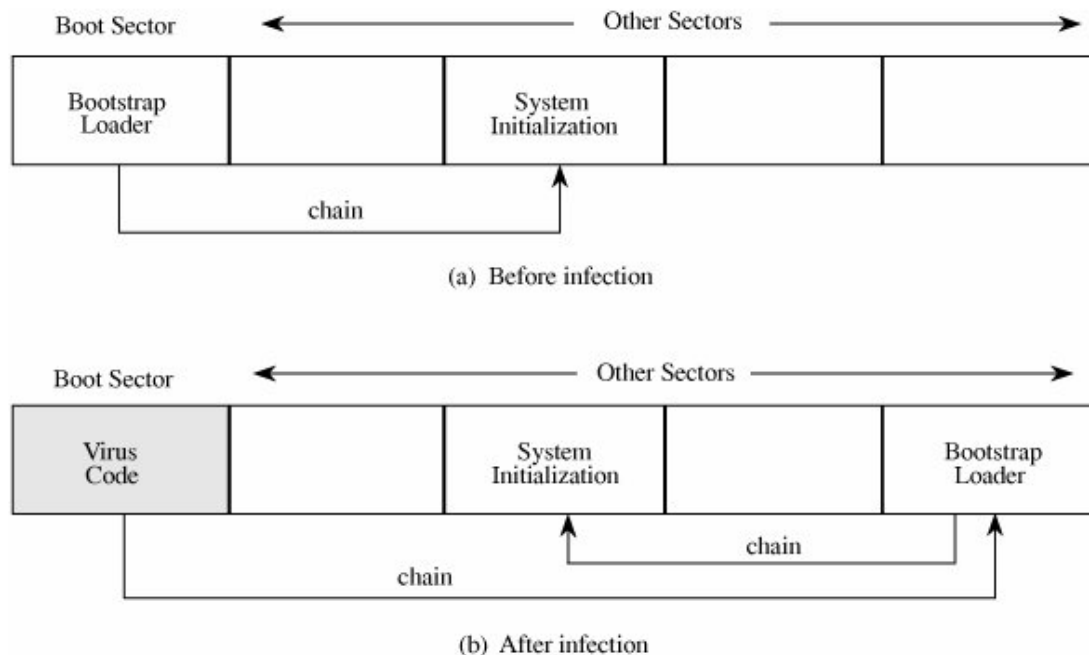


Figure 3-8. Boot Sector Virus Relocating Code.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Other Homes for Viruses

A virus that does not switch to a secure base will fight for itself more. That doesn't mean, in any case, that the virus will go out of business.

An application program is a renowned home for a virus. Numerous applications, e.g. word processors and spreadsheets, provide a "macro" which enables a client to record command advancements by inviting them to do so. Additionally, the program also provides a "start-up macro" which is run on a single application each time the application is executed. A virus writer can make a virus large scale that adds itself to the startup orders for the application. It likewise then implants a duplicate of itself in documents with the goal that the infection spreads to anybody getting at least one of those records.

Libraries often are excellent places for enforcing the rule. Since libraries are used in many initiatives, their coding has a widespread impact. Additionally, libraries are often exchanged between customers and transmitted from one customer to another, a training that spreads the contaminant. At long last, executing code in a library can pass on the viral infection to other transmission media. Compilers, loaders, linkers, runtime screens, runtime debuggers, and even virus control projects are a great possibility for facilitating infections since they are generally shared.

Virus Signatures

Owing to the existence of an similar virus signature with a specific virus, in some cases the anti-virus software can recognise an unknown virus. New viruses have a virus signature that is not used by another virus. In certain cases, the antivirus virus can recognize a virus that is unknown, however new "strains" of realized virus some of the time utilize a similar virus signature as prior strains.

[illegible]

Most viruses connect to programs, such as disk, that are placed away on media. The virus part connected is invariant, so that a signature can be seen from the start of the virus code. In relation to its attached text, the linked element is consistently put in a similar role. For example, 400 bytes from the top or bottom of the infected file can be dependable towards the start of the virus. The virus will probably start recording as the writer of the virus needs to gain control of its output before the true code of the infected software is managed. In the simplest case, the virus code is at the top of the program, and the entire virus is in control until the typical code is infected. In various situations, the infection requires a few guidelines that point or hop in certain, slowly loosely gritty directions. For example, a condition check and a hop or call to another infection module may be part of the infection code. In any case, there is also a fascinating example of the code under which power is shared. In Figure 3.9 these two situations appeared.

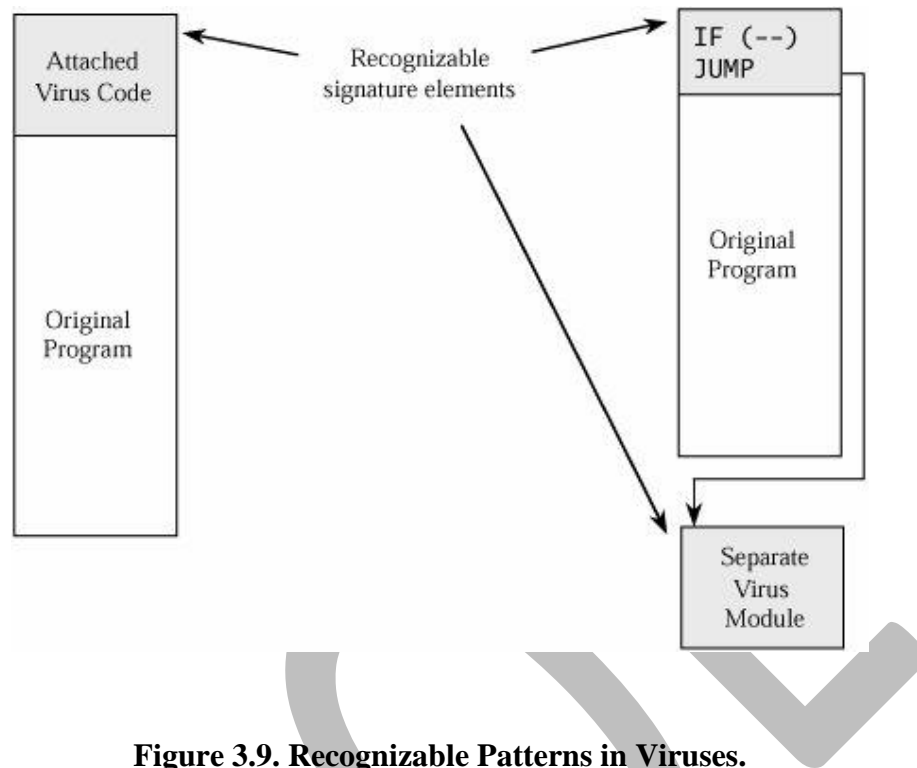


Figure 3.9. Recognizable Patterns in Viruses.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

A virus may enter a document in order to increase the size of the files. Or on the other hand, if the program's size does not change and if the program works, the Virus will destruct all or part of the basic program. Each of these distinctive effects is to be chosen by the virus editor.

You may use the virus scanner to detect changes to a document by using a code or a checksum. It can also try suspicious instances, such as a JUMP instruction as a device initial directive (if the virus has yet to be registered on the basis of a log, as seen in figure 3.9).

Execution Patterns

A virus copywriter may want to make several things simultaneous by a virus, such as spreading the virus, avoiding detection and causing harm. These types of goals are displayed in Table 3.2 in order to easily answer each objective. Unfortunately, many of these behaviors are usually quite normal and may be undetected otherwise. For instance, one purpose is to change the types of files. Most ordinary programs generate files, remove files and even write to mass media storage. There are also no primary signs that a virus exists.

Table 3-2. Virus Effects and Causes.

Virus Effect	How It Is Caused
Attach to an executable program	<ul style="list-style-type: none"> • Modify file directory • Write to the executable program file

Attach to data or control file	<ul style="list-style-type: none"> • Modify directory • Rewrite data • Append to data • Append data to self
Remain in memory	<ul style="list-style-type: none"> • Intercept interrupt by modifying interrupt handler address table • Load self in the nontransient memory area
Infect disks	<ul style="list-style-type: none"> • Intercept interrupt • Intercept operating system call (to format disk, for example) • Modify system file • Modify the ordinary executable program
Conceal self	<ul style="list-style-type: none"> • Intercept system calls that would reveal the self and falsify the result • Classify self as "hidden" file
Spread infection	<ul style="list-style-type: none"> • Infect boot sector • Infect systems program • Infect ordinary program • Infect data ordinary program reads to control its execution
Prevent deactivation	<ul style="list-style-type: none"> • Activate before deactivating program and block deactivation • Store copy to reinfect after deactivation

Transmission Patterns

A virus will only function well if it can be spread from one location to another. As you have already seen it, viruses are able to migrate in the boot cycle simply by attached or traveling in data to an executable file. During the running of the contaminated system, the journey takes place itself. Provided that a virus can carry out any instructions that a course can provide, it is not limited to any particular method or pattern of execution. For example, a virus can enter a disk or even a network, go to the start sector of a hard drive during its execution, reboot the host machine on the next occasion and remain in the memory of infecting other drives as they hit.

Polymorphic Viruses

Virus signature will typically be the most assured means of obtaining a virus for a form of virus scanner. If the virus always starts with string 47F0F00E08 (in hexadecimal) and includes line

00113FFF on term 12, it's doubtful that the exact properties of certain programs or data files will be included. The chance of a right match increases for longer signatures.

When the disease scanner always looks like these strings, next door the clever viral scanner will trigger something else to do that. Most instructions simply have no impact, like 0 per quantity, comparing a lot with each other or jumping to the next instruction. Such suggestions may be poured into a code element in order to render a specification, often called no-ops. In order to explain this virus, two different, but equivalent, terms could be found, and after installation the virus can select one for its first word with only a few clicks. In order to look at these types, then a computer virus scanner must have. A virus that may alter its appearance is called a polymorphic virus. (Poly means "many" and morph means "form.")

As two independently used viruses, a two-form polymorphic virus can easily be handled. Hence herpes article writer planning to avoid the virus diagnosis would either need a wide or even infinite variety so that the number of potential types for a disease scanner can also be identified. It is not enough simply to slip in a random number or maybe string at a fixed point of the virus' executable version, since the virus trademark actually excludes the random element from the sum .. A new polymorphic virus needs to delete all sections without any problem and change all the fixed data randomly. Instead of usually having "HA! INFECTED BY SIMPLY A VIRUS," a new polymorphic virus often has a fixed (and therefore searchable) string to modify the type.

Trivially, assume that there is a kind of virus writer with a hundred bytes and 50 bytes of code. The writer may distribute the original version in 100 bytes of code followed just by all 50 bytes of data to make two different malware instances. A second type could be 99 bytes of code, a jump instruction, 50 bytes of information and the last code byte. Other forms are 98 code bit, 97 and three, etc. move to the second alternative . The virus copywriter can only produce ample diversity by shifting things around to delude simple virus scanners. On the other hand, when the readers became aware of such tricks, they refined their personal definitions.

A simple set of polymorphic viruses uses encryption to distinguish the type of virus saved. They are often referred to as virus encryption. Three separate things are found in this computer virus. The decryption key item, the computer object code of the virus (encrypted) and the decryption object-code of the virus (unencrypted). For viruses of this kind, decryption software or decryption machine must be transparent so that its signature is present.

Not every copy of a polymorphic virus needs a push to stay away from the detection of all other backups. When the virus differs, any backup does not fit another duplicate with a single individual.

The Source of Viruses

In view of the fact that a virus is very small, it can "hidden" its signal within even bigger and more complicated programs. Two hundred lines with a virus could be split into two hundred lines each of a single leap with a code, each of which can be 100 packets concealed easily in the compiler's , in the database manager or database manager.

The identification of viruses can be beneficial by a method to determine the equivalence between two programs. Nevertheless, assumptive computation is particularly disheartening in relation to the question of equivalence. It is undecidable (although this question can be answered for more than two similar programs). The basic question "Are these two programs equal?" Even if the general indecibility problem is overlooked, two modules may produce subtly different outcomes which may or may not be relevant to safety. One can increase the speed, or typically the former can use a kind of brief work area file, while the latter performs almost all its computations. These variations can become benign or a new disease marker. Therefore, a screening program is not likely to evolve which can separate infected and uninfected searches.

Even if the general is bitter, it isn't really that unusual. When we know that a new virus will enter a computer device, we can all search for it and detect it. However, we are still working to disinfect the operating system once we have detected a herpes simplex virus. If you take the virus away from a new operating machine, you will have to be able to identify the instances more easily than they can distribute.

Prevention of Virus Infection

It's not always possible to get executable code from an infected source in order to prevent the spread of a virus. This principle has become clean to follow as it has become easy to say whether or not a file is executable. The .Exe extension, for instance on PCs, was a clear indication that the file could be executed. However, as we have described, brand new files are more complicated, and a seemingly nonexecutable report may also contain some deep-seated executable codes. For example, a word processor might have commands in the record report, which, as we discussed earlier, make it simple for the user to do complicated or repetitive things. However, in the sense of the record they are entirely executable code. Likewise, spreadsheets, presentation slides, other working or business-related documents or even media documents may contain code and/or scripts that can be carried out and thus harbor viruses in a number of ways. And as we can see, the packages running or using these files may also try to help invoke the executable code robotically, whether you need it running or not! Against the specific security principles, email handlers can be set to open attachments or an embedded code to the recipient automatically (without manipulating access), and animated bears can dance around the pinnacle.

A bit-adjusted feature in the design of the Microsoft document is another method used by virus writers. Even though there is a register. The extension to documents is expected to be a Word report. In fact, at the very beginning of the record the genuine type of file is hidden. This convenience obviously allows a person to name a Word document with either a .PPT(Power-Point) or a different extension accidentally. For a few instances, the gadget will attempt to open the file, but if it does not, the system switches to the secret record style program. Thus the virus writer creates an executable file, names it with an unsuitable extension and sends it to the victim, describing it as an image or a required upload-in code or otherwise. The unsuspecting recipient opens the document and executes the malicious code without continuing.

More recently, in files that contain large data units like images or files for studies only, the executable code has been covered. Such viral code bits can not be quickly identified using virus

scanners and definitely not with the human eye anymore. For example, if a photography document is part of a command string, then the virus can be difficult to reach because every sixteenth bit is part of a command string.

As you generally can not grasp the inflamed materials, you can expect that any source beyond the door would be contaminated. Happily, it is not feasible to cut all communication with the outside environment if your code is obtained from a source beyond the windows.

In their insightful paper examining transmission of laptop viruses with human condition,. Examine that efforts to conserve their computers unfastened by viruses lead to communities that are normally unfastened by viruses because network contributors have little (electronic) international contact outside doors. In such cases, transmission is not restricted because of restrained touch but because of the minimal communication outside the doors. Governments are routinely operating isolated neighborhood groups for military or diplomatic information and strategies. The trick appears to be to decide prudently about our culture. However, as the usage of the Internet and the World Web will expand, it is almost impossible to maintain such a separation.

However, there are many techniques for creating a relatively stable digital network, for example:

Using the most robust and well-established industrial software system. There is always a danger that a major producer will carry an outbreak, and we all apprehend a call. Such businesses, however, have a decent reputation, in which even a disastrous accident could critically harm their products in order for them to go to a few diplomas to keep their products virus free and fix any troublemaking code away. Likewise, distribution groups of software programs should pay attention to the items that they manage.

Take a remote Computer to check all new apps. If you need a questionable supply of software, you should first check the software on an unrelated network computer and do not contain sensitive or important facts. Run the app and search for odd behaviors like unexplained show figures. Simple actions. Check the PC earlier than the suspicious malware is run by a clone of an upgraded virus scanner. Only can you place it on a less isolated computer if the system passes these assessments?

After realizing that they are secure, open attachments most useful. Which is "free" is up to you, as you probably already discovered in this chapter. There is definitely uncertain protection with an attachment from an unknown source. You can also mistrust an attachment, but with a strange message from a known source.

Make a retrievable photo and safely shop. This smooth version will allow you to reboot safely when your system documents are overwritten with smooth copies if your gadget is ignited. For that reason, at some point in the rebuild, you need to keep your photograph published. Prepare the photo, before the contamination, now; it is miles too late after contamination. Prepare a copy of the safe boot photograph for protection.

Create and restore copies of computer files that are executable. In this way you will be able to delete inflamed documents and reinstall easy backup copies in the event of a deadly disease contamination, (stored in a safe offline location and direction). Also, back up important data files incorporating infectious code, such as word-processing documents , spreadsheets, slide shows,

snapshots, sound documents and databases, and maintain them. Hold backups on cheaper files, like CDs and DVDs, so that you can preserve long-term antique backups. If you suspect contamination, you want to start with a smooth backup which is taken before the infection.

Use virus detectors often and update them every day (often known as virus scanners). Some of the virus detectors available will each detect and rid themselves of virus infection. Various scanners are higher than one, since the viruses others miss can often stumble upon. Due to the fact that scanners are searching for viral signatures, new viruses are continuously tested. The seller's web site may include new virus signature documents or new scanner variations; also automated updates can be requested. Maintain the signature report of your detector up to date.

Truths and Misconceptions About Viruses

Due to the drastic effects of viruses on the laptop usage network, they are frequently stressed in the press, especially in the corporate section. There is, however, a lot of incorrect knowledge about viruses in the volume. Let us look roughly at some of the famous arguments.

The simplest Microsoft Windows systems can be corrupted by viruses. True. PCs running on Windows are common computers among students and employees of the office. More people can write programs (and viruses) for them than any other processor. The PC is therefore as much as possible the target, when a person decides to write an epidemic. Virus attachment and contamination levels are common to various processors, such as Macintosh operating systems, Unix and Linux workstations and mainframe computers. Cell phones and PDAs are now also candidates for viruses. No writeable storage laptop is actually resistant to a workable virus attack. , this case method means that any device, together with cars, aircraft, microwaves, radios, televisions, casting systems for voting and radiation therapy equipment, can be shot by a plague.

Viruses can change files that are "hidden" or "read most." That's right. We may also try to save files using two mechanisms of running gadgets. First, a secret file may be created to prevent a person or software from seeing a request for files from displaying all the documents on a storage device. Secondly, the study safety is performed exclusively in the file so that the person can not share the contents of the text. All these safeguards are, however, applied through software and virus software that override the security of the local software. The operating system provides primary security, and the safety of software programs is often layered. If you control a safe running machine before a viral polluter is finished, the operating system will save you infection as it avoids the attacks that the virus causes.

Viruses can appear most useful or most successful in application in fact files or only in Word files. True. What are the figures? What is a record that can be executed? There is not always a direct distinction between these two principles as a database can handle the software and even run software. Often a statistical file lists steps to be taken by reading the statistics in the system and this could include the execution of an application. For example, a number of programmes, with statistics exactly such, contain a configuration report. Similarly, record documents processing phrases may also contain startup commands to execute when logging is opened; malicious code may also be included in those startup commands. Although a virus can be activated and discharged more efficiently, a system actually executes reliable documents by applying them. Smart virus

writers were able to manipulate statistics to document programs which motivated many things to do, including transferring copies of the virus to various information files.

On disks or even via e-mail, viruses unfold the simpler. True. File sharing is also done when one user replicates a report on a mobile disk by writing the file to every other client. Any digital record transfer method will, however, work. A document can be put in a network library or on a newsletter board. You can link it to an email address or download it from an online website. Any mechanism for file share may be used to transmit a virulent disease, including package, data, document and others.

After a complete power off / energy reset, viruses can no longer recall. But it is real. If there is a reminiscence of a deadly illness, the infection is mistaken if reminiscence is lost. In other words, the machine memory (RAM) is volatile, thus losing all contents when power is being misplaced. However, viruses will certainly remain in the disk through a reboot cycle. So you may become infected with the disease, you can write the virus down to the disk (or storage network), you may turn off the gadget and switch back on, and you can restart the virus for a restart time. When rebooting a system (whether for a hardware or reboot software), boot zone viruses benefit from manipulation so that the booting area virus can also proceed with reboots as it is triggered immediately upon completion of the reboot process.

Any low hardware (for example the size of a mounted disk) is retained in reminiscence known as "nonvolatile RAM," but these sites can not be accessed without delay via the applications and are written for the duration of hardware initialization only by packages running from read most memory. They are thus immune from viral attack. They are enormous.

Hardware can not be compromised by viruses. That's right. Viruses can only infect items they can regulate; the primary goals are reminiscence, executable files and records. The garage is virus attack situation where hardware carries a writable garage (so-called "Firmware") which can be accessed by manipulating software. Some firmware viruses have occurred. As a fatal disease can control hardware, it seems as if a hardware was swollen by a fatal disease, but it is actually the software system, which is controlling the infected hardware. In any manner an application may even use hardware. Therefore an outbreak, for example, would reason a disk incessantly to loop, moving the inside and the inside again.

Malevolent, neutral, or benevolent viruses may be. That's right. Not everyone's evil viruses. An endemic can, for example, locate uninfected packages, compress them to be considerably less reminiscent and insert a copy of a normal package that disintegrates this program at the start of its execution. At the same time, the virus spreads the compression to various applications. This virus could dramatically reduce the storage needed for packages stored, possibly up to 50%. Compression should be achieved on the request of the virus and not on the request of this device user, or even with his know-how.

First example of malicious code: the brain virus

One of the first viruses is one of the most studied. This is because of changes to the disk label that targets "BRAIN." The so-called Brain-virus got its name. This specific virus which is thought to come from Pakistan targets PCs running an old operating system from Microsoft. Owing to the number of variants, the virus source code has been released to the underground Virus Culture. There are a great many variants.

What does

The brain tries to pass on the infection like all viruses. First, this virus is located in the top memory and then executes a system call to reset the connected upper memory so that it is not disturbed while it is running. By resetting the interruption table to point to interrupt number 19, you are able to obtain an interrupt number 6 (not used) by setting an address at interrupt number 19 at the preceding address. Antivirus screens can also track disk calls. , managing anyone who reads the start sector (returns original start material to one of the wrong areas); other disk calling is made via interrupt 6 to the usual disk read controller.

It appears that the brain virus only transmits the infection, as if it was an experiment or a logical evidence. Virus variants either remove the drives or kill the file assignment table (the table which shows the files on the storage medium).

How it spreads

In the boot sector and six other portions of the disk, the brain virus is located. The original start code is found in one of the six sectors that have shifted from the original boot sector, while the others contain the virus code. There are a duplication of the other three sectors. The virus defines these six sectors as "defaulty" because it is not attempted by the operating system. (To read the bad parts of the operating system by making low-level calls you can push the disk drive). The virus will continue the startup process.

The virus intercepts disk read requests for the attack drive once placed in memory. With every read, the virus reads the disk boot sector and checks the fifth and sixth bytes for 1234 (your signature) hexadecimal value. If the value is detected, the disk will be infected; if not, the drive will be infected as seen in the preceding paragraph.

What was learned

This virus uses some of the usual viral tricks, such as camouflage, capture, and Interception ,detection in the boot Sector. For future attempts, the virus is almost a prototype. Many other authors of viruses seem to have based their work on this basic virus. Consequently, it can be said to be a valuable resource for the community of virus writers.

Unfortunately, aside from a certain amount of fear and confusion, their infection did not bring awareness of viruses to the public. More broadly, with greater impact, followed by viruses such as the Lehigh virus spreading through Lehigh University's computers, the nVIR viruses which came from prototype code on the bulletin boards and the Scores virus that was first identified at NASA in Washington DC. Fortunately most viruses, such as message show or sound, have a limited effect

to date. This is a matter of fate, however, as writers who could combine the simplest viruses obviously had every ability and skill to make viruses more malicious. No general remedy is available for viruses. Virus scanners are successful against today's viruses and general infection trends, but they can not combat tomorrow's version. The only safe protection is total isolation of external exposure that is not possible; in addition, a virus may even be purchased by a trustworthy supplier from software uses;

Instance: The Internet Worm

The Morris Worm, an automated computer program (worm) written by a student of Cornell University Robert Tappan Morris, was published by MIT on 2 November 1988. Morris believes that, while unintentionally triggering denial of service, it gaged approximately ten percent of the 60,000 machines linked to ARPANET in 1988, the worm was intended to calculate the scale of the precursor's "Internet" at the time – ARPANET. The worm spreads by UNIX sending mail, finger, and rsh / rexec exploiting bugs, and devinating weak passwords.

The Morris Worm investigated whether the computer had been infected previously and was running the Morris Worm process until it spread to a new system. The Morris Worm will re-infect it 1 in 7 cases, if a target computer was already infected. This 1-in-7 re-infection procedure stopped a Morris Worm infection from fully preventing the consumer from making a false Morris Worm pretension that the computer was infected. It also caused some users to get infected several times — if too many Morris Worm processes had run on a target machine, computer resources would run out and start malfunctioning.

The United States v. Morris Court case (1991) led to the first conviction of Morris, which included a three-year sentence, 400 hours of community service, and an additional \$10,000 fine, according to the 1986 Computer Fraud and Abuse Act.

What It Do

Morris programmed the Internet worm according to its code in order to achieve three key objectives:

- Determine to which it can spread.
- Disseminate the infection.
- Remainun discovered and undiscoverable

How do worms work?

Computer worms make use of a victim's computer with some of the deepest and most dangerous vulnerabilities. In the operating system of the computer a worm seams are found, which enable it to install and make copies of itself.. To further propagate, known networking and file transfer protocols are monitored.

For cyber criminals who want to use worms to do their dirty work, this might be a double-edged sword. Since worms are exploiting vulnerabilities in the operating system of a computer, successful infections can provide unparalleled access to the internal functions of the compromised machine. But since these vulnerabilities are so large, operating system vendors are often patched very rapidly, so that a written worm can have a fairly short lifespan. However, the sheer number of companies and individuals who do not keep their operators up-to-date usually gives worms a fertile basis for their work.

How It Performed

The worm used several of the Berkeley version 4 operating system known flaws and configuration failures. It had code which seemed to attempt to accomplish three goals.

Where to spread. Determine. The worm had three methods to detect potential victimizing machines. He sought to locate user accounts to meet the goal computer for the first time. At the same time, the worm tried to take a bug into the finger program and in the sendmail mail handler use a trapdoor. In the general community of Unix, all three of these security faults were recognized.

The first security vulnerability was a user-to-system error, in which the worm tried to guess passwords and found one successful. It is encrypted in the Unix password file so anyone can read the cyphertext in the file. The worm encrypted several popular passwords, comparing the ciphertext with the ciphertext of the password file stored. The worms were testing out the name of the account, the name of your owner ("help," "coffee," "coke," "aaa"), and 432 popular passwords. If none were successful, the worm would use an application orthographing file stored on the device. If it got a match the worm would log in to the corresponding account by providing a plaintext password. (The user error is to pick a recognizable password). As a user, the worm could then look for other machines for access by the user.

The second flaw was fingerd, the software that continuously responds to machine user information requests from other computers. The security fault involved overflowing the entry buffer into the return stack. When the fingerd call was over, the fingerd executed directions pushed as a further part of the buffer overflow to link the worm to a remote container.

A trapdoor in the sendmail program was involved in the third errors. This program usually runs behind the scenes waiting for other people's signals who want the system to send mails. When such a signal is received, sendmail receives a destination address that it checks and then begins a dialog to receive the message. Nonetheless, the worm causes the sendmail to accept the command string and to execute the destination address when executed in debugging mode.

Spread infection. Once a suitable target machine had been found, the worm could send a bootstrap loader to the target machine via one of these three methods. The loader included 99 lines of C code to be compiled on the destination machine and executed. The bootstrap loader will then retrieve the remainder of the worm from the host. The communication between the host and the target included an aspect of good computer security or stealth. The worm provided the host with a one-times password when the goal bootstrap requested the remaining worm. Without this password the host will automatically break down the link to the target, presumably to ensure "rogue" bootstraps

(which could be created by a real administrator to try to get a copy of the rest of the worm in order to be studied later).

Stay undiscovered and undiscoverable

The worm was expanded considerably to prevent its discovery on a host once created. For example, when a propagation error occurs during the rest of the worm, the loader zeroes and then removes any previously code already transferred and exits code.

When the worm obtained its complete code, the code was inserted into the memory, encrypted, and original copies removed from the disk. There were also no traces left on the disk and even a memory dump would not readily reveal the code of the worm. The worm changed its name and process identification periodically so that no single name would take a large amount of time to process.

The thing that was Learned

The Internet worm sent a shock wave through the Internet world, then mostly inhabited by academics and researchers. The affected sites closed some of the worm-exploited loopholes and generally tightened security. Some users exchanged passwords. Two researchers, Farmer and Spafford, developed a system administrator software to check some of the same worm flaws. Security researchers testing for website vulnerabilities, however, find that many of the same security bugs still occur today. A new Internet attack wouldn't succeed on the same scale as the Internet worm, but it could still cause several serious inconvenience.

The Internet worm was benevolent in that it only distributed to other networks, but damaged any portion of them. It gathered confidential data, including account passwords, but did not maintain them. While working as a user, the worm may have removed or overwritten files, distributed them elsewhere, or encrypted them, and ransomed them. Perhaps the next worm isn't so innocuous.

The effects of the worm disturbed several people. One positive result from this experience was creating an infrastructure to track and fix malicious and non-malicious code flaws. The Internet worm emerged at around the same time that Cliff Stoll mentioned his difficulties in tracking an electronic intruder (and then seeking someone to deal with the case). The computer group realized it needed planning. Carnegie Mellon University's resulting Computer Emergency Response Team (CERT) was formed; it and similar response centers around the world have done an outstanding job of gathering and disseminating information on malicious code attacks and their countermeasures. System administrators now exchange issues and solutions information. Health comes from educated defense and practice, not indifference and inaction

3.4. Targeted Malicious Code

We've looked at anonymous code written to indiscriminately impact users and computers. Another type of malicious code is written for a specific device, program, and purpose. Some of the virus writers' methods apply, but also some modern ones.

Trapdoors

A trapdoor is an unknown entry point. Developers insert trapdoors during code development, maybe to test the module, provide "hooks" to attach future changes or upgrades, or allow access if the module fails in the future. Besides these legitimate uses, trapdoors can allow a programmer to access a program once it is put in production.

Trapdoor examples

Since computer systems are complex constructs, programmers typically design and test systems in a meticulous, structured, modular manner, taking advantage of how modules or components form the system. Sometimes, programmers first test each specific component of the system independently from the other components, in a process called unit testing, to make sure the component works by itself. Then developers check components together during integration testing to see how they function when they transmit messages and data from each other. Instead of pasting all components together in a "big bang" approach, the testers group logical clusters of a few components, and each cluster is evaluated in a way that allows testers to monitor and understand what may cause a component or interface to fail.

The developer or tester can not use surrounding routines that prepare input or function with output to test a part alone. Instead, write "stubs" and "drivers," simple routines to insert data and extract results from the part being evaluated, is typically sufficient. As testing continues, they are discarded as they are replaced by the actual components whose functions they mimic. For example, Figure 3-10 tests the two modules MODA and MODB with the driver MAIN and the stubs Type, OUTPUT and NEWLINE.

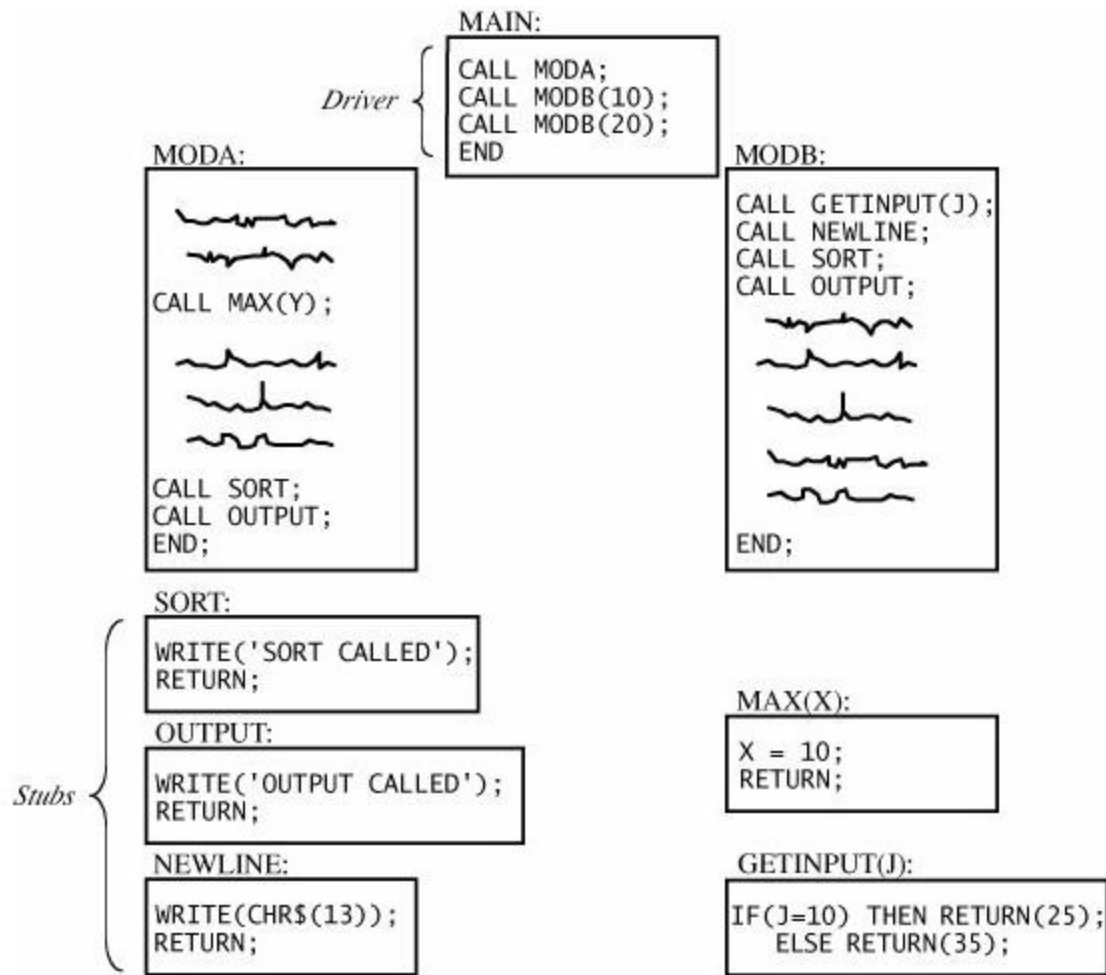


Figure 3.10. Stubs and Drivers.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

During both unit and integration testing, component faults are typically discovered. Sometimes, when the source of a problem is not clear, developers insert debugging code into suspicious modules; the debugging code makes visible what happens as components execute and interact. Therefore, the extra code can force components to show a computation's intermediate results, print each phase number as it is performed, or perform extra computations to test the validity of previous components.

To control stubs or invoke debugging code, the programmer inserts special control sequences into the system design specifically to facilitate testing. For example, a part in a text-formatting system can be programmed to recognize commands like. PAGE, TITLE, and. SKIP. During testing, the programmer could have invoked the debugging code using a command with a set of type var = value parameters. This command allows the programmer to change the values of internal program

variables throughout execution, either to check corrections to this component or to supply values transferred to that component.

Command insertion is a recognized testing practice. If left intact after checking, the additional commands may become a issue. They are undocumented control sequences producing side effects that can be used as trapdoors. In fact, the Internet worm propagated its infection using a debugging trapdoor in an email program.

Poor error-checking is another trapdoor source. A good developer must build a system to verify any data value before use; checking includes ensuring that the data form is right and ensuring that the value is within reasonable limits. However, in some poorly designed systems, inappropriate feedback can not be captured and transmitted for use in unpredictable ways. For instance, a component code can search for one of three expected sequences; finding none of the three would notice an error. Assume the developer uses CASE statement to look for each of the three possibilities. A negligent programmer can allow failure to simply fall through the CASE without flagging as an error. The fingerd bug exploited by the Morris worm happens exactly that way: before returning a pointer to a supposed next character, a C library I / O routine fails to check whether characters remain in the input buffer.

Hardware processor architecture is another typical example of this security flaw. Sometimes, not all possible binary opcode values have corresponding computer instructions. Often unknown opcodes execute odd instructions, either because of intent to check the processor design or negligence by the processor designer. Undefined opcodes are the equivalent to poor software error checking.

Unlike viruses, trapdoors aren't all bad. We will help identify security vulnerabilities. Auditors also request trapdoors to introduce fake, but observable, transactions into the system. And auditors track these transactions through the program. However, trapdoors must be documented, access to them must be strictly controlled and designed and used with full understanding of the potential consequences.

Causes of Trapdoors

Developers typically delete trapdoors during software creation after losing their intended usefulness. Trapdoors in development systems, however, because developers

- Forget to delete it
- Intentionally leaving them in the program for testing
- Intentionally leave them in the finished system maintenance plan, or
- Intentionally leave them as a hidden means of access to the item after being an agreed part of the production system

The first event was an unintentional security blunder, the next two are significant security vulnerabilities, and the fourth is the first step towards an attempt to attack. Know that the fault is not with the trapdoor itself, which can be a useful technique for program checking, correction, and maintenance. Rather, the fault lies with the system development process, which does not ensure that the trapdoor is "closed" when no longer necessary. The trapdoor is a weakness if nobody considers or acts to avoid or monitor its use in vulnerable circumstances.

Trapdoors are a weakness when exposing the system to alteration during execution. These can be manipulated by original creators or used by those finding the trapdoor by mistake or exhaustive trials. A device isn't safe when anyone assumes nobody else can find the hole.

Salami Attack

An attack called a salami attack. This technique gets its name from fusing unusual bits of meat and fat in a sausage or salami. Similarly, a salami attack merges pieces of apparently contradictory data for efficient effects. For example, systems sometimes ignore small sums of money in their computations, as when fractional pennies are measured as interest or tax. These programs may be subject to a salami attack, as small sums are shaved from each calculation and accumulated elsewhere, as in the bank account of the programmer! The cut sum is so low that an particular event is unlikely to be detected, and the calculation can be done to keep the books in balance. Accumulated sums, however, can add up to a tidy sum, funding a programmer's early retirement or new vehicle. Sometimes the resulting spending, not the shaved numbers, gets the authorities' attention.

Types of Salami Attacks

Typical salami attack tale requires interest calculation. Suppose your bank charges 6.5% interest on your deposit. Interest is reported quarterly, but measured monthly. If your bank balance is \$102.87 after the first month, the bank will measure interest as follows. For a month with 31 days, we divide the interest rate by 365 to get the regular rate, then subtract it by 31 to get the monthly interest. Thus, the gross 31-day value is $31/365 * 0.065 * 102.87 = \0.5495726 . Since banks deal only in full cents, rounding down if a residue is less than half a cent is a common procedure, and rounding up if a residue is half or more. However, few people closely test their interest measurement, and fewer will even worry about rounding down the \$0.5495 number to \$0.54 instead of \$0.55. Many currency computing systems understand that, due to rounding, a number of individual computations may be a few cents different from the computation applied to the balance sum.

What are these fractional cents? A programmer who received fractional cents and assigned them to a single account tells the information security folk legend: hers! The interest plan merely had to balance net interest charged to interest owed on the individual accounts' overall balance. Auditors can not note the operation in a particular account. The roundoff error can be significant in a case with multiple accounts, and the programmer's account pocks this roundoff.

Yet salami attacks can be more net and interesting. For example, the programmer may take a few cents from each account instead of shaving fractional cents, again assuming no person has the

desire or understanding to recompute the amount of bank reports. Most people seeing a result a few cents different from the bank's will embrace the bank's figure, attributing the discrepancy to an mathematical mistake, or ignoring the terms in which interest is paid. Or a system can record a \$20 fee for a specific service, while the company norm is \$15. If unchecked, extra \$5 could be credited to a programmer's option account. The sums shaved are not inherently small: one attacker might make withdrawals of \$10,000 or more against accounts with no recent activity; probably the attacker hoped the owners ignored their accounts.

Why Salami Attacks Persist

Computer computations are famously subject to minor rounding and truncation errors, particularly when combining large numbers with small ones. Rather of recording the exact errors, programmers and users should consider a small amount of error as normal and unavoidable. To reconcile records, the programmer requires calculation error correction. Inadequate auditing of these corrections is one reason to ignore the salami attack.

Typically a system's source code is too big or complicated to be audited for salami attacks, unless one is suspected. Size and time are certainly on the programmer's side.

Rootkits and the Sony XCP

A later virus style variant is the rootkit. A rootkit is a piece of malicious code that goes to great lengths not to be detected or, if detected and disabled, restored whenever feasible. The term rootkit refers to the attempt by the application to act as root, a super-privileged Unix system user.

A traditional rootkit interferes with usual user-operating system interaction as follows. If the user executes a command indicating the existence of the rootkit, e.g. by listing files or processes in memory, the rootkit intercepts the request and filters the result returned to the user such that the rootkit does not appear. For example, if a directory contains six files, one of which is the rootkit, the rootkit can pass the directory command to the operating system, intercept the result, remove the listing, and show only the other five files to the user. The rootkit also changes things like total file size to conceal itself. Notice that the rootkit has to intercept this data between the test and the presentation interface (the software that formats the user's results).

Two play the game. Suppose you believe code interferes with your show system. You then write a program that shows data, analyze the disk and file system directly to enumerate files, and compare these two results. A rootkit revealer is such a program.

A computer security expert named Mark Russinovich built a rootkit revealer on one of his devices. Surprised to find a rootkit. He knew the rootkit had been mounted when he loaded and played a music CD on his computer. Felten and Halderman investigated this rootkit, called XCP (short for extended copy protection).

What XCP Does

The XCP rootkit prevents a user from copying a music CD while allowing music to play the CD. To do so, it needs its own special player to play the CD. But XCP interferes with any other access to the secured music CD by garbling the result in trying to read from the CD.

The rootkit will mount itself when the CD is first inserted on the PC drive. To do this, XCP relies on Windows' "helpful" feature: Windows searches for a file with a particular name with "autorun," and if it finds it, it opens and executes the file without user intervention. (The file name can be modified in Windows, but autorun.exe is by default.) You can disable the autorun function.

XCP will hide from the user so the user can't delete it. It blocks the view of any program whose name begins with \$sys\$ (which is how it is named). Sadly for Sony, this function not only concealed XCP, but any software beginning with \$sys\$ from any source, malicious or not. Any virus writer might, for example, hide a virus by calling it \$sys\$virus-1.

Sony did two things wrong: First, as we just noted, it distributed code that unintentionally exposes the network of an innocent user to other malicious code writers infection. Second, Sony installs the code without user awareness, far less permission, and employs techniques to avoid removing the code.

Patching the Penetration

The storyline of XCP became very public within November 2005 when Russinovich described what he located and several news solutions picked up the account. Confronted with serious negative marketing, Sony decided to launching an uninstaller for the particular XCP rootkit. Remember, on the other hand, from the start associated with this chapter why "penetrate and patch" was left behind as a security approach? The pressure for the quick repair sometimes directed to shortsighted solutions of which addressed instant situation in addition to not the underlying lead to: Fixing one problem usually caused a failure anywhere else.

Sony's uninstaller by itself opened serious security slots. It was presented as being a web page that saved and executed the deletion. However the programmers did certainly not check what code they will were executing, hence the website page would run any kind of code from any resource, not just the planned uninstaller. And worse, typically the downloading code remained sometimes after uninstalling XCP, which means that the vulnerability remained. (In fact, Sony applied two different rootkits coming from two different sources and even, remarkably, the uninstallers with regard to both rootkits had this particular same vulnerability.)

The number of computers were infected with this rootkit? Nobody knows without a doubt. Kaminsky found 500, 000 referrals in DNS tables towards the site the rootkit connections, but some of individuals DNS entries could help accesses by hundreds or perhaps thousands of computers. The number of users of computers where the rootkit was mounted are aware of this? Again nobody knows, or does anybody know exactly how many of those installation may not yet have already been removed.

Privilege Escalation

Their access rights and privileges are governed by that context. Most programs run in the user's background. If system access rights are set correctly, you can create, alter, or delete things that you own, however essential system objects are protected outside of your context. Malicious code writers want to control not only your objects, but also those outside your context. To do this, the

malicious code will run higher than you have. A privilege escalation attack is a way of running malicious code by a user with lower privileges, but with higher privileges.

A Privilege Escalation Example

Symantec revealed a patch to a program vulnerability in April 2006 (Sym06-007). Symantec provides security software such as virus scanners and blockers, spam filters e-mail, and network integrity tools. To ensure that a user's software still has up-to-date code and data support (such as virus specification files), Symantec has a Live Update option whereby the software regularly retrieves and installs new versions from Symantec place. A user can also invoke Live Update anytime to get up-to - date notifications. The Live Update function has to run with high privileges as it must download and install programs in the system program directory. The upgrade process actually includes running multiple programs we call LU1, LU2, Sys3, and Sys4; LU1 and LU2 are Live Upgrade components, and Sys3 and Sys4 are standard operating system components. Such four pieces complete download and install.

Operating systems use a scan route to locate executable programs. The search path is a list of directories or folders to locate a program named. When program A calls program B, the OS looks for B in the first directory listed in the search path. If the operating system detects such a program, it executes it; otherwise it continues to look in the search path in the corresponding directories until it detects B or fails to locate B by the end of the list. The OS uses the first B it detects. The user can modify the search path to run a user's program B in another directory instead of another same name program. For example, you can specify the position of a program directly, c:\program files\symantec\LU1 to monitor exactly which version runs.

In some Macintosh updates, Symantec allowed Live Update to find programs from the search path, instead of explicit position. Note the Live Update runs with elevated privileges, transferring those elevated privileges to Sys3 and Sys4. But if the user sets a search path starting in the user's space, and the user has a program called Sys3, the user's Sys3 version runs with elevated privileges.

Effect of Privilege Escalation

A malicious code writer likes privilege escalation. Creating, installing or changing a system file is difficult, but loading a file into user space is simpler. In this example, the malicious code writer must build only a small shell program, name it Sys3, store it anywhere (including in a temporary directory), reset the search path, and invoke a program (Live Update). These acts are common for non-malicious downloaded code.

The effect of running this attack is that the malicious version of Sys3 receives control in privileged mode and can overwrite operating system files, download and install new code, change system tables, and inflict virtually any other damage. After running once with higher privilege, the malicious code will set a flag for future elevated privileges.

Interface Illusions

Elias Levy borrows this attack's name. An interface illusion is a spoofing attack where all or part of a web page is incorrect. The attacker's purpose is to persuade the consumer to do anything illegal, such as entering personal banking details on a site that is not the bank's, clicking yes on a button that actually means no, or simply scrolling the screen to trigger an event that causes malware to be installed on the victim's computer. Levy's excellent article offers some fantastic examples.

The problem is every screen dot is addressable. So, if a genuine interface can paint dot 17 red, a malicious interface can. Because a malicious interface can display fake address bars, scroll bars that are not scroll bars, and even a display that looks identical to the real thing, because the attacker wants to be identical in all ways.

Nothing's different, of course. People assiduously save copies of e-mail messages as proof of receiving such a message when a simple text editor actually produces any authentic message you want. System pranksters like sending facetious messages to unsuspecting users, warning the computer is irritated. These all stem from the same point: there's nothing special, there's no reliable way to be a private and legitimate contact channel directly to the user.

Keystroke Signing

Remember the movies in which a detective will spot a note pad on a desk, keep it to the camera, and read the vague indication of a message written, and then rip it off? That tactic also has a computer counterpart.

Next, understand that there is no clear path between a key on your keyboard and the software (say a word processor) that manages the keystroke. When you press A, it triggers a switch that produces a signal that a computer driver receives, converts and analyzes and transmits until eventually your word processor receives A; there is even more conversion, analysis and transmission before A appears on your screen. Many programs function in this row. You could adjust a program at many points along the way so that when you pressed W, A would appear on the screen.

When all programs function as planned, they effectively receive and submit characters and discard each character as soon as it's sent and another arrives. A malicious software called a keystroke logger maintains a surreptitious snapshot of pressed keys. Most keystrokes are uninteresting, but we may want to protect identification numbers, authentication strings, and love notes privacy.

A keystroke logger may be independent (retaining a log of each key pressed) or connected to a certain program, retaining data only when a particular program (such as a banking application) runs.

Man-in-the-Middle Attacks

A keystroke logger is a special man-in-the-middle attack type. There are two variants of this attack: we cover the application form here, then extend the network principle

A man-in-the-middle attack is one in which a malicious program communicates with two other applications, usually between a user's input and an application's outcome. One example of a man-in-the-middle attack might be a program running between your word processor and the file system, and any time you thought you were saving your file, the middle machine stopped it, or scrambled your text or encrypted your file. What ransom would you pay to get back the paper you had been working on last week?

Timing Attacks

Computers are quick, operating much faster than humans can obey. However, as we all know, the time a computer takes to perform a task depends on the size of the task: creating 20 database records takes almost twice as long as creating 10. Or, at least in principle, if we could reliably calculate machine time and monitor certain items being performed in the device, we could infer the size of machine input. For certain cases, scale is fairly uninteresting. But even the smallest details can be important in cryptography.

Brumley and Boneh studied a web site-encryption RSA system. The authors seek to extract the key from successive assumptions of increasing importance as key possibilities. While the attack specifics are beyond this book's reach, the aim is to use a trick to maximize RSA encryption. Over-simplified, encryption with numbers less than the key takes longer periods of time as the numbers reach the key, but then the time to encrypt drops dramatically once the key value is transferred. Guessing brute force is prohibitive in time. But the authors prove you needn't try all the principles. From the left, you infer the key a few bits (most significant bit). You could try 00xxx, 01xxx, 10xxx, and 11xxx, finding that the measurement time rises from 00xxx to 01xxx, rises from 01xxx to 10xxx, and falls from 10xxx to 11xxx. The key value is between 10xxx and 11xxx. The attack works with much longer keys (about 1000 bits), and the authors use about one million possibilities for the xxx section. Also, this method helps writers to infer the key at a time, all depending on how long the encryption takes. The authors conducted their experiments on a network, not with precise local timing instruments, and still could deduce keys.

Cryptography is the primary field where speed and size are not disclosed information. But you should know that malicious code will undetect similar attacks.

Covert Channels: Programs That Leak Information

So far, we've looked at malicious code performing inappropriate acts. First, we're moving to services that transmit information to those who shouldn't. Communication moves unseen, preceding other, more appropriate messages. The term for these exceptional contact routes is covert networks. A covert channel definition comes from a Lampson paper; Millen provides a strong covert channel taxonomy.

Suppose a group of students prepares for an exam for which each question has four options (a, b, c, d); one group member, Sophie, understands the material completely and decides to help others. She says she'll show the answers to the questions, so coughing "a" once, sighing "b," and so on. Sophie uses a contact system that outsiders can not notice; an open system hides her communications. It's a human example of a secret outlet.

We explain how a programmer can construct hidden channels. The attack is more complicated than a single programmer accessing a data source. A programmer with direct access to data may normally read the data, copy it to another file, or print it out. When, for example, the programmer is one step away from the data, without the company possessing the data programmer must find out to get to the data. One way is to supply a bona fide system with an integrated Trojan horse; once the horse is activated, it finds and transmits data. However, producing a report labeled "Send this report to Jane Smith in Camden, Maine" would be too bold; the programmer must plan to collect data more subreptitiously. Covert channels are a clandestinely extracting data.

Figure 3-11 shows a "service program" with a Trojan horse attempting to copy information from a legitimate user (who is allowed access to information) to a "spy" (who should not be allowed access to information). The user does not know a Trojan horse is working and may not be in conspiracy to leak spy information.

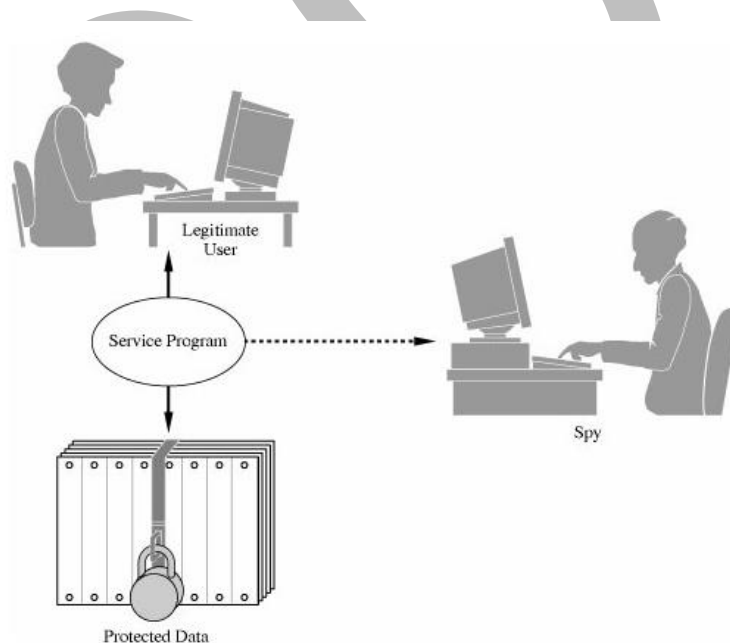


Figure 3.11. Covert Channel Leaking Information.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Covert Channel Overview

A programmer does not have access to confidential data collected after the software is placed into operation. For example, a bank programmer has no need to access names or balances in depositor accounts. Programmers for a securities company do not know what customers will purchase and sell orders. Access to real data can be justifiable during system review, but not after approving the software for daily use.

Still, a programmer can benefit from knowing that a customer is about to sell a large amount of a particular stock or that a large new account has just been opened. Often a programmer might want to create a system that secretly communicates the data it works on. In this case, the programmer is the "spy" and the "user" who runs the program written by the programmer

How to Create Covert Channels

A programmer will still find ways to covertly communicate data values. Running a program generating a particular production report or showing a value might be too obvious. For example, in some installations, security staff may sometimes review a printed report before it is distributed to its intended recipient.

If the printing of data values becomes too evident, the programmer may encode data values in another harmless report by varying output size, modifying line lengths, or printing or not printing those values. Changing the word "TOTAL" to "TOTALS" in a heading would not be heard, but this creates a 1-bit covert channel. The S's absence or presence conveys some detail. Numeric values can be placed in insignificant output field positions and can alter the number of lines per row. Such subtle channels are shown in Figure 3-12.

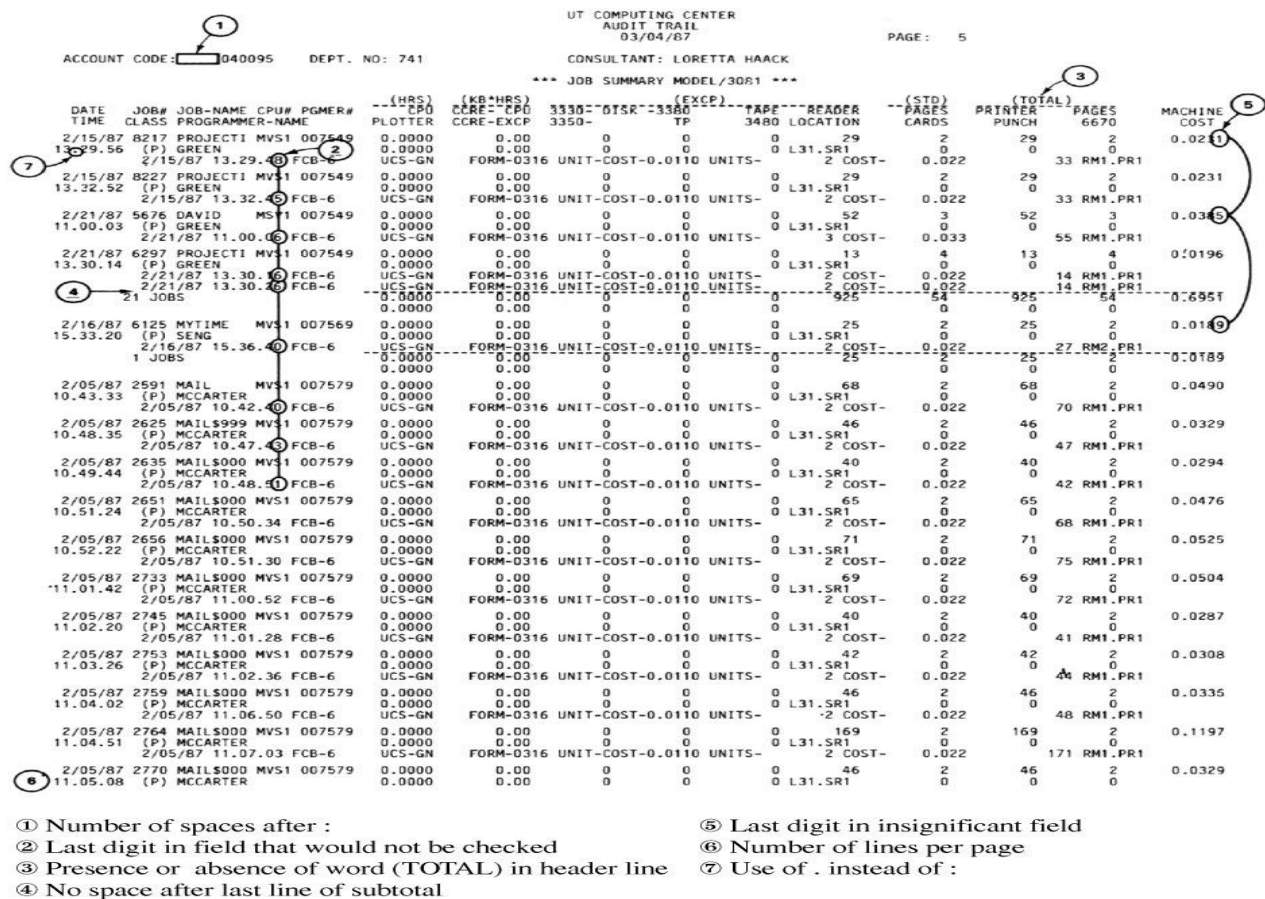


Figure 3.12. Covert Channels.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleege

Storage Channels

Many covert channels are named storage channels because these people pass information when using the occurrence or absence of items in storage.

A basic sort of a hidden channel will be the file fasten channel. In multiuser devices, files could be "locked" in order to prevent two people through writing to the similar file at the exact same time (which could dodgy the file, if anyone writes over some involving the particular other wrote). The particular main system or database managing system allows only 1 program to write into a file at an occasion by blocking, delaying, or even rejecting write requests coming from other programs. A concealed channel can signal a single bit of information simply by whether or not some sort of file is locked.

Bear in mind that the service plan has a Trojan horse composed by the spy yet run by the unsuspecting customer. As shown in Figure 3-13, the service software reads confidential data (to which the spy must not have access) and indicators the data one little bit at any given time by

locking or perhaps not locking some data file (any file, the items of which are human judgments and not even modified). The service program plus the spy need a new common timing source, damaged into intervals. To sign a 1, the power program locks the data file for the interval; with regard to a 0, your locking mechanism. Later in the period, the spy tries in order to lock the file on its own. When the spy program is unable to lock the file, that knows the service plan must have locked the particular file, and thus typically the spy program concludes typically the service program is signaling a one; if the secret agent program can lock the particular file, it knows the particular service program is signaling a 0.

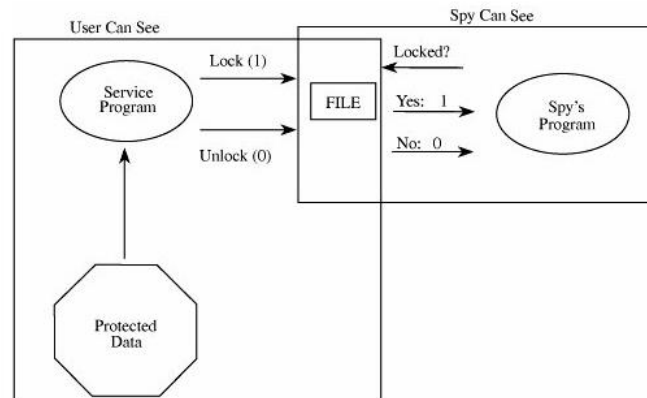


Figure 3.13. File Lock Covert Channel

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The similar approach can end up being used with disk storage area quotas or other sources. With disk storage, the particular service program signals an one by creating an massive file, so large that will it consumes most regarding the available disk room. The spy program after tries to create some sort of large file. If this succeeds, the spy software infers that the power program did not produce a large file, plus so the service plan is signaling a zero; otherwise, the spy software infers an one. Similarly typically the existence of a data file or other resource regarding a particular name can easily be used to sign. Observe that the spy does indeed not need usage of a new file itself; the only existence of the record is adequate to sign. The spy can identify the presence of some sort of file it cannot study by trying to make a file of the particular same name; if typically the request to create is definitely rejected, the spy decides that the utility plan has such a data file.

To signal more compared to one bit, the services program and the criminal program signal one tad in each time period. Figure 3.14 shows the service program signaling typically the string 100 by toggling the existence of a new file.

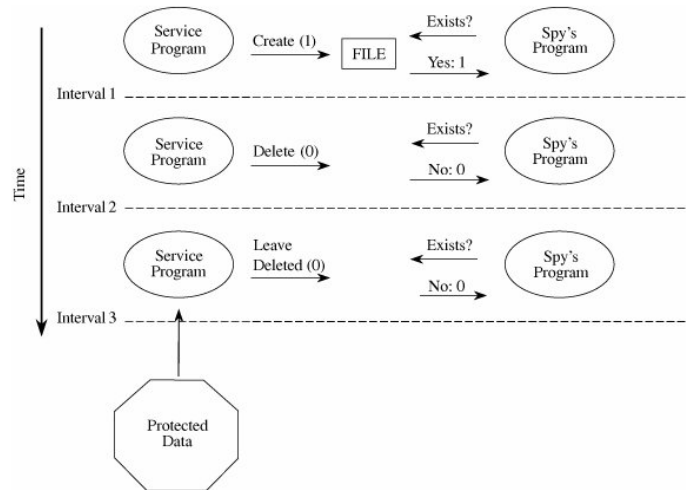


Figure 3.14. File Existence Channel Used to Signal 100.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Inside our final example of this, a storage channel utilizes a server of special identifiers. Recall that many bakeries, banks, as well as other professional establishments have a device to distribute numbered seat tickets so that customers could be served within the purchase in which they appeared. Some computing systems offer a similar server of distinctive identifiers, usually numbers, utilized to name temporary files, in order to tag and track emails, or to record auditable events. Different processes could request the next special identifier from your server. Yet two cooperating processes could use the server in order to send a signal: Typically the spy process observes regardless of whether the numbers it will get are sequential or regardless of whether a number is lacking. A missing number indicates that the service plan also requested a range, thereby signaling 1.

Inside all of these good examples, the service program plus the spy need gain access to to a shared useful resource (such as a data file, or even knowledge associated with the existence of some sort of file) and a contributed sense of time. Because shown, shared resources happen to be common in multiuser conditions, where the resource could possibly be as seemingly innocuous since whether a file is available, a tool is no cost, or space remains upon the disk. An origin regarding shared time is additionally usually available, since many courses need access to typically the current system time to be able to set timers, to document the time at which in turn events occur, or in order to synchronize activities. Karger and even Wray offer a real-life illustration of a covert route in the movement of the disk's arm and next describe ways to control the potential information seepage from this channel.

Moving data one bit each time must seem awfully gradual. But computers operate from such speeds that your minuscule rate of one bit per millisecond (1/1000 second) would never become noticed but could effortlessly be handled by 2 processes. In which rate regarding 1000 bits per 2nd (which is unrealistically conservative), this entire book can be leaked in concerning two days.

Increasing the pace by an order regarding magnitude or two, which often is still quite conventional, reduces the transfer moment to minutes.

Time Channels

Other covert stations, called timing channels, go away information by using typically the speed where things take place. Actually, timing channels happen to be shared resource channels throughout which the shared reference is time.

A support program uses a time channel to communicate by simply using or not utilizing an assigned amount of work time. In the basic case, a multiprogrammed program with two user functions divides time into obstructions and allocates blocks associated with processing alternately to 1 process and the some other. A process is presented processing time, but in the event that the process is awaiting another event to arise and contains no processing to be able to do, it rejects typically the offer. The service procedure either uses its stop (to signal a 1) or rejects its stop (to signal a 0). Such a situation is usually shown in Figure 3.15, first with the support process and the spy's process alternating, and after that with the service procedure communicating the string info to the spy's procedure. In the second element of the example, the particular service program wants to be able to signal 0 in the particular third time block. That will do this through the use of just enough time in order to determine it wants to be able to send a 0 plus then pause. The criminal process then receives handle for the remainder involving the time block.

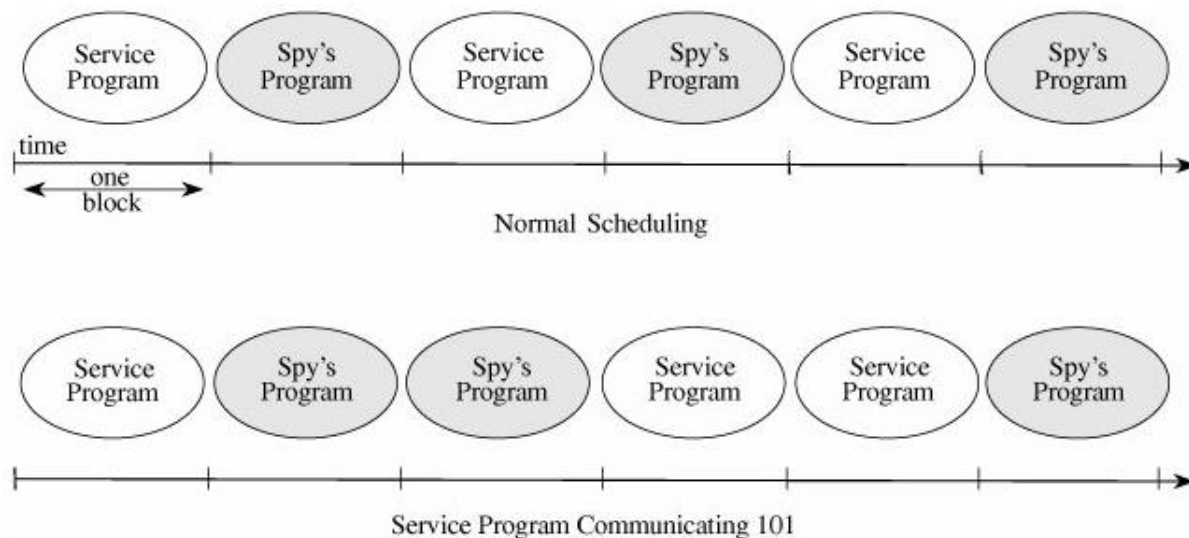


Figure 3.15. Covert Timing Channel.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Thus far, all examples have got involved just the assistance process and the spy's process. In fact, multiuser computing systems typically have got more than just a couple of active processes. The just complications added by even more processes are that typically the two cooperating processes

should adjust their timings plus deal with the feasible interference from others. Intended for instance, with the exclusive identifier channel, other operations will also request verifications. If on average d other processes will obtain m identifiers each, then your service program will obtain over $n*m$ identifiers regarding a 1 and not any identifiers for a zero. The gap dominates the particular result of all various other processes. Also, the services process plus the spy's method can use sophisticated code techniques to compress their particular communication and detect and deal with transmission errors caused simply by the consequence of other unrelated techniques.

Identifying Potential Covert Channels

In this particular description of hidden channels, ordinary things, many of these as the existence associated with a file or period used for a calculation, happens to be the medium via which a covert station communicates. Covert channels happen to be not easy to discover because media are therefore numerous and often utilized. Two relatively old strategies remain the standards regarding locating potential covert stations. One works by examining the time of the system, and the various other works at the resource code level.

Shared Resource Matrix

Since the foundation of a covert funnel is a shared source, the search for possible covert channels involves locating all shared resources plus determining which processes may write to and examine through the resources. The approach was introduced by Kemmerer. Although time-consuming, the technique can become automated.

To make use of this method, you construct a matrix of resources (rows) and even processes that can gain access to them (columns). The matrix entries are R with regard to "can read (or observe) the resource" and Meters for "can set (or modify, create, delete) the particular resource. " For an example of this, the file lock funnel has got the matrix shown inside Table 3.3. You then look for two columns and two rows having the following pattern:

	Service Process	Spy's Process
Locked	R, M	R, M
Confidential data	R	

Table 3.3. Shared Resource Matrix

	M		R	
	R			

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

This particular pattern identifies two solutions and two processes in a way that the second process will be not in order to read through the second resource. Nevertheless, the first process can easily pass the information in order to the second by looking at the second useful resource and signaling the info by way of the first resource. As a result, this pattern implies the actual information flow as displayed here.

	M		R	
	R		R	

Next, you full the shared resource matrix by adding these meant information flows and studying the matrix for undesired flows. Thus, you can easily tell the spy's method can read the secret data by using the covert channel through the particular file lock, as displayed in Table 3.4.

	Service Process	Spy's Process
Locked	R, M	R, M
Confidential data	R	R

Table 3.4. Complete Information Flow Matrix.

Above Table taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Data Flow Approach

Denning derived a new technique for flow research from a program's format. Conveniently, this analysis could be automated within a compiler to ensure that information flow possibilities could be detected while some sort of program is under enhancement.

Using this method, we could recognize nonobvious flows details between statements in the program. For example, all of us know that the affirmation $B := A$, which designates the value of A new to the variable M, obviously supports an information flow from A in order to B. This type involving flow is called a good "explicit flow." Likewise, the pair of claims $B := A$; $C := B$ indicates an information movement from A to Chemical (by way of B). The conditional statement IN THE EVENT THAT $D=1$ THEN $B := A$ has two flows: coming from a to B because involving the assignment, but furthermore from D to M, because the value associated with B can change when and only when the price of D is one This second flow is known as an "implicit flow. inches

The statement $B := fcn(args)$ supports an information circulation in the function fcn to be able to B. At a succinct, pithy level, we could say that will there is a probable flow from the fights args to B. Even so, we could more carefully analyze the function to be able to determine whether the function's value depended on most of its arguments in addition to whether any global ideals, not part of typically the argument list, affecting the particular function's value. These details flows can be followed from the bottom upwards: At the bottom right now there must be functions that will call no other attributes, and can analyze all of them after which use those benefits to analyze the features that call them. Simply by looking at the fundamental functions first, we may say definitively whether at this time there is a potential info flow from each disagreement to the function's end result and whether there are usually any flows from international variables. Table 3-5 databases several samples of syntactic

Statement	Flow
$B := A$	from A to B
IF $C=1$ THEN $B := A$	from A to B; from C to B
FOR $K:=1$ to N DO stmts END	from K to stmts
WHILE $K>0$ DO stmts END	from K to stmts
CASE (exp) val1: stmts	from exp to stmts
$B := fcn(args)$	from fcn to B
OPEN FILE f	none

READ (f, X)	from file f to X
WRITE (f, X)	from X to file f

Table 3-5 databases several samples of syntactic

Above Table taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

format analysis area of compilation. This particular analysis may also be performed upon the higher-level design standards.

Covert Channel Conclusions

Covert Channel represent a true danger to secrecy in data systems. A covert station attack is fairly superior, but the basic strategy is just not beyond the abilities of even a general programmer. Considering that the subverted system can be practically virtually any user service, such while a printer utility, growing the compromise can always be as easy as growing a virus or any kind of another kind of Trojan malware horse. And up in order to date experience has displayed how readily viruses might be planted.

Capacity in addition to speed does not issue; our estimate of a multitude of bits per second is usually unrealistically low, but perhaps at that rate a great deal information leaks swiftly. Using modern hardware architectures, selected covert channels inherent within the hardware design possess capacities of millions associated with bits per second. And even the attack does not really require significant finance. As a result, the attack could end up being very effective in most circumstances involving highly sensitive information.

For these reasons, protection researchers have worked faithfully to develop tips for concluding covert channels. The drawing a line under results have been annoying; in ordinarily open surroundings, there is essentially little control of the agitation, destabilization of a utility system, nor is there a good effective way of verification such programs for concealed channels. And other within a few very large security systems, systems are unable to control the flow associated with information from a hidden channel. The hardware-based programmes may not be closed, given the particular underlying hardware architecture.

3.5. Controls Against Program Threats

The particular style we have simply described is simply not pretty. Presently there are many ways an application can fail and a lot of methods to turn the fundamental faults into security problems. It is needless to say much better to focus on elimination than cure; how carry out we use controls in the course of software developmentthe specifying, building, writing, and testing regarding the programto find plus eliminate the sorts associated with exposures we certainly have discussed? Typically the discipline Software engineering details this question more internationally, devising methods to ensure typically the quality of software. we provide a good overview of several approaches that can prove beneficial in finding and correcting security flaws

In this section performing at three types associated with controls: developmental, operating method, and administrative. We go over each in return.

Developmental Controls

Many controls can end up being applied during software enhancement to ferret out in addition to fix problems. So allow us begin by searching at the size regarding development itself, to discover what tasks are engaged in specifying, designing, making, and testing software.

The Nature of Software Development

Software development is usually considered solo effort; the programmer sits with a new specification or design in addition to grinds out line following line of code. But also in fact, software development is really a collaborative effort, involving individuals with different skill sets that combine their expertise to make a working product. Development needs people who can

designate the machine, by capturing the particular requirements and building a new model of how typically the system should work by the users' point involving view

design the method, by proposing an answer to the problem referred to by the requirements plus creating a model involving the solution

implement the machine, by using the style as a blueprint with regard to building a working option test the system, in order to ensure that it fits the requirements and accessories the solution as named for within the design

evaluation the system at numerous stages, to make positive that the final products are generally consistent with the requirements and design models

record the device, so that customers can be trained plus supported

manage the machine, to be able to estimate what resources can be needed for advancement also to track when the particular system is going to be done

preserve the system, tracking difficulties found, changes needed, in addition to changes made, and analyzing their effects on total quality and efficiency

One particular person could do each one of these things. But more usually than not knowing, some sort of team of developers performs together to execute these duties. Sometimes a team associate does more than 1 activity; a tester could take part in the requirements review, for example, or an implementer can certainly write documentation. Each staff is different, and staff dynamics play a significant role in the team's success.

Keep in head the kinds of superior attacks described in the particular previous section. Balfanz reminds us all that we must design and style systems that are equally secure and usable, advocating these points:

- You still cannot retrofit usable security.
- Resources aren't solutions.
- Mind the particular upper layers.
- Keep your buyers satisfied.
- Think locally; work locally.

We can analyze product and process to view how both contribute to be able to quality and in specific to security as being an element of quality. Let people start out with the product, to be able to get a sense teaching how we recognize superior quality secure software.

Modularity, Encapsulation, and also the exact product information Hiding

Program code usually includes a long shelf-life and is enhanced more than time as needs alter and faults are found out and stuck. For this kind of reason, a key basic principle society engineering is to be able to create a design or perhaps code in small, plus self-contained units, called parts or modules; when a strategy is written this way, many of us declare it is flip. Modularity offers advantages with regard to program development generally in addition to security in particular.

In case a component is isolated through the effects of other pieces, it is easier to be able to trace a problem for the fault that caused that and to limit the particular damage the fault reasons. It is also much easier to maintain the system, considering that becomes an isolated part never affect other elements. Plus its easier to discover where vulnerabilities may lay if the component will be isolated. We call this kind of isolation encapsulation.

Information hiding is another characteristic involving modular software. When info is hidden, each part hides its precise rendering or some other style and design decision from the some others. Thus, each time a change is certainly needed, the overall style can remain intact whilst only the necessary modifications are created to particular components.

Allow us take a look at these attributes in more detail.

Modularity

Modularization is the method of dividing a process into subtasks. This section is done on some sort of logical or functional base. Each component performs a new separate, independent part associated with the task. Modularity is usually depicted in Figure 3.16. The goal is in order to have each component fulfill four conditions:

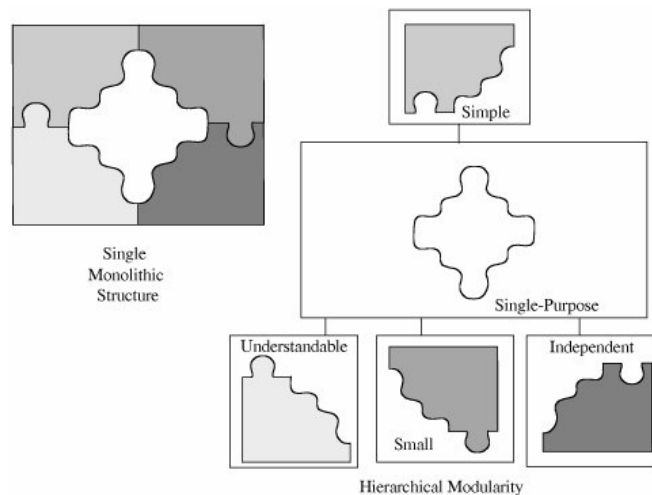


Figure 3.16. Modularity

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

- **single-purpose:** functions one function
- **small:** is composed of some information intended for which a human could readily grasp both framework and content
- **simple:** benefits a low degree involving complexity in order that a man can readily be well known with purpose and composition with the module independent: works a job isolated through other modules

Other element characteristics, such as possessing a single input and one output or using the limited set of encoding constructs, indicate modularity. By a security standpoint, modularity should improve the chance that an implementation is usually correct.

Particularly, smallness is definitely an important quality that will help security analysts understand exactly what each component does. Of which is, in good software program, design and program devices should be only since large as required to execute their required functions. Presently there are several advantages in order to having small, and 3rd party components.

Maintenance. In case a part implements a single operate, it can be substituted easily which has a revised 1 if necessary. The newest element may be needed because of to a change throughout requirements, hardware, or atmosphere. Sometimes the replacement is definitely an enhancement, using the smaller, faster, more appropriate, or else better module. Typically the interfaces between

this element plus the remainder of typically the design or code are usually few and well defined, so the associated along with the replacement are noticeable.

- **Understandability.** A system consisting of many small elements is usually simpler to know than one large, unstructured block of code.
- **Recycling.** Components developed for starters aim can often be used again consist of systems. Recycle of correct, existing design and style or code components can easily significantly lower the difficulty associated with implementation and testing.
- **Correctness.** A failure could be rapidly traced to its trigger if the components conduct only one task every.
- **Testing.** A single aspect with well-defined inputs, results, and function could be examined exhaustively by itself, without having concern for its outcomes on other modules (other compared to expected function and even output, of course).

Safety analysts must be capable to understand each element as an independent device and be assured associated with its limited effect in other components.

A flip component usually has large cohesion and low joining. By cohesion, we result in that all the components of an element have a new logical and functional purpose for being there; each factor of the part is associated with the component's single purpose. A remarkably cohesive component contains a substantial degree of focus upon the reason; a low level of cohesion signifies that the particular component's contents is surely an not related jumble of actions, frequently put together because involving time-dependencies or convenience.

Joining appertains to the education with which an aspect is determined by other components throughout the system. Thus, reduced or loose coupling is definitely better than high or perhaps tight coupling because the particular loosely coupled components are really free from unwitting disturbance from other components. This specific difference in coupling is usually shown in Figure 3.17.

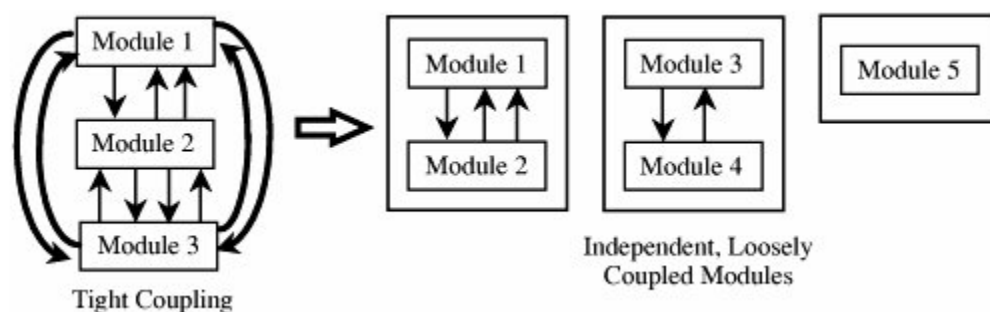


Figure 3.17. Coupling.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Encapsulation

Encapsulation hides the component's implementation details, although it does not automatically mean complete isolation. A lot of components must share details with other components, generally with good reason. Nevertheless, this sharing is thoroughly documented so that some sort of component is affected just in known ways simply by others inside the system. Discussing is minimized so of which the fewest interfaces potential are used. Limited terms lower the number of concealed channels that could be constructed.

A good encapsulated component's protective border can be translucent or perhaps transparent, as needed. Berard paperwork that encapsulation could be the "technique for packaging the info [inside a component] in such an approach as to hide just what should be hidden make visible what is designed to be visible. very well.

Information Hiding

Developers who else work where modularization is definitely stressed can be certain that other components may have limited effect upon the ones they create. Thus, we can believe of a factor as the kind of black field, with certain well-defined plugs and outputs and some sort of well-defined function. Other components' designers do not want to know how the particular module completes its perform; it really is enough to get assured that the aspect performs its task in certain correct manner.

This concealment is the information concealing, depicted in Figure 3.18. Information hiding is appealing because developers cannot very easily and maliciously alter typically the components of others when they do not recognize how the components does job.

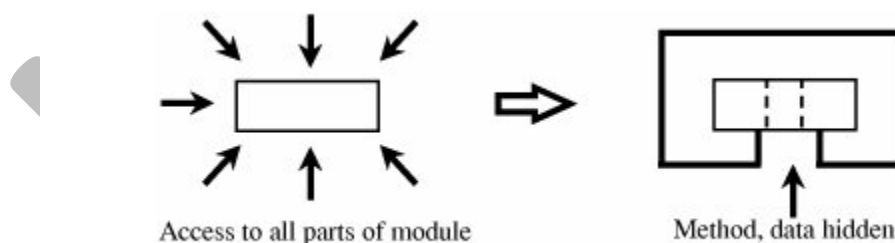


Figure 3.18. Information Hiding.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

These kinds of three characteristicsmodularity, encapsulation, and even also the precise item information hidingare fundamental guidelines society engineering. They will be also good security methods because they lead in order to modules that can get understood, analyzed, and relied on.

Mutual Suspicion

Programs are usually not always trustworthy. In spite of an operating system to be able to enforce access limitations, that may be impossible or perhaps infeasible to bound typically the access privileges of a great untested program effectively. Within this case, an individual Circumstance is

legitimately worried concerning a new program S. Yet , program P might be invoked by one more program, Q. There is usually no way for Queen that P is right or proper, any even more than an user perceives that of P.

Consequently, we use the principle of mutual suspicion to be able to describe the relationship in between two programs. Mutually shady programs operate as when other routines in the particular system were malicious or even incorrect. A calling plan cannot trust its referred to as subprocedures to be right, and a called subprocedure cannot trust its phone program to be appropriate. Each protects its program data so the other provides only limited access. Regarding example, a procedure to be able to sort the entries in the list cannot be trustworthy never to modify those components, while that procedure are not able to trust its caller in order to provide any list from all or to offer you the number associated with elements predicted.

Confinement

Confinement is a technique applied by an operating program on a suspected software. A confined program will be strictly limited in exactly what system resources it can certainly access. If the program will be not trustworthy, the information that can access are firmly limited. Strong confinement can be helpful in limiting the particular spread of viruses. Due to the fact a virus spreads simply by means of transitivity and even shared data, every one of the information and programs in a solitary compartment of a limited program can affect just the data and plans in the same area. Therefore, the virus can certainly spread only to items for the reason that compartment; it are unable to get outside the area

Genetic Diversity

At the local electronics shop an individual can buy a blend printer scanner copier fax machine. It arrives at a good cost (compared to costs in the four separate components) since there is considerable overlap in operation among those four. This is compact, and an individual need only install 1 thing on your program, not four. But in the event that any part of this fails, you lose a new lot of capabilities almost all at once.

Related in order to the argument for modularity and information hiding and even reuse or interchangeability regarding software components, some folks recommend genetic diversity: that is risky having a lot of components of something take place from one source, i have heard it said.

Geer at al. wrote a written report examining the monoculture regarding computing dominated by one particular manufacturer: Microsoft today, APPLE yesterday, unknown tomorrow. These people are at the similar in agriculture where the entire crop is susceptible to a single virus. Malicious code from the particular Morris worm to typically the Code Red virus has been especially harmful just because an important proportion of the earth's computers ran versions involving the identical operating

Tight integration involving products is a comparable concern. The Windows running system is tightly connected to Internet Explorer, any office Suite, and the View e-mail handler. A weaknesses in a of these could also affect the some others. Because of the small integration, fixing a weakness in one will surely have a great impact on the other people, whereas a vulnerability

inside another vendor's browser, with regard to example, can affect Expression only to the level they communicate through a new well-defined interface.

Peer Critiques

We turn next in order to the process of growing software. Certain practices in addition to techniques can assist all of us in finding real and even potential security flaws (as well as other faults) and fixing them prior to we turn the device more than to the users. If recommend several major tactics for building what that they call "solid software":

- expert reviews
- hazard analysis
- assessment
- good design and style
- prediction
- permanent analysis
- configuration management
- evaluation of errors

Here, all of us look at each training briefly, and we explain its relevance to safety measures controls. We begin using peer reviews.

You have got probably been doing many type of review for since many years as a person have been writing signal: desk-checking your work or even asking a colleague in order to look more than a routine to be able to ferret out any troubles. Today, an application assessment is associated with a number of formal process steps to be able to make it more efficient, and we review just about any artifact of the enhancement process, not just signal. But the essence associated with a review remains identical: sharing a product together with colleagues able to remark about its correctness. Generally there are careful distinctions amongst three types of expert reviews:

Review: The creature is presented informally to some team of reviewers; typically the goal is consensus plus buy-in before development earnings further.

Walk-through: The creature is presented to typically the team by its originator, who leads and handles the topic. Here, schooling is the goal, plus the focus is on learning about a solitary document.

Inspection: This a lot more formal process is some sort of detailed analysis when the creature is checked against a new prepared list of problems. The creator does not necessarily lead the discussion, in addition to the fault identification plus correction are often manipulated by statistical measurements.

- An intelligent engineer who finds some sort of fault can deal using it in at minimum three ways:
- by understanding how, when, and exactly why errors occur

- by using activity to prevent mistakes
- by simply scrutinizing products to get the instances and results of errors that have been skipped

Expert reviews address this difficulty directly. Unfortunately, many agencies give only lip assistance to peer review, plus reviews continue to be not portion of mainstream software anatomist activities.

But you will find persuasive reasons to do testimonials. An overwhelming amount involving evidence shows that several types of peer overview in software engineering may be extraordinarily effective. Regarding example, early studies with Hewlett-Packard in the nineteen eighties revealed that those designers performing peer review in their projects enjoyed a new significant advantage over all those relying only on standard dynamic testing techniques, regardless of whether black box or white colored box. Figure 3.19 even comes close to the fault discovery charge (that is, faults found out per hour) among white-box testing, black-box testing, home inspections, and software execution. It's clear that inspections learned far more faults throughout the same period regarding time than other choices. This end result is especially compelling for big, secure systems, where live life running for fault finding is probably not an option

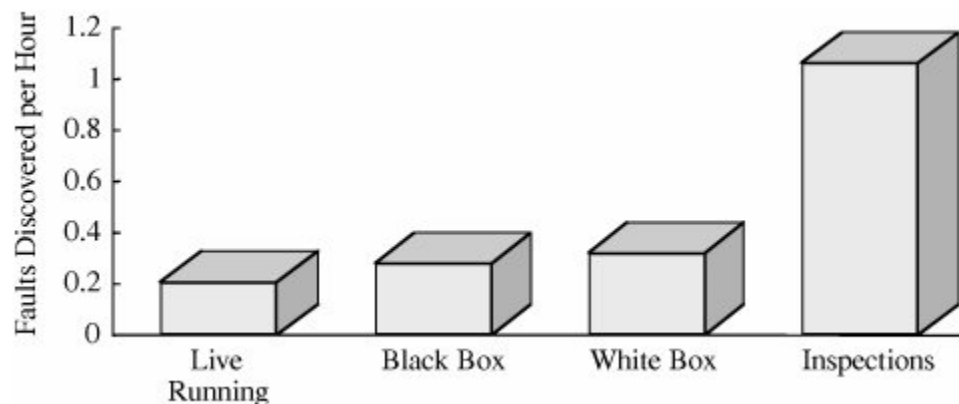


Figure 3.19. Fault Discovery Rate Reported at Hewlett-Packard.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Scientists and practitioners have consistently shown potency involving reviews. For instance, Smith described the info in his significant repository of project details to paint an image of how reviews in addition to inspections find faults within accordance with other finding activities. Because products change so wildly by dimensions, Table 3-6 presents typically the fault discovery rates comparative to the number associated with 1000s of lines of code inside the delivered product.

Discovery Activity	Faults Found (Per Thousand Lines of Code)
Requirements review	2.5
Design review	5
Code inspection	10
Integration test	3
Acceptance test	2

Table 3-6 presents typically the fault discovery rates comparative to the number associated with 1000s of lines of code

Above Table taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Hazard Analysis

Hazard analysis is definitely a pair of systematic techniques supposed to expose potentially unsafe system states. In specific, it can help people expose security concerns and even then identify prevention or even mitigation ways of address these people. That is, hazard examination ferrets out likely will cause of problems so that will we could then apply the appropriate way of stopping the problem or treatment its likely consequences. As a result, it usually involves establishing hazard lists, as okay as procedures for discovering "what if" scenarios in order to trigger consideration of nonobvious hazards. The sources regarding problems can be hiding in any artifacts associated with the development or servicing process, not merely in the particular code, so a danger analysis must be wide-ranging in its domain involving investigation; in other words and phrases, hazard analysis is really a technique issue, not just a new code issue. Similarly, generally there are many types of issues, ranging from incorrect program code to unclear consequences of the particular action. A very good hazard analysis takes almost all of them into mind.

Though hazard analysis is mostly fine practice on any job, it is required inside some regulated and essential application domains, and this can be invaluable with regard to locating security flaws. This is never too early on to be thinking concerning the types of hazards; typically the analysis must start when a person first start thinking concerning creating a new program or when someone offers a significant upgrade to be able to an existing system. Risk analysis should continue over the system life cycle; you have to identify potential hazards that may be introduced during system design and style, installation, operation, and preservation.

Many different techniques support the particular identification and management associated with potential hazards. Among typically the most effective are risk and operability studies (HAZOP), failure modes and results analysis (FMEA), and wrong doing tree analysis (FTA). HAZOP is a structured research technique originally developed for that process control and substance plant industries. Over the particular last several years it offers been adapted to uncover potential hazards in safety-critical software systems. FMEA will be a bottom-up technique utilized at the system element level. A team pinpoints each component's possible flaws or fault modes; the particular team then determines may trigger the fault and even exactly what systemwide results each fault might have got. By keeping system implications in mind, the group often finds possible method failures which are not really made visible by some other analytical means. FTA matches FMEA. It is a new top-down technique that commences with a postulated harmful system malfunction. Then, typically the FTA team works in reverse to identify the probable precursors to the accident. By tracing back coming from a specific hazardous failure, the team can identify unexpected contributors to incidents, and can then search for opportunities to mitigate typically the risk

These techniques is definitely clearly great for finding in addition to preventing security breaches. We all decide which strategy is usually most appropriate by knowing how much we find out about causes and results. For example, Table 3.7 suggests that when all of us know the cause in addition to effect of a provided problem, we can enhance the description of precisely how the system should react. This clearer picture will assist requirements analysts understand just how any problem is associated to other requirements. This also helps designers recognize exactly what the technique

should do testers how to verify typically system is		Known Cause	Unknown Cause	helping know check to that the behaving
	Known effect	Description of system behavior	Deductive analysis, including fault tree analysis	
	Unknown effect	Inductive analysis, including failure modes and effects analysis studies	Exploratory analysis, including hazard and operability	

appropriately. If we can explain a known effect using unknown cause, we make use of deductive techniques such while fault tree analysis in order to help us understand the particular likely causes of the particular unwelcome behavior. Conversely, we might know the cause involving a problem but is not recognize all the effects; in this article, we use inductive strategies such as failure ways

Table 3-7. Perspectives for Hazard Analysis

Above Table taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

and effects analysis in order to help us trace through cause to all or any possible results. For example, suppose all of us know that a subsystem is unprotected and may lead to securities failing, but we do not necessarily understand how that failure can impact the rest involving the system. We can easily use FMEA to produce a list of achievable effects and then assess the tradeoffs between more protection and possible troubles. Finally, to get problems in relation to which organic beef not but be aware, we will perform an exploratory evaluation, for example, a hazard and operability stud

Testing

Testing is a method activity that homes found in on product quality: building the product failure no cost or failure tolerant. Every software problem (especially any time it relates to security) has the potential not really only to make software are unsuccessful but also for detrimentally affecting a business or perhaps a life. Thomas Little, head of NASA's research of the Mars lander failure, noted that "One of the things many of us kept in mind in the course of the course of each of our review is that inside of the conduct of place missions, you get just one strike, not three. Even when thousands of functions will be performed flawlessly, just 1 mistake may be catastrophic in order to a mission" . This same sentiment applies for security: The malfunction of one control presents a vulnerability that is definitely not ameliorated by any kind of number of functioning adjustments. Testers improve software top quality by finding as several faults as you possibly can and by simply writing up their studies carefully so that programmers can locate the leads to and repair the troubles if possible.

Tend not to dismiss a point from Thompson : Security assessment is hard. Side outcomes, dependencies, unpredictable users, in addition to flawed implementation bases (languages, compilers, infrastructure) all bring about to this difficulty. Although the essential complication along with security testing is of which we cannot look in just the behavior the program gets correct; we also have to be able to look for the 100s of ways the plan might go wrong.

Screening usually involves several levels. First, each program element is tested on the own, isolated from the particular other components in typically the system. Such testing acknowledged as module testing, aspect testing, or unit examining, verifies that the element functions properly with typically the sorts of input expected by a study of typically the component's design. **Unit testing** is done in some sort of controlled environment whenever achievable so that the check team can feed the predetermined set of files for the component being examined and observe what end result actions and data will be produced. In addition, the particular test team checks typically the internal data structures, reasoning, and boundary conditions for that input and output info.

When collections of elements have been put through product testing, the next stage is making certain the barriers among the components are usually defined and handled effectively. Indeed, interface mismatch may be a significant safety vulnerability.

Integration testing is definitely the procedure for verifying that will the system components communicate as described in the particular system and program design and style specifications.

Once we will be sure that information is definitely passed among components inside accordance with the style, we test the program to ensure that that has the required functionality.

The **function test** evaluates the particular system to determine regardless of whether the functions described simply by the requirements specification will be actually performed by the particular integrated system. The end result is a functioning program.

The function test comes anywhere close to the system being constructed with the functions described within the developers' requirements specification. And then, a **performance test** comes anywhere close the system with typically the remainder of the software in addition to hardware requirements. It is definitely during the function and gratification tests that security demands are examined, and the particular testers confirm that typically the system is as protected as it is needed to be.

Once the functionality test is complete, builders are certain that the device functions according to their particular comprehension of the system explanation. The next step is definitely conferring with the buyer to make certain that will the device works according in order to customer expectations. Developers sign up for the customer to accomplish a good **acceptance test** when the technique is checked against typically the customer's requirements description. On completing acceptance testing, typically the accepted system is set up in the environment inside which it will end up being used. One final **installation test** is set you back again make sure that the particular system still functions since it should. However, safety requirements often state that will a process should not perform something

The goal of device and integration testing is usually to ensure that the particular code implemented the design and style properly; that is certainly, that the particular programmers have written codes to do what the particular designers intended. System assessment includes a very different goal: to ensure that the machine does what the consumer wants to carry out. Regression testing, an element of system testing, is certainly particularly important for safety measures purposes. After a transform is made to boost the system or fix a challenge, **regression testing** ensures of which all remaining functions happen to be still working and that will performance has not recently been degraded by the alter.

Each of the sorts of tests listed here can easily be performed from 2 perspectives: black box and even clear box (sometimes named white box). **Black-box testing** treats a system or perhaps its components as dark-colored boxes; testers cannot "see inside" the system, thus they apply particular plugs and verify that they will get the expected result. **Clear-box testing** allows presence. Here, testers can look at the design and signal directly, generating test situations using the code's actual design. Thus, clear-box testing has learned that component X employs CASE

statements and may look for instances when the input causes control to drop through to a sudden line. Black-box testing needs to rely on read more regarding the required inputs in addition to outputs because the real code is just not available regarding scrutiny.

The particular combo of techniques correct for testing a provided system depends on typically the system's size, application website, quantity of risk, and several other factors. But comprehending the effectiveness of each and every technique helps us recognize what is correct intended for each particular system. With regard to instance, Olsen describes the advancement at Contel IPC regarding a system containing 184, 000 lines of signal. He tracked faults uncovered during various activities, plus found differences:

- 17. 3 or more percent of the flaws were found during assessments with the system design
- 19. 1% during component design and style examination
- 15. 1 % during code inspection
- 30. 4 percent during the usage testing
- 16. 6 per cent during system and regression test

Only 0.1 percent of the errors were revealed after the particular system was placed inside the field. Thus, Olsen's work shows the value of using different methods to uncover different varieties of faults during enhancement; it is not adequate to count on a solitary method for catching just about all problems.

Who does the particular testing? From the security point of view, **independent testing** is very desirable; it may stop a developer from trying to hide something in a new routine or keep some sort of subsystem from controlling typically the tests that is applied in order to it. Thus, independent screening increases the likelihood a test will expose typically the result of an invisible feature.

An example regarding a testing is exclusive to computer security: **penetration testing**. Within this form involving testing, testers specifically attempt to make software fall short. That is, instead regarding testing to find out that application does do what this is expected to (as could be the goal in the particular other types of screening we just listed), the particular testers try to notice if the software truly does what it is certainly not intended to, which is definitely to fail or, especially, fail to enforce safety. Because penetration testing normally pertains to full systems, not really individual applications,

Good Design

We saw earlier found in this chapter that modularity, information hiding, and encapsulation are characteristics of very good design. Several design-related procedure activities are particularly interesting building secure software:

- by using a philosophy of fault threshold
- having a consistent coverage for handling failures
- recording the style rationale and background
- using design patterns

- All of us describe each of these types of activities in turn.

Developers should try to assume faults and handle all of them in manners that lessen disruption and maximize security and security. Ideally, we wish our system to get fault free. But throughout reality, we must think about the system will fall short, and that we make sure that will unexpected failure will not take the system down, ruin data, or destroy living. For example, rather compared to waiting for the device in order to fail (called passive wrong doing detection), we might produce the device so that that reacts in an satisfactory way to a failure's occurrence. Active fault recognition could possibly be practiced by, regarding instance, adopting a viewpoint of mutual suspicion. Rather of assuming that information passed from other devices or components are proper, we are able to always check that will the data are in bounds and of the correct type or format. We are able to also use redundancy, contrasting the outcome of two or even more processes to determine that will they agree, before we all use their result within a task.

If improving a fault is also risky, inconvenient, or costly, we can choose as an alternative to train fault tolerance: separating destruction caused by the particular fault and minimizing dysfunction to users. Although mistake tolerance is just not always consideration of as a safety technique, it supports the particular idea, discussed in Part 8, our security plan allows us to elect to mitigate the effects regarding a security problem as an alternative of preventing it. Regarding instance, rather than mount expensive security controls, all of us may choose to acknowledge the risk that essential data may be dangerous. If actually a safety measures fault destroys important information, we may decide to be able to isolate the damaged info set and automatically go back to some backup data set in place to ensure that users can carry on to perform system features.

More generally, we are able to design and style or code defensively, simply as we drive defensively, by constructing a regular policy for handling disappointments. Typically, failures include

- faltering to realise a service
- providing the particular wrong service or info
- corrupting information

We can easily build into the design and style a particular way regarding handling each problem, choosing from one of about three ways:

- **Retrying:** restoring the machine to its previous point out and performing the services again, using a diverse technique
- **Correcting:** restoring the particular system to its earlier state, correcting some technique characteristic, and performing typically the service again, using the particular same strategy
- **Reporting:** repairing the system to it is previous state, reporting the condition to an error-handling part, but not providing the services once more

This consistency involving design helps us look at for security vulnerabilities; many of us look for instances of which are different from typically the standard approach.

Design rationales and history tell individuals the reasons the technique is made one way rather of another. Such data helps us since the technique evolves, so we can easily integrate the design associated with our security functions with no compromising the integrity involving the system's overall style.

Moreover, the design historical past enables us to appearance for patterns, noting exactly what designs work best through which situations. For example, we all can reuse patterns of which have been successful throughout preventing buffer overflows, inside ensuring data integrity, or even in implementing user username and password checks.

Prediction

Among the many varieties of prediction we carry out during software development, many of us try to predict the hazards involved in building in addition to using the program. As many of us see in depth throughout Chapter 8, we should postulate which unwelcome activities might occur and in that case make plans to stop all of them or at least offset their effects. Risk conjecture and management are specifically important for security, in which we are always coping with unwanted events that possess negative consequences. Our intuitions help us decide which often controls to make use of and exactly how many. For example, anytime we think the danger of a particular protection breach is small, organic beef not want to spend a large amount regarding money, time, or energy in installing sophisticated settings. Or we may employ the likely risk influence to justify using various controls at once, a strategy called "defense in level. inch

Static Analysis

Just before a process is up and working, we could examine its style and code to find and repair security faults. We noted earlier that will the peer review procedure involves this kind regarding scrutiny. But static evaluation is somewhat more than peer evaluation, in fact it is usually performed just before peer review. We will use tools and strategies to examine you will involving design and code to be able to see if the features warn us of achievable faults lurking within. Regarding example, a large range of amounts of nesting may well indicate how the design or perhaps code is not easy to go through and understand, so that it is simple for a malicious creator to bury dangerous computer code deep within the method.

For this end, we could examine several aspects regarding the design and signal:

- control flow structure
- files flow structure
- data composition

The control flow could be the sequence in which guidelines are executed, including iterations and loops. This element of design or signal can also show just how often a particular coaching or routine is carried out.

Data flow follows the particular trail of an info item since it will be accessed and modified by system. Many times, purchases put on data are structure, and that we use data stream measures to show people how then when each files item is written, study, and changed.

The files structure is the method by which the data are structured, in addition to the particular system itself. For illustration, in the event the data are set up as lists, stacks, or even queues, the algorithms intended for manipulating them are very likely to be well realized and well defined.

There are various approaches to static research, especially because there will be so many ways to be able to create and document some sort of design or program. Computerized tools are available to be able to generate not only amounts (such as depth associated with nesting or cyclomatic number) but also graphical depictions of control flow, information relationships, and the range of paths from series of code to one more. These aids can support us observe how the flaw in one section of a system can influence other parts.

Configuration Administration

When we develop application, it is important to be able to know who is producing which changes to just what then when:

- **corrective changes:** preserving control over the system's everyday features
- **adaptive changes:** sustaining control of system changes
- **perfective changes:** perfecting current acceptable features
- **preventive adjustments:** preventing system performance by degrading to unacceptable degrees

We want a point associated with control over the computer software changes so that 1 change will not inadvertently unnecessary the effect of the prior change. And we desire to control what is usually a proliferation of various versions and releases. Intended for instance, a product may well run using several different websites or in numerous different conditions, necessitating different code to be able to support the same operation. Configuration management is the particular process by which many of us control changes during advancement and maintenance, and it also presents several advantages in protection. In particular, configuration managing scrutinizes new and transformed code to ensure, amongst other things, that safety flaws have not recently been inserted, intentionally or inadvertently.

Four activities are included in configuration management:

- construction identification
- configuration control and even change management
- configuration auditing
- status accounting

Configuration identity sets up baselines that all other code is going to be compared after shifts are made. That may be, we all build and document a great inventory of all elements that comprise the program. The inventory includes certainly not only the code a person and your colleagues may well have created, but furthermore database management systems, third party software, libraries, test instances, documents, and more. After that, we "freeze" the base and carefully control exactly what happens to it. Any time a change is suggested and made, it will be described when it comes to how the particular baseline changes.

Configuration handle and configuration management guarantee we can coordinate split, related versions. For instance, there might be closely related types of the system to implement on 16-bit and 32-bit processors. Three ways to be able to control the changes usually are separate files, deltas, and even conditional compilation. If we all use separate files, we all have different files intended for each release or variation. For example, we may possibly build an encryption program in two configurations: 1 that utilizes a short key element length, to comply using the law in selected countries, and another that will run on the long key. And then, version 1 may always be composed of components A2 through Ak and B1, while version 2 will be A1 through Ak in addition to B2, where B1 and even B2 do key duration. That is, the types are the same besides for the separate key element processing files.

Alternatively, all of us can designate a certain version as the key version of the system and even then define other types in terms of just what is different. The variation file, called a delta, contains editing commands to explain the ways to change the main version straight into the variation.

Lastly, we can do conditional compilation, whereby a solitary code component addresses just about all versions, counting on the compiler to determine which transactions to apply to which often versions. This approach appears appealing for security software because every one of the code seems in one place. Nevertheless, if the variations are incredibly complex, the code is quite difficult to read in addition to understand.

Once a construction management strategy is selected and applied, the program ought to be audited regularly. The configuration audit confirms how the baseline is complete and even accurate, that changes happen to be recorded, that recorded adjustments are made, and of which the actual software (that is, the software because used in the field) is reflected accurately inside the documents. Audits happen to be usually done by impartial parties taking one involving two approaches: reviewing just about every entry inside the baseline in addition to comparing it with typically the software in use or perhaps sampling from a bigger set just to verify compliance. For systems together with strict security constraints, the particular first approach is more suitable, but the second method may be more useful.

Finally, status accounting documents advice about the elements: where they originated in (for instance, purchased, reused, or even written from scratch), typically the current version, the switch history, and pending transformation requests.

All four pieces of activities are executed by way of a configuration and transformation control board, or CCB. The CCB contains reps from all organizations along with a vested fascination with typically the system, perhaps including clients, users, and developers. The particular board reviews all offered changes and approves adjustments based on need, design and style integrity, future plans with regard to the software, cost, and even more. The developers putting into action and testing the modification work with a system librarian to control plus update relevant documents in addition to components; they also publish detailed documentation about typically the changes and test effects.

Configuration management offers 2 advantages to those associated with us with security worries: protecting against unintentional risks and guarding against harmful ones. Both goals are usually

addressed when the construction management processes protect the particular integrity of programs plus documentation. Because changes take place only after explicit endorsement from the configuration management expert, all changes are in addition carefully evaluated for part effects. With configuration supervision, previous versions of courses are archived, so a new developer can retract a new faulty change when this is necessary.

Malicious adjustment is made very tough with a strong overview and configuration management method set up. In fact, , poor configuration control offers resulted in no less than one method failure; that sidebar furthermore confirms the principle regarding easiest penetration from Phase 1. Once an examined program is accepted with regard to inclusion in a program, the developer cannot go in to make smaller, and subtle changes, many of these as inserting trapdoors. Typically the developer has access to be able to the running production plan only through the CCB, whose members are conscious of such security removes.

Standards of Program Development

No software development firm worth its salt enables its developers to create code without notice in any kind of manner. The good software program development practices described previously in this chapter have got all been validated by simply many years of training. Although none is Brooks's mythical "silver bullet" that will guarantees program correctness, good quality, or security, they most add demonstrably to typically the strength of programs. Therefore, organizations prudently establish specifications for how programs are usually developed. Even advocates associated with agile methods, which provide developers a peculiar degree regarding flexibility and autonomy, inspire goal-directed behavior based upon earlier experience and past good results. Standards and guidelines can easily capture wisdom from past projects and boost the probability that the resulting method will be correct. Throughout addition, you want to ensure that will the systems we construct are reasonably simple to preserve and are compatible along with the systems with which in turn they interact.

We could exercise some degree associated with administrative control over application development by considering various kinds of standards or perhaps guidelines:

- **standards of style**, including using specified style tools, languages, or techniques, using design diversity, plus devising strategies for mistake handling and fault threshold
- **standards of documentation**, terminology, and coding style, which include the layout of code for the page, choices of titles of variables, and work with of recognized program set ups
- **standards of programming**, like mandatory peer reviews, intermittent code audits for correctness, and compliance with criteria
- **standards of testing**, this sort of as using program confirmation techniques, archiving test benefits for future reference, making use of independent testers, evaluating analyze thoroughness, and encouraging test out diversity
- **standards of setup management**, to control entry to and changes involving stable or completed system units

Standardization improves the particular conditions under which just about all developers work by building a common framework in order that no one developer is usually indispensable. It also enables carryover from a single project in order to another; lessons learned about previous projects available regarding use by all within the next project. Standards, in addition, assist in maintenance, given that the maintenance team can easily find required information inside a well-organized program. Yet, we must take treatment that the standards perform not unnecessarily constrain the particular developers.

Firms concerned regarding security and committed in order to follow software development specifications often perform security audits. In a security review, an independent security analysis team arrives unannounced to check on each project's compliance together with standards and guidelines. These people reviews requirements, designs, records, test data and ideas, and code. Knowing of which documents are routinely looked at, a developer is less likely to set suspicious code inside a component in typically the first place.

Process Standards

You have a couple of friends. Sonya is really well organized, she maintains lists of things you can do, the girl always knows how to find some sort of tool or who provides a specific book, and every thing is completed before it is usually needed. Dorrie, on the particular other hand, is a new mess. She can by no means find her algebra reserve, her desk has consequently many piles of paperwork you cannot see the particular top, and he or perhaps she seems to package with everything as being a problems because she is likely to overlook things until the last second. Who would you select to organize and operate a major social functionality, a new product release, or even a multiple-author paper? Many people would pick Sonya, concluding that her corporation skills are very essential. There is no promise that Sonya would carry out a better job as compared to Dorrie, but you may possibly assume the chances happen to be better with Sonya.

All of us know that software growth is difficult in portion since it has inherently individuals aspects that are quite difficult to judge beforehand. Still, we may consider that software built within an orderly manner provides a better potential for becoming good or secure.

The application Engineering Institute developed typically the ability Maturity Model (CMM) to evaluate organizations, not necessarily products. The International Criteria Organization (ISO) developed method standard ISO 9001, that is somewhat identical to the CMM. Lastly the U. S. Country wide Security Agency (NSA) designed the System Security Design CMM observe. Almost all of these are procedure models, in that they will examine how an firm does something, not precisely what it does. Thus, these people judge consistency, and a lot of folks extend consistency to top quality. For views on that will subject, see Bollinger and even McGowan and Curtis. El Emam has also viewed at the reliability associated with measuring a procedure.

Right now go back to typically the original descriptions of Sonya and Dorrie. Who might make the better creator? That question is complicated because many of people have friends like Dorrie who are fabulous coders, but we may furthermore know great programmers that resemble Sonya. And several successful teams have each. Order, structure, and regularity can lead to good software jobs, nonetheless it is not sure to be able to be the only approach to go.

3.6 Review Question

1. Suppose you are a customs inspector. You are responsible for checking suitcases for secret compartments in which bulky items such as jewelry might be hidden. Describe the procedure you would follow to check for these compartments.
2. Your boss hands you a microprocessor and its technical reference manual. You are asked to check for undocumented features of the processor. Because of the number of possibilities, you cannot test every operation code with every combination of operands. Outline the strategy you would use to identify and characterize unpublicized operations.
3. Your boss hands you a computer program and its technical reference manual. You are asked to check for undocumented features of the program. How is this activity similar to the task of the previous exercises? How does it differ? Which is the most feasible? Why?
4. Could a computer program be used to automate testing for trapdoors? That is, could you design a computer program that, given the source or object version of another program and a suitable description, would reply Yes or No to show whether the program had any trapdoors? Explain your answer.
5. A program is written to compute the sum of the integers from 1 to 10. The programmer, well trained in reusability and maintainability, writes the program so that it computes the sum of the numbers from k to n . However, a team of security specialists scrutinizes the code. The team certifies that this program properly sets k to 1 and n to 10; therefore, the program is certified as being properly restricted in that it always operates on precisely the range 1 to 10. List different ways that this program can be sabotaged so that during execution it computes a different sum, such as 3 to 20.
6. One means of limiting the effect of an untrusted program is confinement: controlling what processes have access to the untrusted program and what access the program has to other processes and data. Explain how confinement would apply to the earlier example of the program that computes the sum of the integers 1 to 10.
7. List three controls that could be applied to detect or prevent salami attacks.
8. The distinction between a covert storage channel and a covert timing channel is not clear-cut. Every timing channel can be transformed into an equivalent storage channel. Explain how this transformation could be done.
9. List the limitations on the amount of information leaked per second through a covert channel in a multiaccess computing system.
10. An electronic mail system could be used to leak information. First, explain how the leakage could occur. Then, identify controls that could be applied to detect or prevent the leakage.

11. Modularity can have a negative as well as a positive effect. A program that is overmodularized performs its operations in very small modules, so a reader has trouble acquiring an overall perspective on what the system is trying to do. That is, although it may be easy to determine what individual modules do and what small groups of modules do, it is not easy to understand what they do in their entirety as a system. Suggest an approach that can be used during program development to provide this perspective.

12. You are given a program that purportedly manages a list of items through hash coding. The program is supposed to return the location of an item if the item is present or to return the location where the item should be inserted if the item is not in the list. Accompanying the program is a manual describing parameters such as the expected format of items in the table, the table size, and the specific calling sequence. You have only the object code of this program, not the source code. List the cases you would apply to test the correctness of the program's function.

13. You are writing a procedure to add a node to a doubly linked list. The system on which this procedure is to be run is subject to periodic hardware failures. The list your program is to maintain is of great importance. Your program must ensure the integrity of the list, even if the machine fails in the middle of executing your procedure. Supply the individual statements you would use in your procedure to update the list. (Your list should be fewer than a dozen statements long.) Explain the effect of a machine failure after each instruction. Describe how you would revise this procedure so that it would restore the integrity of the basic list after a machine failure.

14. Explain how information in an access log could be used to identify the true identity of an impostor who has acquired unauthorized access to a computing system. Describe several different pieces of information in the log that could be combined to identify the impostor.

15. Several proposals have been made for a processor that could decrypt encrypted data and machine instructions and then execute the instructions on the data. The processor would then encrypt the results. How would such a processor be useful? What are the design requirements for such a processor?

1.7 References

1. Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger - RAND Corporation Publisher: Prentice Hall
2. Cryptography and Network Security - Principles and Practice fifth edition Stallings William Publisher: Pearson
3. Cryptography And Network Security 3rd Edition behrouz a forouzan and debdeep mukhopadhyay 3/E Publisher: McGraw Hill Education
4. Cryptography and Network Security, 3e Atul Kahate Publisher: McGraw Hill

Chapter 4. Protection in General-Purpose Operating Systems

4.0 Protection in General-Purpose Operating Systems

4.1. Protected Objects and Methods of Protection

4.2. Memory and Address Protection

4.3. Control of Access to General Objects

4.4. File Protection Mechanisms

4.5. User Authentication

4.6 Review Question

4.7 References

4.0 Protection in General-Purpose Operating Systems

An Operating System provides a couple of goals: managing shared access and implementing an interface to allow that access. Underneath those goals will be assist actions, incorporating identification and authentication, naming, filing objects, scheduling, communication among the processes, and reclaiming and reusing objects. Operating system characteristics consists of

- access control
- identity and credential management
- information flow
- audit as well as integrity protection

each one of these activities offers security ramifications. Operating systems vary from basic types assisting an individual job at any given time .this kind of operating system may operate a personal digital assistant to complex multiuser, multitasking systems, and, obviously, security factors help to increase when operating systems become more and more complicated.

We begin by studying the contributions of operating systems have made to positively user security. An operating system supports multiprogramming that is, the concurrent use of a system by more than one user, so operating system designers have developed ways to protect one user's computation from inadvertent or malicious interference by another user. Among all those features provided for this goal are memory protection, file protection, general control of usage of objects as well as user authentication. This chapter studies finally, the controls that provide these types of four features. We have oriented this discussion to the user: How do the controls safeguard users, and how can end users incorporate all those controls?

4.1. Protected Objects and Methods of Protection

We begin by reviewing the history of protection in operating systems. This background helps us understand what kinds of things operating systems can protect and what methods are for sale to protecting them

A Bit of History

Previously, there were no os's: Users entered all their applications directly into the device in binary by way of switches. Most of the time, program entry was made by physical manipulation of the toggle switch; in various other instances, the entry was worked with a more complicated electronic method, by using an input device, for instance, a keyboard. Because each individual had exclusive usage of the computing program, users were needed to arrange blocks of time intended for running the device. These users were accountable for loading their personal libraries

of support exercise routines assemblers, compilers, distributed subprograms and "clearing up" after making use of by removing any susceptible code or data.

The first os's were simple utilities, known as executives, designed to help individual programmers and also to smooth the transition from user to another. Early executives supplied linkers and loaders intended for relocation, comfortable access to compilers and assemblers, and then automated loading of subprograms coming from libraries. The executives dealt with the tedious facets of programmer support, concentrating on one programmer during execution.

Operating systems had taken on a much wider role (and a distinct identity) as the concept of multiprogramming was first implemented. Seeing that two users can interleave use of the resources of an individual computing system, researchers designed concepts, for example, scheduling, posting, and parallel make use of. Multiprogrammed operating systems also called monitors, oversaw every program's execution. Monitors had taken an active role, while executives had been passive. That may be, an executive remained in the history, waiting for being called into provider by their requesting user. Yet a monitor definitely asserted control of the processing system and provided resources towards the user only once the request was according to general good utilization of the system. Likewise, the executive waited to get a request and provided provider on demand; the monitor has taken care of control over almost all resources, allowing or denying most computing and loaning assets to end users as they required them.

Multiprogramming brought another change to computing. Each time one person was utilizing a system, the just force to be guarded against was the end user himself or herself. A person making a mistake could have experienced silly, still, one user cannot impact the computation of any kind of another user negatively. However, multiple users presented more difficulty and risk considerably. User A could properly become upset if Consumer B's applications or data developed a negative effect on A's program's functionality. Therefore, safeguarding one user's applications and data from various other users' programs are becoming an important issue in multiprogrammed os's.

4.1. Protected Objects and Methods of Protection

We begin by reviewing the history of protection in operating systems. This background helps us understand what kinds of things operating systems can protect and what methods are available for protecting them.

A Bit of History

Once upon a time, there were no operating systems: Users entered their programs directly into the machine in binary by means of switches. In many cases, program entry was done by physical

manipulation of a toggle switch; in other cases, the entry was performed with a more complex electronic method, by means of an input device such as a keyboard. Because each user had exclusive use of the computing system, users were required to schedule blocks of time for running the machine. These users were responsible for loading their own libraries of support routine assembles, compilers, shared subprograms and "cleaning up" after use by removing any sensitive code or data.

The first operating systems were simple utilities, called **executives**, designed to assist individual programmers and to smooth the transition from one user to another. The early executives provided linkers and loaders for relocation, easy access to compilers and assemblers, and automatic loading of subprograms from libraries. The executives handled the tedious aspects of programmer support, focusing on a single programmer during execution.

Operating systems took on a much broader role (and a different name) as the notion of multiprogramming was implemented. Realizing that two users could interleave access to the resources of a single computing system, researchers developed concepts such as scheduling, sharing, and parallel use. **Multiprogrammed operating systems**, also known as **monitors**, oversaw each program's execution. Monitors took an active role, whereas executives were passive. That is, an executive stayed in the background, waiting to be called into service by a requesting user. But a monitor actively asserted control of the computing system and gave resources to the user only when the request was consistent with general good use of the system. Similarly, the executive waited for a request and provided service on demand; the monitor maintained control over all resources, permitting or denying all computing and loaning resources to users as they needed them.

Multiprogramming brought another important change to computing. When a single person was using a system, the only force to be protected against was the user himself or herself. A user making an error may have felt foolish, but one user could not adversely affect the computation of any other user. However, multiple users introduced more complexity and risk. User A might rightly be angry if User B's programs or data had a negative effect on A's program's execution. Thus, protecting one user's programs and data from other users' programs became an important issue in multiprogrammed operating systems.

Protected Objects

- In fact, the surge of multiprogramming led to many areas of a computing system needed protection:
- memory space
- sharable I/O equipment, for instance, disk drives
- reusable I/O equipment serially, for instance, computer printers and tape drives
- sharable subprocedures and applications
- networks
- sharable data

Because it presumed task designed for handled posting, the operating system had a need to safeguard such objects. In the next sections, we look at some of the mechanisms with which os's have forced these varieties of objects' protection. Various operating-system safeguard mechanisms have already been maintained hardware

Security Methods of Operating Systems

The foundation of protection is generally separation: staying one user's objects distinguish from all other end users. Rushby and as well, Randell mentioned that parting within an operating-system can take place in numerous ways:

- **physical separation**, wherein unique processes make use of several physical objects, such as separate printers meant for output needing several examples of protection
- **temporal separation**, wherein processes going through several security criteria will be performed at different times
- **logical separation**, wherein users work beneath the impression the fact that no additional processes are available, because when an operating-system constraint a program's accesses to the program is not able to gain access to objects outside their authorized domains
- **cryptographic separation**, wherein processes hide their data and computations in such a way that they can be unintelligible to help you outside processes

Obviously, mixtures of two or more of such kinds of separation can also be possible. The types of separation are outlined approximately in increasing order of difficulty to implement, and, for the first three, in lowering order of the security supplied. On the other hand, the first two methods are extremely rigid and may result in poor resource usage. As a result, we would like to transfer the duty of protection to the operating system to permit concurrent execution of processes having to cope with varied security requirements.

Nevertheless, separation is merely half the answer. We would like to different end users and their objects, but we also want to manage to offer to share for some of these objects. For example, two users with different protection, levels might want to employ precisely the same search algorithm or function call.

All of us wants users to be able to share the features and algorithms without diminishing their unique specific security needs. An operating-system support parting and sharing in various techniques, providing safeguard any type or form of time in many levels.

- **Do not protect.** Os's without security work when delicate methods are receiving operate in different situations.
- **Isolate.** When an operating-system gives isolation, several procedures operating aren't aware of the existence of one another simultaneously. Every procedure possesses its address space, files, and other objects also. The operating system has to restrict as a result every process for some reason the objects of various other processes will be obscured totally.

- **Share all or talk about almost nothing.** Due to this sort of security, who owns an object promises this to be personal or public. An open public object exists to everyone users, while a private object exists with their owner simply.
- **Share through access restriction.** Because of protection by gain access to constraint simply, the operating system inspections the allowability of each user's potential use of an object. That's, access control usually is applied for a specific eliminate consumer including a specific object. Prospect lists of suitable actions guide the operating-system in deciding in the event that the specified user will need to have use of a particular object. In a few sense, the operating system offers a shield among items and users, ensuring authorized has usage of happening exclusively.
- **Share by features.** An extension of little access sharing, this kind of safeguard enables powerful developing of sharing privileges designed for objects. The known degree of sharing depends upon this owner or the topic may be, within the framework of the computation, or within the thing itself.
- **Limit usage of an object.** This kind of protection limitations not merely the access an object but also the utilize made of that object after it can often be accessed immediately. For example, a user may be given permission to be to see an extremely sensitive record, however, never to print a duplicate of it. More strongly even, a user could be given permission to end up being entry to data inside a database to assist you to get statistical summaries (for instance, average income at a particular grade level), however, never to ascertain particular data values (salaries of people).

4.2. Memory and Address Protection

The obvious issue in multiprogramming is generally protecting against one program from having an effect on the data and programs within the memory space of other users. However, protection could be constructed into the hardware mechanisms which usually restrain effective utilization of memory space, therefore solid protection could be offered at effectively no extra cost.

Fence

The easiest type of memory space protection was presented in single-user operating systems to avoid a defective user program from doing damage to the area of the resident section of the operating system. As the name indicates, a fence can be described as a strategy to restrict users to one side of the boundary.

In a single implementation, the fence must have been a predetermined memory space address, allowing the operating system to reside in on a single side as well as the user to remain on the other. An illustration of this case is shown in Figure 4.1. However, this type of implementation was extremely limited just because a predetermined magnitude of space was in fact usually

available to the operating system, regardless of it had been required or not. If lower than the predetermined space was needed, the extra space was misused. On the other hand, if the operating system required extra space, it might not really expand beyond the fence boundary.

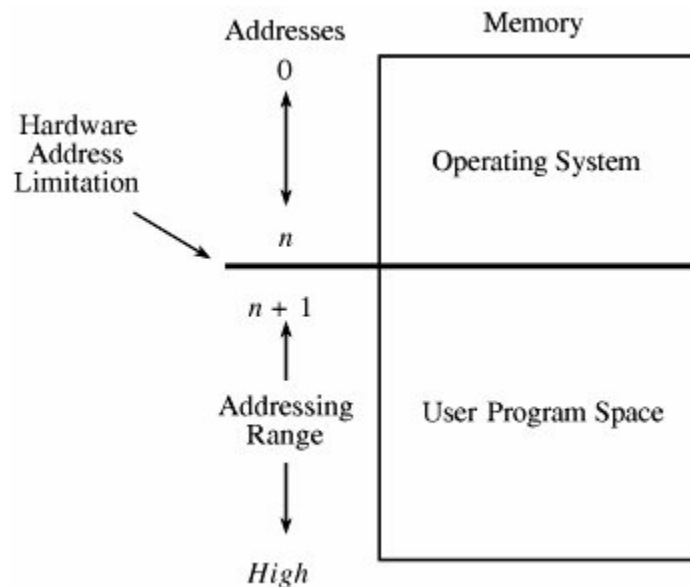


Figure 4.1. Fixed Fence

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

An additional implementation utilized a hardware register, known as a fence register, including the address of the end of the operating system. Contrary to a fixed fence, from this scheme, the positioning of the fence could possibly be modified. Every time a user system produced an address intended for data alteration, the address was indeed immediately compared to the fence address. If the address was more than the fence address (that could be, within the user area), the instruction was performed; if it was, in fact, lower than the fence address (that could be, inside operating system place), the fault condition grew up. The utilization of fence registers is demonstrated in Figure 4.2.

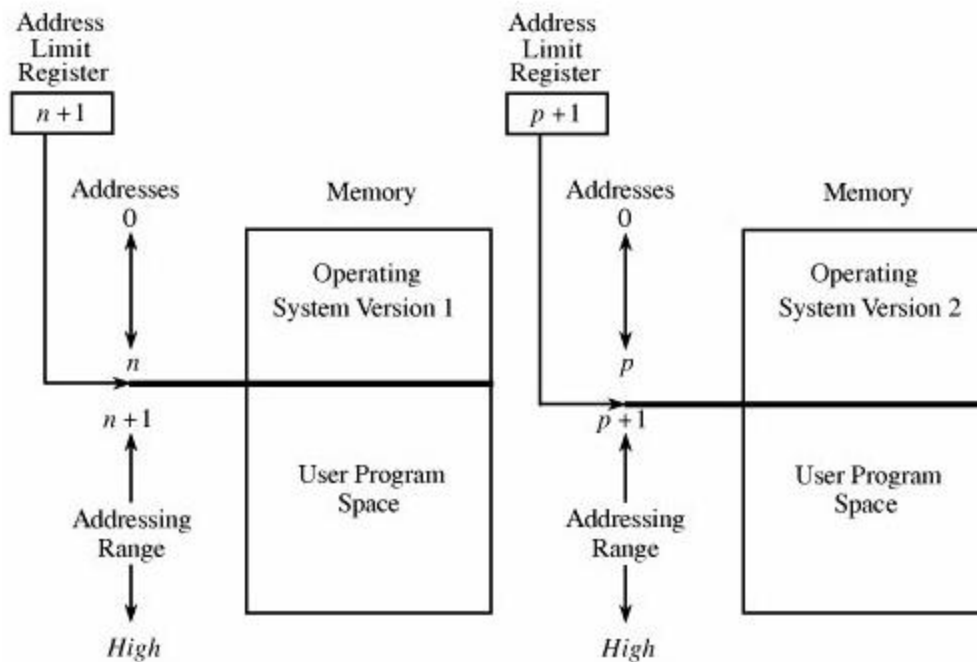


Figure 4.2. Variable Fence Register.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

A fence register shields just in a single direction. To put it differently, an operating system could be safeguarded from an individual user, however the fence are not able to shield a single user from an additional user. In the same way, an user are not able to determine specific regions of this program as invulnerable (such as code in the program itself or maybe a read-only data area).

Relocation

In the event the operating system could be believed for being of the predetermined size, programmers may create their particular code believing that the program will begin at a constant address. This kind of feature of the operating system makes it simple to look for the address of any kind of object inside the program. On the other hand, it also makes the idea effectively difficult to improve the starting address if, as an illustration, a fresh version of the operating system is usually larger or maybe smaller than the actual. Generally, if the size of the operating system can be permitted to modify, afterward programs should be developed in a fashion that will not be based upon positioning at any particular area in memory space. Relocation is a procedure for having a program developed just as if it initiated at address 0 and then converting all addresses which will show the actual particular address from which the program is positioned in memory. In many cases, this kind of work simply requires putting in a consistent relocation factor with each address in the program. That is certainly, the relocation

factor is the beginning address of the memory space allocated for the program. Ideally, the fence register can be utilized from this situation to provide you an essential excessive advantage: The fence register could be a hardware relocation device. The details of the fence register will be placed into every program address. This course of action equally relocates the address and assurances that nobody can gain access to an area less than the fence address. (Addresses will be considered as unsigned integers, therefore conjoining the value in the fence register to every number is normally bound to build a result at or maybe over a fence address.) Special instructions could be added intended for a couple of occasions when a program legally expects to gain access to an area from the operating system.

Base/Bounds Register

A significant advantage of an operating program because of fence registers is a capacity to relocate; this sort of feature is vital in a multiuser environment particularly. Because of many customers, none of them can certainly understand in advance in which a program is going to be loaded designed for execution. The relocation register resolves nevertheless, the nagging problem giving a base or starting address. All the addresses within a scheduled plan will be offsets out of this base address. A variable fence register is normally termed as a base register. A variable fence register is normally described as a base register.

Fence registers provide a lower bound (a beginning address) however, no upper one. A higher bound can be helpful in understanding how very much space is normally allocated furthermore to examining for overflows into "forbidden" areas. To get over this sort of problems, To, another register is added, as proven in Figure 4-3. The next register, referred to as bounds register, is definitely a higher address limit, very much the same that the fence or base sign-up is actually a lower address limit. Every plan address will end up being over a bottom address because the contents of the bottom register will be put into the address; every address is additionally examined to ensure that it's beneath the bounds address truly. This way, a program's addresses will end up being properly limited to the region between the base and also the bounds registers.

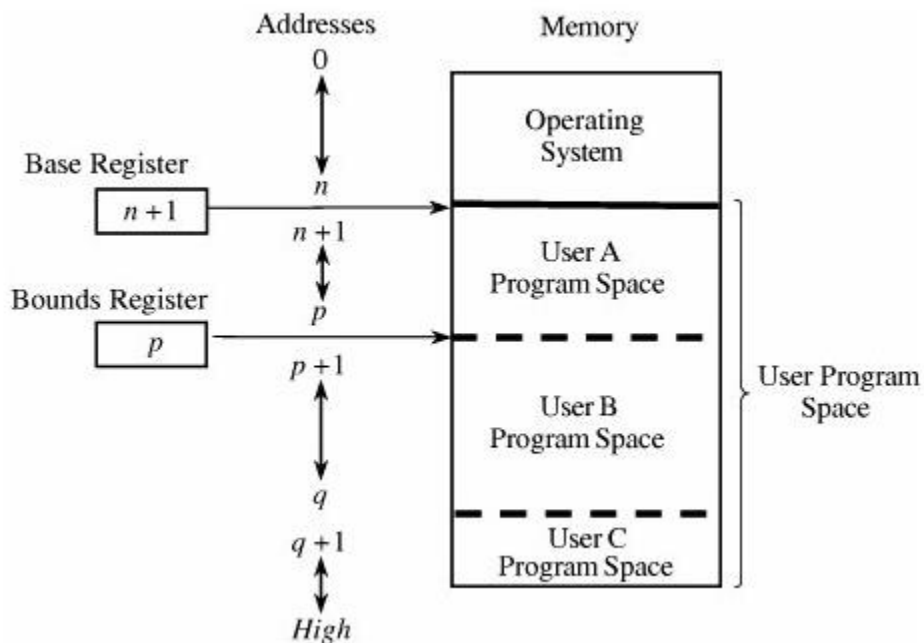


Figure 4-3. Pair of Base/Bounds Registers.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

This method shields a program's addresses from changes simply by another end user. Once execution adjustments from a single user's program to another's, the operating system need to replace the contents of the base and bounds registers to reveal the actual address space for the user. This kind of change is usually section of the basic planning, This kind of change is section of the basic planning usually, known as context switch, the fact that operating system needs to execute once shifting control from one user to another.

Using a couple of base/bounds registers, users are usually effectively guarded right from outside users, users are effectively guarded right from outside users usually, or perhaps, even more properly, outside users will be guarded right from errors in different various user's program. Incorrect addresses within a user's address space could impact that may program since the base/bounds looking at assurances just that every address is usually within the user's address space. To illustrate, users mistake may happen each time a subscript beyond the range or maybe an undefined variable produces an address reference inside the user's space although, however, within the executable instructions of the user's program. In this way, a user may unintentionally store data on the top of instructions. This kind of error allows a user unintentionally damage an application, yet (luckily) only the user's own program.

We are able to resolve this kind of overwriting issue by utilizing an additional set of base/bounds registers, one particular intended for the instructions (code) from the program another for the data space. In that case, only instruction fetches (instructions for being executed) will be relocated as well as, examined along with the first register pair, and after that data accesses (operands of instructions) will be relocated and examined along with the second register pair. The usage of two pairs of base/bounds registers is shown in Figure 4- 4. Even though two pairs of registers will not protect against all program errors, they will limit the effects of data-manipulating instructions towards the data space. The pairs of registers provide one more crucial advantage: the capability to divided a program into two pieces which can be relocated individually.

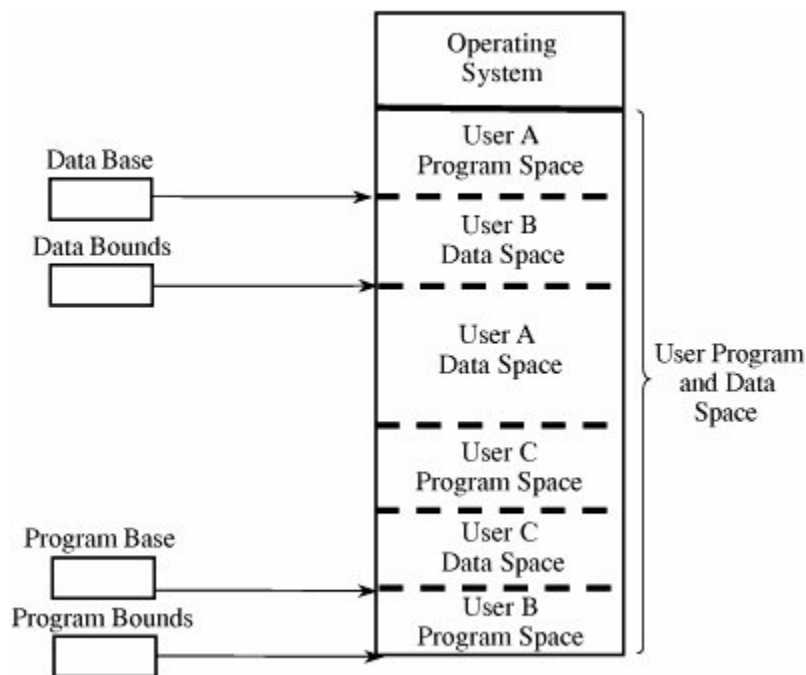


Figure 4-4. Two Pairs of Base/Bounds Registers.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Tagged Architecture







An additional issue with using base/bounds registers to get relocation or protection is normally their particular contiguous characteristics. Every group of registers limitations has usage of to a successive collection of addresses. A compiler or loader can merely piece together an application to ensure that all code areas will end up being adjacent and all data areas will end up being adjacent.

However, in some instances, you might want to shield a couple of data values however, not virtually all. To illustrate, a workers record may need shielding this field designed for income however, not office get in touch with and location number. Furthermore, a programmer may want to ensure the ethics of particular data ideals by letting them finish up being made if this program is normally initialized yet forbidding this program from changing them afterward. This types of program shields from errors in the programmer's very own code. A programmer could also make use of a distributed subprogram from the common library. We are able to address a few of these nagging problems by making usage of good style, both in the operating system and in the various other programs keeping operate. Recall that in Chapter 3 most of us learned great design features, for instance, details modularity and hiding in plan design. These varieties of characteristics determine that one program component have to tell another component the real minimal capability of data needed for both of them to execute their job.

Further, operating-system-specific style features shall help, too. Bottom/bounds registers generate an all-or-nothing scenario designed for sharing: Whether plan makes all its data open to become utilized and changed or it forbids access all. Whether there were a than band of registers designed for distributed data, all data will together have to be placed. A procedure could hardly share data items a, B, and C with one element, A, C, and D with another, and A, B, and D with a third. In order to to achieve the kind of sharing we desire is to maneuver every ideal band of data values for some contiguous space. However, this type or types of treatment might not be appropriate if the data items had been large records, arrays, or structures.

An alternate is tagged architecture, by which each and every word of machine storage offers several excessive bits to identify the access privileges compared to that word. These varieties of entryway bits could possibly be arranged by just privileged (operating-system) instructions. The bits will be tested each right time an instruction has usage of that location.

To illustrate, as proven in Figure 4.5, one storage area could be guarded seeing that execute-just (to illustrate, the thing code of guidelines), although another is normally guarded designed for fetch-only (one of these is, read) data gain access to, and another attainable for adjustments (to illustrate, write). This way, two adjacent places might have different access privileges. Second of all, by extra tag bits, different classes of data (numeric, character, pointer or address, and undefined) could possibly be separated, and data areas could be covered for privileged (operating-system) access only.

Tag	Memory Word
R	0001
RW	0137
R	0099
X	
X	
X	
X	
X	
X	
R	4091
RW	0002

Code: R = Read-only RW = Read/Write
X = Execute-only

Figure 4.5. Example of Tagged Architecture.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

This type or sort of safety technique has been applied to a few devices, though the level of tag bits have already been somewhat little also. The Burroughs B6500-7500 program used three tag bits to split data phrases (three types), descriptors (ideas), and control phrases (stack ideas and addressing control phrases). The IBM System/38 utilized a tag to control both access and integrity.

An alternative utilized one tag that positioned on a combined group of successive locations, such as for example 128 or 256 bytes. Because of one tag for the block of addresses, additional expense for applying tags was not as high much like the main one tag per area. The Intel I960 prolonged architecture processor chip used a tagged architecture using a little on each memory phrase which often designated the word such as a "capability," rather than as a typical location for guidelines or data. A capability managed the use of a variable-sized memory segment or block. This kind of many feasible tag ideals backed storage sections that ranged in proportions from 64 to 4 billion bytes, having a promising 2256 different security domains.

Compatibility condition of code proven problems together with the acceptance of a tagged architecture. A tagged architecture might not be as useful as more modern methods, as we shortly see. Some of the important pc vendors continue being working with operating systems which have been designed and applied in the past designed for architectures of this period. Certainly, most producers are locked towards an extra standard memory architecture because of this of large option of elements including a choose to preserve compatibility among os's and machine families. A tagged architecture may need important changes to consider all the operating system code, a necessity which can be really costly. But since the price of memory is constantly on the fall, the implementation of a tagged architecture turns into even more possible

Segmentation

We present a few additional methods to protection, every of which could be applied on the top of a standard machine framework, recommending a much better possibility of approval. Even though these types of methods are actually traditional by simply computing's criteria these were built somewhere between 1965 and 1975 they've been applied on various devices since that time. Secondly, they provide essential strengths in dealing with, with memory protection as being a wonderful reward.

The most important of those couple methods, segmentation, entails the easy idea of separating a program in to distinguish parts. Each part includes a logical unity, demonstrating a association of all of code or data values. By way of example, a segment could be the code of a single method, the data associated with an array, or the number of all local data values utilised by a particular component. Segmentation was created as being a feasible way to create the result of the counterpart of the unbounded number of base/bounds registers. To put it differently, segmentation enables a program to be split up into various parts needing diverse access privileges.

Each area incorporates a particular name. A code or data items inside section is tended to as the pair <name, offset>, where name is the name of the fragment containing the information thing and balance is its area inside the portion (that is, its separation from the beginning of the fragment).

Intelligently, the software engineer pictures a program as a long gathering of fragments. Sections can be independently migrated, enabling any portion to be put in any accessible memory areas. The connection between a coherent portion and its actual memory position is appeared in Figure 4.6.

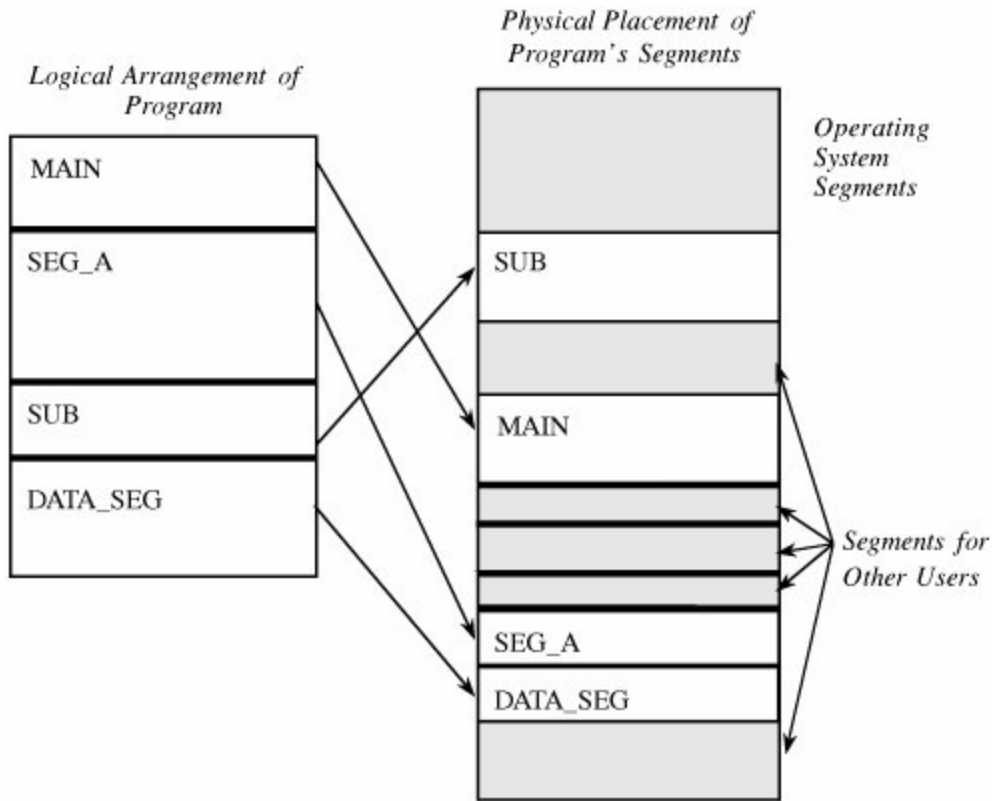


Figure 4.6. Logical and Physical Representation of Segments.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Consequently, a User's program does not realize what genuine memory tends to its employment. It has no way and no need to decide the accurate location related with a specific <name, offset>. The <name, offset> pair is satisfactory to get to any data or instruction to which a program ought to approach.

This concealing up of addresses has three positive aspects of working for the operating system.

the operating system can put any location at any area or move any location to any area, even after the program starts to execute. Since it converts all address references by a section address table, the operating system needs possibly update the location in that one table when a portion is moved.

A location can be expelled from primary memory (and put away on a helper gadget) on the off chance that it isn't being utilized presently.

Each location reference goes through the operating system, so there is a chance to check everyone for security.

Because of this last function, a process can get access to a phase simplest if that process seems in that process's phase translation table. The operating system controls which software have entries for a specific phase in their section deal with tables. This manage presents robust safety of segments from getting entry to with the aid of unpermitted tactics. For example, program A may have got right of entry to segments BLUE and GREEN of person X but now not too different segments of that user or of some other user. In a truthful way, we will permit a person to have distinctive protection instructions for one of a kind segments of an application. For instance, one phase might be read-best statistics, a second might be execute-best code, and a 3rd might be writeable data. In a state of affairs like this one, segmentation can approximate the intention of separate safety of various pieces of an application, as outlined in the preceding section on tagged architecture.

- Segmentation offers those safety blessings:
- Each address with reference is checked for safety.
- Many different instructions of data objects may be assigned one of a kind degrees of protection.
- Two or extra persons can be shared get right of entry to to a segment, with probably different access rights.
- A person can not generate an address with or access to an unpermitted segment.

Paging

One opportunity to segmentation is paging. The program is divided into equal-sized portions referred to as pages, and memory is split into equal-sized gadgets known as page frames. (For implementation reasons, the page size is typically selected to be a strength of among 512 and 4096 bytes.) As with segmentation, every address in a paging scheme is a two-element item, this time which include <page, offset>.

Each cope with is again translated with the aid of a system similar to that of segmentation: The operating machine keeps a desk of consumer page numbers and their proper addresses in reminiscence. The page component of each <page, offset> reference is transformed to a web page body deal with with the aid of desk research; the offset portion is added to the web page body deal with to provide the real memory address of the object known as <page, offset>. This process is illustrated in Figure 4.8.

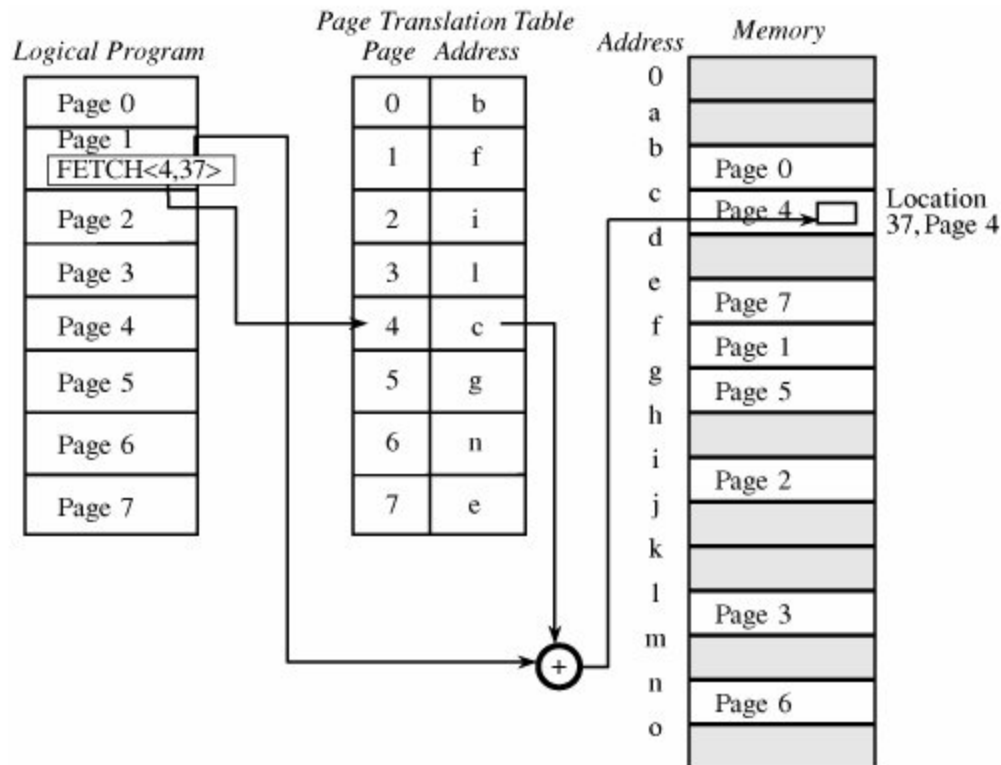


Figure 4-8. Page Address Translation.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

4.3. Control of Access to General Objects

Protecting memory is a selected case of the more trendy problem of protective objects. As multiprogramming has evolved, the numbers and sorts of objects shared have additionally expanded. Here are some examples of the types of items for which safety is appropriate:

- memory
- a file or data set on an auxiliary storage device
- an executing software in memory
- a directory of files
- a hardware device
- a data structure, which include a stack
- a table of the operating system
- commands, especially privileged instructions
- passwords and the person authentication mechanism

- the protection mechanism itself

The memory protection mechanism may be pretty simple due to the fact each memory access is guaranteed to undergo specific points in the hardware. With extra trendy objects, the number of factors of getting right of access to may be large, a central authority thru which all accesses pass may be missing, and the kind of get admission to may not clearly be restricted to read, write, or execute.

Furthermore, all accesses to memory occur thru an application, so we can discuss with the program or the programmer as the having access to the agent. In this chapter, we use phrases just like the user or the subject in describing get access to entry to general objects. This user or subject could be a person who uses a computing device, a programmer, a software, some other object, or something else that seeks to apply an object.

There are several complementary desires in defensive objects.

- **Check every access.** We may additionally need to revoke a consumer's privilege to get entry to an object. If we have formerly legal the user to get permission to access the object, we do no longer always intend that the user has to hold indefinite get right of access to to the object. In fact, in a few situations, we may additionally want to save you, in addition, get the right of access to immediately when we revoke authorization. For this reason, each gets entry to via a consumer to an object ought to be checked.
- **Enforce least privilege.** The principle of least privilege states that a topic needs to have access to the smallest wide variety of objects important to carry out a few jobs. Even if greater information could be vain or harmless if the user has been to have got access to objects, the users must not have that extra get right of access to objects. For instance, an application needs to not have access to absolutely the memory address to which a page number address is translated, even though this system could not use that address in any effective manner. Not permitting access to needless objects guards in opposition to safety weaknesses if part of the protection mechanism ought to fail.
- **Verify acceptable usage.** The ability to get entry to is a yes-or-no decision. But it is equally essential to check that the interest to be activity on an object is appropriate. For example, a data structure that includes a stack has particularly suitable operations, consisting of push, pop, clear, and so forth. We can also need no longer only to control who or what has access to a stack however additionally to be guaranteed that the accesses finished are valid stack accesses.

In the next section, we don't forget protection mechanisms suitable for standard objects of unspecified sorts of, such as the varieties of objects listed above. To make the explanations easier to understand, we occasionally use an example of a specific object, along with a document. Note, but, that a preferred mechanism can be used to protect any of the types of object indexed.

Directory

One simple way to shield an object is to use a mechanism that works like a file directory. Imagine we're trying to shield field (the set of objects) from users of a computing system (the set of objects). Every file has a completely unique owner who possesses "control" get admission to rights (along with the rights to claim who has what gets entry to the system) and to revoke access to any individual user at any time. Each user has a file directory, which lists all of the files to which that user has get right of entry to the system.

Clearly, no user can be allowed to write within the file directory because that might be a manner to forge get entry to file. Therefore, the operating system must hold all files directories, under command from the proprietors of files. The obvious rights access to files are commonly study, write, and execute acquainted on many shared systems. Furthermore, some other right access, the owner, is possessed by using the owner, allowing that person to provide and revoke access rights. Figure 4.10 shows an example of a file directory.

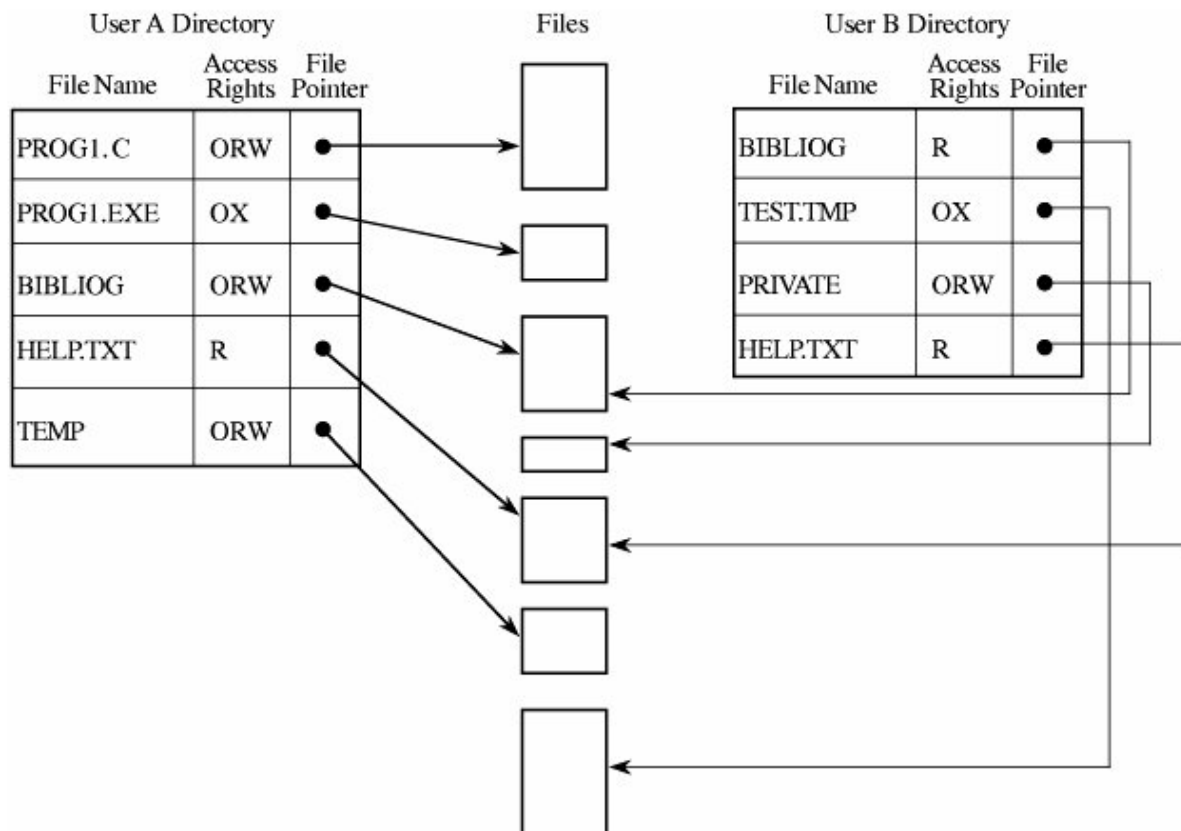


Figure 4.10. Directory Access.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

This method is straightforward to enforce as it uses one list per person, naming all of the objects that users are authorized to get an entry into the system. However, several difficulties can arise. First, the list will become too big if many shared objects, along with libraries of subprograms or a not unusual table of users, are accessible to all users. The directory of every user should have one entry for each such shared object, despite the fact that the person has no intention of accessing the object. Deletion must be pondered in all directories.

A second difficulty is revocation to getting entry to. If proprietor A offers exceeded to person B the proper to study document F, an access for F is made inside the directory for B. This granting of access implies a stage of consider among A and B. If A later questions that trust, A can also want to revoke the get right of entry to proper of B. The working device can reply without difficulty to the single request to delete the proper of B to get admission to F due to the fact that motion entails deleting one access from a specific listing. But if A wants to take away the rights of absolutely everyone to get admission to F, the operating system ought to search every man or woman directory for the entry F, a pastime that can be time-consuming on a massive machine. For instance, big timesharing systems or networks of smaller systems can effortlessly have 5,000 to 10,000 active debts. Moreover, B may also have passed the access right for F to another user, so A may not recognize that F's get entry to exists and should become revoked. This trouble is mainly serious in a network.

One-Third difficulty involves pseudonyms. Owners A and B would have two different data files called F, and they might both ought to allow access simply by S. Obviously, the directory for S could not contain two entries underneath the exact name for different data files. Therefore, S needs to be capable to uniquely determine the F for A (or B). One procedure is to are the first owner's designation as though it were section of the file name, with a notation such as A:F (or B:F).

Suppose, however, that S has trouble remembering file contents from the name F. Another approach is to allow S to name F with any name unique to the directory of S. Then, F from A could be called Q to S. As shown in Figure 4.11, S may have forgotten that Q is usually F from A, and so S requests access again from A for F. But by now A may have more trust in S, so A transfers F with greater rights than before. This action opens up the possibility that one subject, S, may have two distinct sets of access rights to F, one under the name Q and one under the name F. In this way, allowing pseudonyms leads to multiple permissions that are not necessarily consistent. Thus, the directory approach is most likely too simple for most object protection situations.

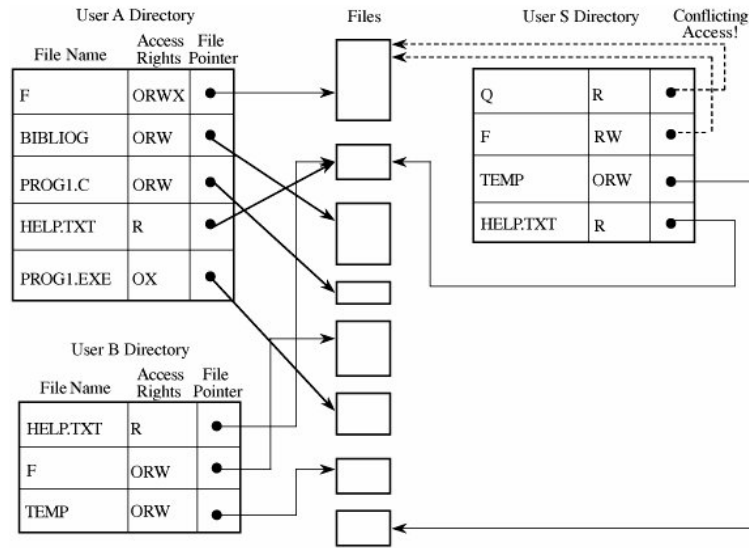


Figure 4.11. Alternative Access Paths.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Access Control List

An elective portrayal is the entrance control list. There is one such list for each instance or object, and the list demonstrates all subjects who ought to approach the instance or object and what their access rights are. This methodology varies from the directory list on the grounds that there is one access control list for every object; a directory is made for each subject. In spite of the fact that this distinction appears to be little, there are some huge focal points.

To perceive how, think about subjects A and S, both of whom approach object F. The operating system will keep up only one access list for F, demonstrating the access rights for A and S, as appeared in Figure 4.12. The access control list can incorporate general default sections for any user. Along these lines, unique clients can have unequivocal rights, and every other user can have a default set of rights. With this association, an open document or program can be shared by every conceivable user of the systems without the requirement for a passage for the object in the individual directory of the individual users.

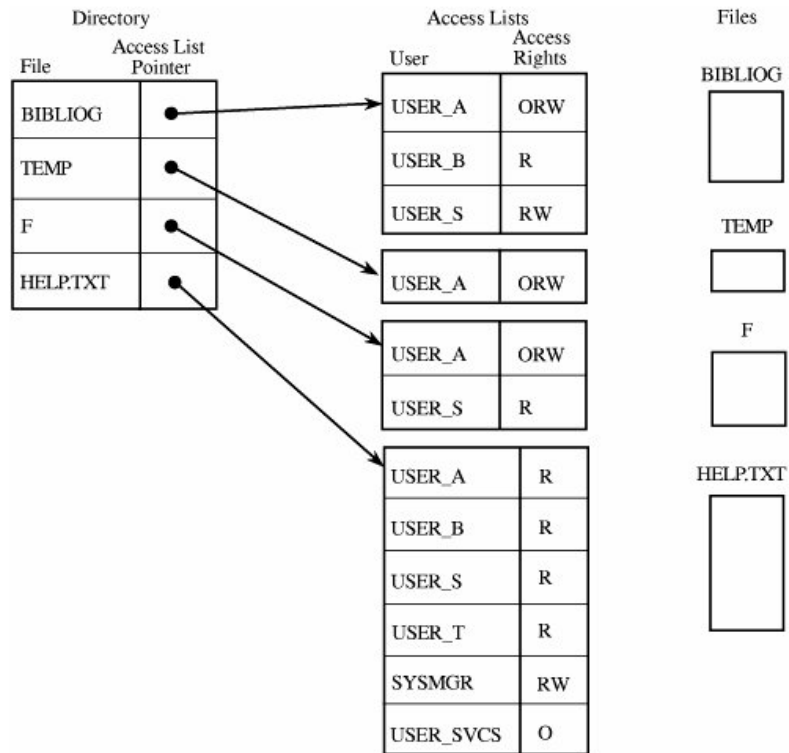


Figure 4-12. Access Control List.

Above Image taken from the book **Security in Computing, Fourth Edition** By **Charles P. Pfleeger** - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Access Control Matrix

We could think of the directory site like a listing of things accessible by the single theme, and the access checklist as a table determining subjects that can gain access to just one object. The information in these two illustrations are equivalent, the differentiation being the ease associated with use in given circumstances.

As an alternative, we are able to use an access management matrix, a table inside which each row signifies a subject, each steering column represents an object, and entry is the collection of access rights with regard to that subject to that will object. An example rendering of your access control matrix is shown in Stand 4.1. In general, typically the access control matrix is usually sparse (meaning that just about all cells are empty): Just about all subjects do not possess access rights to just about all objects. The access matrix can be represented since a list of triples, getting the form $\langle \text{subject, object, rights} \rangle$. Searching a large range of these triples is usually inefficient enough that this particular implementation is seldom applied.

	BIBLIOG	TEMP	F	HELP.TXT	C_COMP	LINKER	SYS_CLOCK	PRINTER
USER A	ORW	ORW	ORW	R	X	X	R	W
USER B	R	-	-	R	X	X	R	W
USER S	RW	-	R	R	X	X	R	W
USER T	-	-	-	R	X	X	R	W
SYS_MGR	-	-	-	RW	OX	OX	ORW	O
USER_SVCS	-	-	-	O	X	X	R	W

Table 4.1. Access Control Matrix.

Above Table taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Until now, we have examined security schemes in which typically the operating system must maintain program all the safety objects and rights. Yet other approaches put a few of the burden around the user. For example, the user may be expected to have a plane ticket or pass that allows access, much like some sort of ticket or identification credit card that should not be duplicated. Extra formally, we say that will a capability is definitely an unforgeable token that gives the particular possessor certain rights to an object. The Multics in addition to Hydra systems used capabilities intended for access control. Theoretically, a new subject can create innovative objects and can stipulate the operations allowed upon those objects. For illustration, users can create items, such as files, info segments, or subprocesses, and even can also specify typically the acceptable forms of operations, many of these as read, write, and even execute. But an customer can also create entirely new objects, such since new data structures, in addition to can define types regarding accesses previously unknown to be able to the system.

A functionality is a ticket providing permission to a controlled by include a certain type involving usage of an object. Intended for the capability to offer you solid protection, the admission must be unforgeable. 1 way to help it become unforgeable is to not provide the ticket directly to the particular user. Instead, the functioning system holds all seat tickets on behalf of the particular users. The operating technique returns to the customer a pointer to the os data structure, which in turn also links to typically the user. A capability can easily be created only simply by a specific request by your user to typically the operating system. Each functionality also identifies the permitted accesses.

Alternatively, capabilities may be encrypted under the key available only in order to the access control device. If the encrypted capacity contains the identity coming from the rightful owner, customer A cannot copy typically the capability and provide it in order to user B.

One achievable access directly to a great object is transfer or even propagate. A subject getting this right can go copies of capabilities to be able to other subjects. In change, each of these abilities also offers a list regarding permitted types of has access to, one of which may also be transfer. Within this instance, process A new can pass a duplicate of a power to B, that can then pass a new copy to C. N can prevent further submission of the capability (and therefore prevent further scattering in the access right) by simply omitting the transfer ideal from the rights handed in the capacity to G. B might still go certain access rights to be able to C, but not the particular justification to propagate gain access to rights to subjects.

Since a process executes, this operates in a domain name or local name room. The domain is the particular collection of objects to be able to which the process offers access. A domain with regard to an user in a presented time might include a few programs, files, data sectors, and I/O devices many of these as a printer and even a terminal. An illustration of a domain is definitely shown in Figure 4.13.

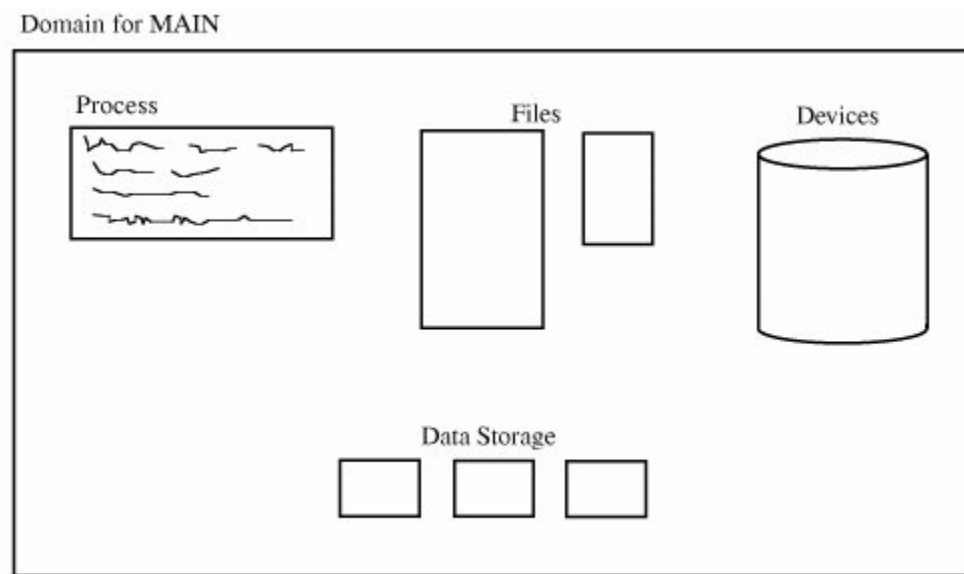


Figure 4-13. Process Execution Domain

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

As execution proceeds, the procedure may call a subprocedure, passing a portion of the object to which it approaches as arguments to the subprocedure. The domain of the subprocedure isn't really equivalent to that of its calling procedure; in fact , a calling technique may pass just a portion of its articles to the subprocedure, and the subprocedure may approach rights to different items not available to the calling strategy. The caller may likewise pass just a portion of its access

rights for the items it goes to the subprocedure. For instance, a method may go to a subprocedure the privilege to peruse however not alter specific information esteem.

Since every capacity distinguishes a single item in a domain , the accumulation of abilities characterizes the domain. At the point when a procedure calls a subprocedure and passes certain items to the subprocedure, the operating form a heap of the considerable number of abilities of the present system. The operating system at that point makes new abilities for the subprocedure, as appeared in Figure 4.14.

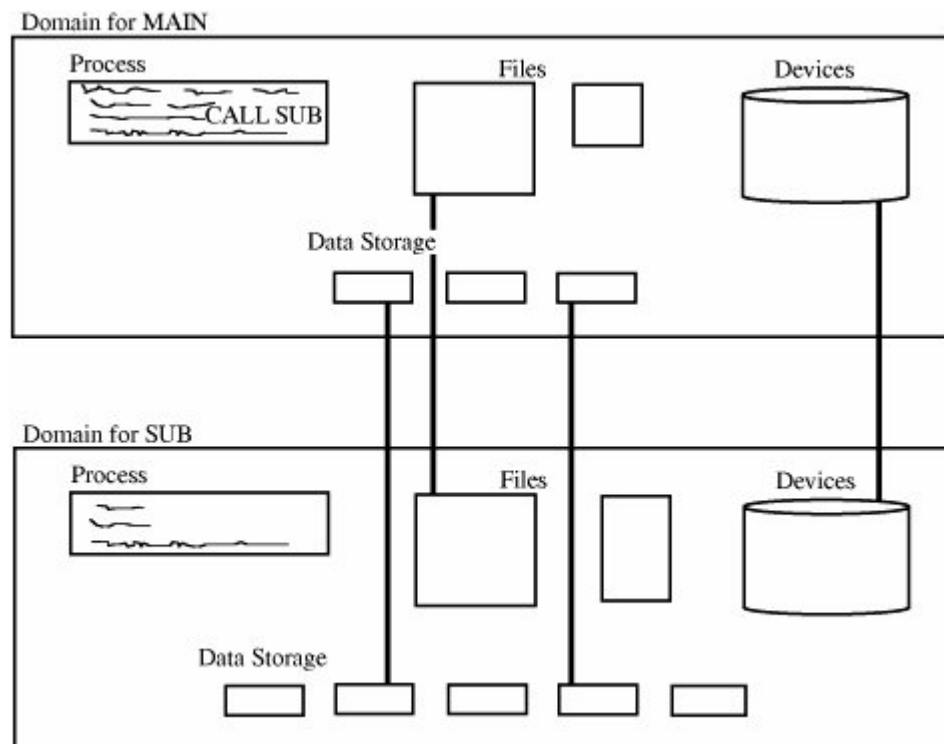


Figure 4.14. Passing Objects to a Subject.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Operationally, capabilities are an uncomplicated method to keep track associated with the access rights regarding subjects to objects in the course of execution. The capabilities happen to be backed up by some sort of more comprehensive table, many of these as an access management matrix or an entry control list. Each time a process seeks in order to utilize a new object , the operating-system examines the particular master listing of objects and even subjects to ascertain whether typically the object is accessible. If you do, the operating system produces a capability for of which object.

Capabilities should be kept in memory inaccessible in order to normal users. One method of accomplishing this really is to be able to store capabilities in sections not pointed at with the user's segment table in order to enclose them in protected memory as from a couple of base/bounds registers. Another strategy is to use some sort of tagged architecture machine in order to identify capabilities as set ups requiring protection.

During execution, only the capabilities regarding objects that have already been accessed with the current method are kept readily offered. This restriction improves typically the speed with which access to an object could be checked. This technique is basically the one used in Multics.

Capabilities can be suspended. For the issuing theme revokes a capability, zero further access beneath the terminated capability should be authorized. A capability table can easily contain pointers to typically the active capabilities spawned underneath it so that typically the operating system can find what access rights must be deleted if a capacity is revoked. A related problem is deleting abilities for users which are simply no longer active.

Kerberos

Essential research on capabilities placed the groundwork for future production use in systems for instance Kerberos. Kerberos implements both authentication and access authorization simply by means of capabilities, known as tickets, secured with symmetrical cryptography. Microsoft has established much of its gain access to control in NT+ in Kerberos.

Kerberos requires 2 systems, called the authentication server (AS) and the particular ticket-granting server (TGS), which usually are both section of the important distribution center (KDC). Some sort of user presents an authenticating credential (such as the password) to the authentication server and receives a new ticket showing that typically the user has passed authentication. Obviously, the ticket should be encrypted to prevent the particular user from modifying or even forging one claiming to be able to be a different customer, and the ticket should contain some provision to be able to prevent one user coming from acquiring another user's solution to impersonate that customer.

Now let us suppose that an user, Paul, would like to access a source R (for example, the file, printer, or system port). Joe sends the particular TGS his authenticated admission and a request to work with R. Assuming Joe is usually allowed access, the TGS returns to Joe a couple of tickets: One shows May well that his access in order to R has been certified, and the second is usually for Joe to existing to R in buy to access R.

Kerberos implements single sign-on; that will is, an user symptoms on once and inside the future all typically the user's (allowable) actions happen to be authorized without the customer needing to sign about again. When an consumer wants usage of a reference in a different domain name, say on a various system or in some sort of different environment or maybe a various company or

institution, mainly because long as authorization protection under the law happen to be established between the particular two domains, the customer's access occurs without typically the user's signing on to another system.

Kerberos accomplishes their local and remote authentication and authorization with a great approach to shared top secret encryption keys. In truth, each user's password is usually used as an security key.

Procedure-Oriented Access Control

One target of access control is definitely restricting not just which often subjects have access to be able to an object, but likewise the actual can carry out to that object. Go through versus write access may be controlled rather readily simply by most operating systems, although more complex control will be not so easy to attain.

By procedure-oriented protection, all of us imply the existence regarding a procedure that handles access to objects (for example, by performing its very own user authentication to improve the standard protection provided simply by the basic operating system). In essence, the process forms a capsule all-around the object, permitting just certain specified accesses.

Procedures can ensure that has access to an object end up being made via a trusted user interface. For example, neither consumers nor general main system regimens might be allowed instant access to the stand of valid users. Rather, the only accesses authorized might be through about three procedures: one to put an user, one to be able to delete an user, in addition to one to check regardless of whether a particular name matches to a valid end user. These procedures, especially put and delete, could employ their particular checks to help to make sure that calls for them are legitimate.

Procedure-oriented protection tools the principle of data hiding because the method of implementing a theme are known only to be able to the object's control method. Naturally, this degree associated with protection includes a penalty regarding inefficiency. With procedure-oriented security, there can be not any simple, fast access, fixed up object is regularly used.

Our survey regarding access control mechanisms offers intentionally progressed from very simple to complex. Historically, while the mechanisms have offered greater flexibility, they include succeeded in doing therefore with a price associated with increased overhead. For instance, implementing capabilities that wants to be checked upon each access is challenging than implementing a basic index structure that is inspected only on a subject's first access to a great object. This complexity is definitely apparent both for the customer and to the implementer. The user is conscious of additional protection characteristics, but the naive customer may be frustrated or perhaps intimidated at having in order to select protection options along with little knowledge of their performance. The implementation complexity turns into apparent in slow reply to users. The complete amount between simplicity and even functionality is a continuous battle in security.

Role-Based Access Control

We possess not yet distinguished between kinds of users, yet we wish some users (such as administrators) to experience significant privileges, and all of us want others (such like regular users or guests) to have lower benefits. In companies and academic institutions, this can obtain complicated for the normal user becomes an manager or a baker techniques to the candlestick makers' group. Role-based access handle allows us to connect privileges with groups, these kinds of as all administrators could accomplish this or candlestick makers are forbidden in order to do this. Administering protection is easier if all of us can control access by simply job demands, not by simply person. Access control retains up with an specific who changes responsibilities, and even the system administrator will not have to pick the appropriate access handle settings for somebody. For even more details on the detailed aspects of role-based access handle.

4.4. File Protection Mechanisms

Up to now, we have examined strategies to protecting a basic object, no matter typically the object's nature or variety. Sometimes protection schemes will be particular to the sort. To see the way they function, we focus within this part on file protection. Typically the examples we present will be only representative; they carry out not cover all probable means of file defense on the market.

Basic Forms of Protection

We mentioned earlier that all multiuser operating systems must give some minimal protection to be able to keep one user coming from maliciously or inadvertently being able to access or modifying the data files of another. As typically the quantity of users has cultivated, so even offers the difficulty of these protection strategies.

All None Protection

Inside the primary IBM OS systems, data were by default general public. Any user could study, modify, or delete some sort of file owned by any kind of other user. Instead involving software- or hardware-based security, the main protection involved have confidence in along with ignorance. System developers supposed that users can be trusted not in order to read or modify others' files because the customers would expect the identical respect from others. Lack of education helped this case, because a good user could access the file only by brand; presumably users knew typically the names only of these files to which that they had legitimate access.

Nevertheless, it was acknowledged of which particular system files have been sensitive and that typically the system administrator could guard them with an username and password. A typical user could workout this feature, but account details were viewed as many valuable for protecting running system files. Two sagesse guided password use. Occasionally, passwords controlled all has access to (read, write, or delete), giving the system manager complete control of most files. But quite frequently passwords controled only compose and delete accesses given that these two

actions influenced others. In either situation, the password mechanism needed a system operator's treatment each time access to be able to the file began.

Even so, this all-or-none protection is usually unacceptable for a number of reasons.

- **Lack of trust :** The presumption of trustworthy users will be not necessarily justified. Regarding systems with few customers who all know every other, mutual respect may possibly suffice; but also in large methods where its not most user knows every various other user, there is zero basis for trust.
- **Too coarse:** Even if a great user identifies an established of trustworthy users, generally there is no convenient solution to allow access only in order to them.
- **Rise of sharing :** This protection scheme is usually more suitable for the batch environment, through which consumers have little opportunity to socialize with other users and even in which users conduct their thinking and checking out when not reaching typically the system. However, on shared-use systems, users interact together with others and programs symbolizing other classes of customers.
- **Complexity:** Because (human) owner intervention is required intended for this file protection, running system performance is degraded. For this reason, this specific type of file defense is discouraged by calculating centers for all nevertheless the most sensitive files sets.
- **File listings:** With regard to accounting purposes and in order to help users remember intended for what files they are usually responsible, various system resources can make a record of all files. As a result, users are not always ignorant of what documents reside on the program. Interactive users may try out to browse through virtually any unprotected files.

Group Protection

As the all-or-nothing approach has numerous drawbacks, researchers sought a better way to protect documents. They focused on discovering groups of users that had some common romantic relationship. In the typical Unix+ setup, the planet is divided into 3 classes: the user, some sort of trusted working group connected with the user, and even the remaining portion of the users. Intended for simplicity we are able to call these types of classes user, group, plus world. Windows NT+ makes use of groups for instance Administrators, Electric power Users, Users, and Friends. (NT+ administrators can furthermore create other groups.)

All authorized users happen to be separated into groups. The group may consist regarding several members working upon a common project, the department, a class, or perhaps a single user. The schedule for group membership is usually must share. The party members incorporate some common curiosity and therefore are presumed to have files in order to share with another party members. In this strategy, no user belongs in order to multiple group. (Otherwise, the

member owned by groupings A and B can pass along an Some sort of file to another N group member.)

Whenever creating a file, the user defines access protection under the law to the file intended for the user, for various other members of the similar group, and for just about all other users generally speaking. Usually, the choices for gain access to rights are a restricted set, such as update, readexecute, read, writecreateddelete. For a particular record, an user might state read-only access to the particular general world, read plus update access to the particular girls, and all privileges to the user. This particular approach would be appropriate for a paper getting developed by a bunch, wherein the different members by the crew might improve sections being written inside the group. The papers itself should be readily available for people outside the class to examine but not really change.

A key good thing about the group protection strategy is its ease involving implementation. A person is recognized simply by two identifiers (usually numbers): an user ID in addition to a group ID. These types of identifiers are kept in the particular file directory entry regarding each file and happen to be obtained by the working system when an customer logs in. Therefore, the particular operating system can quickly check if the recommended access to a data file is requested from somebody whose group ID complements the group ID with regard to the file to become seen.

Although this protection plan overcomes some of the particular shortcomings of the all-or-nothing scheme, it introduces several new difficulties from the particular own.

Group affiliation: An individual user cannot belong in order to two groups. Suppose Mary is owned by one group using Ann and to some sort of second group with Costs. If Tom indicates that will a file is to be able to be readable by typically the group, to which group(s) does this permission recommend? Suppose a file associated with Ann's is readable simply by the group; does Expenses have access to that? These ambiguities are almost all simply resolved by proclaiming that every user is supposed to be to exactly one party. (This restriction does not necessarily mean that most users fit to the same team.)

Multiple personalities: To be able to overcome the one-person one-group restriction, certain people may possibly obtain multiple accounts, enabling them, in effect, in order to be multiple users. This specific hole inside the protection method leads to new issues because a single individual can be only 1 user each time. To discover how problems arise, imagine Tom obtains two balances, thereby becoming Tom1 within a group with Ann and Tom2 in the group with Bill. Tom1 is not really in the identical group as Tom2, thus any files, programs, or even aids developed under typically the Tom1 account could be offered to Tom2 only when they are offered to the whole world. Multiple personalities guide to a proliferation associated with accounts, redundant files, restricted protection for files involving general interest, and hassle to users.

All groupings: To avoid multiple personas, the device administrator may choose that Tom should include access to all his / her files any time he or she is active. This solution sets the responsibility on Mary to regulate with whom he or she shares what things. Intended for example, he might get in Group1 with Ann and Group2 with Invoice. He creates a Group1 file to share using Ann. But if they are active in Group2 next time he is logged throughout, he still sees typically the Group1 file and may possibly not understand that it will be not accessible to Costs, too.

Limited sharing: Data can be shared simply within groups or together with the world. Users would like to be able in order to identify sharing partners intended for a file on the per-file basis; for illustration, sharing one file using ten people and an additional file with twenty other folks.

Individual Permissions

In revenge of their drawbacks, the particular file protection schemes many of us have described are comparatively simple and straightforward. Typically the simplicity of implementing these people suggests other easy-to-manage strategies which provide finer degrees associated with security while associating agreement with a single data file.

Persistent Permission

From the other situations you are familiar along with persistent permissions. The common implementation of this scheme utilizes a name (you declare a dinner reservation beneath the name of Sanders), a symbol (you demonstrate your driver's license or perhaps library card), or the secret (you say a new secret word or offer the club handshake). In the same way, in computing you will be allowed access because they are upon the access list, offering a token or admission, or giving a pass word. User access permissions could be required for any kind of access or only with regard to modifications (write access).

Most these approaches present apparent difficulties in revocation: Using someone off one record is easy, but it really is considerably more complicated to find almost all lists authorizing someone in addition to remove him or the girl. Reclaiming a symbol or even password is more difficult.

Temporary Acquired Permission

Unix+ techniques provide an interesting authorization scheme based on a new three-level usergroupworld hierarchy. The particular Unix designers added a new permission called set userid (suid). Issue protection is definitely set for a data file to be executed, typically the protection level is of which of the file's user, not the executor. To be able to see how it performs, suppose Tom owns a new file and allows Ann to execute it using suid. When Ann completes the file, she features the protection rights associated with Tom, not of himself.

This peculiar-sounding permission features an useful application. This permits an user to determine data files to which often access is allowed just through specified procedures.

Intended for example, suppose you need to build a computerized dating services that manipulates a databases of folks available on certain nights. Sue might get interested in a particular date for Saturday, but the lady may have already refused a new request from Jeff, declaring she had other programs. Sue instructs the assistance to not reveal to Barry that she's available. In order to use the service, Drag into court, Jeff, and others should be able to read typically the file and write in order to it (at least indirectly) to determine who will be available or to write-up their availability. But in case Jeff can read the particular file directly, he might discover that Sue has humiliated. Consequently, your dating assistance must force Sue in addition to Jeff (and all others) to access this document only through an entry program that would monitor your data Jeff obtains. Yet if the file gain access to is limited to go through and write by a person as the owner, File suit and Jeff will never ever be able to get into data into it.

Typically the solution is the Unix SUID protection. You generate the database file, offering only you access agreement. You additionally write the plan that is to reach the particular database, and save that with the SUID security. Then, when Jeff completes your program, he quickly acquires your access agreement, but only during performance of the program. Shaun never has direct gain access to to the file mainly because your program will carry out the exact file access. If Jeff exits from the program, he regains their own access rights in addition to loses yours. Thus, your own program can access typically the file, but the software must display to Shaun only the data Barry is allowed to observe.

This mechanism is easy for system functions of which general users should get able to perform simply within a prescribed way. Regarding example, the particular program should be able to be able to modify the file associated with users' passwords, but personal users must be able to change their very own own passwords at any time these people wish. With the SUID feature, a password modification program can be possessed with the system, which can therefore have full gain access to to the system pass word table. The program to be able to change passwords also offers SUID protection to ensure that if a normal user completes it, the program could modify the password record in a carefully limited way on behalf involving the person.

Per-Object and Per-User Protection

The primary restriction of these protection strategies is the ability to be able to create meaningful groups regarding related users who ought to have similar entry to connected objects. The access handle lists or access manage matrices described earlier give very flexible protection. Their own disadvantage is for the person who wants to enable access to many customers and to many various data sets; such a great user must still stipulate each data set in order to be accessed by every user. As a fresh user is added, of which user's special access privileges must be specified simply by all appropriate users.

4.5. User Authentication

An OS bases much of their protection on knowing that an user of the particular system is. In real life situations, people commonly question for identification from men and women they do not realize: A bank employee may well ask for a driver's license before cashing typically the, library employees may demand some identification before asking out books, and settlement officials ask for given as evidence of personality. In-person identification is often simpler than remote identification. Regarding instance, some universities never report grades over the particular telephone because the workplace workers do not automatically know the students dialling. Nevertheless, a professor that recognizes the voice involving a certain student can easily release that student's marks. Over time, organizations and even software has developed indicates of authentication, using papers, voice recognition, fingerprint plus retina matching, and additional trusted means of identity.

In computing, the options are more limited along with the possibilities less secure. Any individual can attempt to sign in into a computing program. Unlike the professor which recognizes a student's words, the computer cannot identify electrical signals in one individual as being any totally different from those of anyone different. Thus, most computing authentication systems has to be based about some knowledge shared simply by the computing method and the user.

Authentication mechanisms use any involving three qualities to verify an user's identity.

- **Some thing the user knows.** Accounts, PIN numbers, passphrases, some sort of secret handshake, and mom's maiden name are samples of what an user might know.
- **Something the customer has.** Identity badges, actual physical keys, a driver's certificate, or an uniform usually are common examples of issues people have that help make them recognizable.
- **Something the particular user is.** These authenticators, called biometrics, depend on some sort of physical characteristic of the particular user, such as a new fingerprint, the pattern associated with a person's voice, or even a face (picture). These authentication methods are old (we recognize friends in man or woman by way of some sort of faces or on some sort of telephone by their voices) but are just beginning to be used found in computer authentications.

Two or a lot more forms could be combined with regard to more solid authentication; intended for example, a bank credit card along with a PIN combine some thing the consumer has with anything the person knows

Passwords as Authenticators

The nearly all common authentication mechanism with regard to user to operating technique is a password, a new "word" known to PC and user. Although password protection seems to present a relatively secure method, human practice sometimes degrades its quality. With this segment we

consider passwords, requirements for selecting them, in addition to ways of using these people for authentication. We consider by noting other authentication techniques through studying troubles in the authentication method, notably Trojan horses masking as the computer authentication process.

Use of Security passwords

Passwords are mutually agreed-upon code words, assumed to be able to be known only in order to the user and the particular system. In some instances an user chooses account details; in other cases the device assigns them. The span and format of the particular password also vary coming from one system to a new.

Also though they are commonly used, passwords suffer coming from some difficulties of usage:

- **Loss.** Depending on how the particular passwords are implemented, this is possible that not any one will be ready to replace a dropped or forgotten password. The particular operators or system directors can certainly intervene in addition to unprotect or assign a specific password, but often they can determine what password the user has chosen; when the user loses the particular password, home must become assigned.
- **Use.** Supplying some sort of password for each entry to a file can become inconvenient and time taking in.
- **Disclosure.** If a pass word is disclosed to a great unauthorized individual, the data file becomes immediately accessible. In the event that the user then alters the password to reprotect the file, all typically the other legitimate users has to be informed of the fresh password because their aged password will fail.
- **Revocation..** To revoke one wearer's access right to the file, someone must transform the password, thereby evoking the same problems since disclosure.

The use regarding passwords is fairly easy. A user enters a few part of identification, such since a name or a great assigned user ID; this kind of identification can be accessible to the public or quick to guess because that does not provide the particular real security of the particular system. The machine then needs a password from typically the user. If the username and password matches that on use for the user, the particular user is authenticated in addition to allowed access to the device. If the password complement fails, the system asks for the password again, within case the user mistyped.

Additional Authentication Information

Besides the name and password, we are able to use other information obtainable to authenticate users. Assume Adams works in typically the accounting department through the switch between

8: 00 a new. m. and 5: 00 p. m., Monday by way of Friday. Any legitimate entry attempt by Adams have to be made during all those times, through a workstation within the accounting department workplaces. By limiting Adams in order to logging in under individuals conditions, the machine protects towards two problems:

Someone through outside might try in order to impersonate Adams. This test would be thwarted simply by either the time regarding access or the interface through which the gain access to was attempted.

Adams may well attempt to access the program from home or about a weekend, planning in order to use resources prohibited or even to do something of which would be too high-risk with other people about.

Limiting users to specific workstations or certain instances of access can result in complications (as when a great user legitimately should operate overtime, a person offers to get into the system although out of town over a business trip, or the particular workstation fails). Nevertheless, some companies use these types of authentication techniques because typically the added security they offer outweighs inconveniences.

Using further authentication information is named multifactor authentication. Two types of authentication (which is, obviously, acknowledged as two-factor authentication) vs. one, assuming of study course that this two forms are usually strong. But as typically the number of forms rises, so also does typically the inconvenience. (For example, consider about getting through the security checkpoint at a great airport.) Each authentication factor requires the method and its administrators in order to manage more security info.

Attacks on Passwords

Exactly how secure are passwords them selves? Passwords are somewhat restricted as protection devices as a result of relatively small number regarding bits of information these people contain.

Here are a few ways you might end up being able to determine a great user's password, in reducing order of difficulty.

Consider all possible passwords.

- Attempt frequently used passwords.
- Try out passwords likely for typically the user.
- Hunt for the program list of passwords.
- Request the user.

Loose-Lipped Systems

So far the particular process seems secure, although in fact it includes a few vulnerabilities. To see exactly why, consider the actions associated with a would-be intruder. Authentication is based on the particular actual <name, password> pair. A complete intruder is presumed to realize nothing of the method. Suppose the intruder endeavors to access a method in the following way. (In the following cases, the system messages are generally in uppercase, and typically the user's responses are usually in lowercase.)

WELCOME TO TYPICALLY THE XYZ COMPUTING SYSTEMS

ENTER IN USER NAME: adams

ILL USER NAMEUNKNOWN USER

GET INTO USER NAME:

We thought that the intruder recognized nothing of the program, but and not having to do a lot, the intruder found out there that adams is not really typically the name of an approved user. The intruder may try other common titles, first names, and very likely generic names like technique or operator to create a new list of authorized customers.

An alternative solution arrangement of the particular login sequence is proven below.

THANKS FOR VISITING THE XYZ COMPUTING DEVICES

ENTER CONSUMER NAME: adams

ENTER PASS WORD: john

INVALID ACCESS

ENTER IN USER NAME:

This method notifies a user associated with a failure only right after accepting both the customer name and the security password. The failure message ought to not indicate unique typically the user name or pass word that is unacceptable. Inside this way, the burglar does not know which usually failed.

These examples furthermore gave a clue about which computing system is definitely being accessed. The legitimate outsider does not have right in order to know that, and legit insiders already know just what system they may have accessed. Inside the example below, the particular user is given not any information until the technique is assured in the personality of the user.

GET INTO USER NAME: adams

GET INTO PASSWORD: john

INVALID ENTRY

ENTER USER NAME: adams

ENTER PASSWORD: johnq

HERE YOU ARE AT THE XYZ COMPUTING TECHNIQUES

Exhaustive Attack

In a good exhaustive or brute power attack, the attacker attempts all possible passwords, normally in certain automated fashion. Regarding course, the quantity of possible security passwords depends on the execution of the particular processing system. For example, in case passwords are words containing of the 26 figures AZ and can turn out to be of any length by 1 to 8 heroes, there are 26¹ security passwords of 1 character, 26² passwords of 2 figures, and 26⁸ passwords involving 8 characters. Therefore, typically the system in general has $26^1 + 26^2 + \dots + 26^8 = 26^9 - 1 \approx 5 \times 10^{12}$ or five million feasible passwords. The number involving seems intractable enough. When we were to work with a computer to produce and even try each password in a rate of looking at one password per nanosecond, it could take on the particular order of 150 decades to test all account details. But if we might accelerate the search in order to one password per microsecond, the work factor falls to about 8 days. This amount of moment is reasonable if the particular reward is large. With regard to instance, an intruder may well try to break the particular password on a document of bank card numbers or perhaps bank account information.

Although the break-in time may be made more tractable in several ways. Searching for some sort of single particular password will not necessarily require almost all passwords to be tried out; an intruder needs to be able to try only until typically the correct password is determined. If the group of most possible passwords were equally distributed, an intruder is likely to need to try simply half the password place: the expected number involving searches to find virtually any particular password. However, a good intruder can also work with to advantage the point that account details are not evenly sent out. As a password has in order to be remembered, people have a tendency to pick simple accounts. This feature reduces the dimensions of the password space.

Possible Passwords

Think of the word.

Could be the word a person thought of long? Is definitely it uncommon? Is that challenging to spell or in order to pronounce? The response to almost all three of these issues is probably no.

Penetrators looking for passwords realize these kinds of very human characteristics and even rely on them to their edge. Therefore, penetrators try strategies that are prone to prospect to rapid

success. In case people prefer short account details to long ones, typically the penetrator will plan in order to try all passwords although to try them throughout order by length. Right now there are only 26^1 and up. $26^2 + 26^3 = 18,278$ passwords of length a few or less. At the particular assumed rate of 1 password per millisecond, almost all of these passwords could be checked in 18,278 seconds, hardly a concern having a computer. Even growing the tries to 5 or 5 characters increases the count only to be able to 475 seconds (about 6 minutes) or 12,356 seconds (about 3. five hours), respectively

15	0.50%	were a single(!) ASCII character
72	2%	were two ASCII characters
464	14%	were three ASCII characters
477	14%	were four alphabetic letters
706	21%	were five alphabetic letters, all the same case
605	18%	were six lowercase alphabetic letters
492	15%	were words in dictionaries or lists of names
2831	86%	total of all above categories

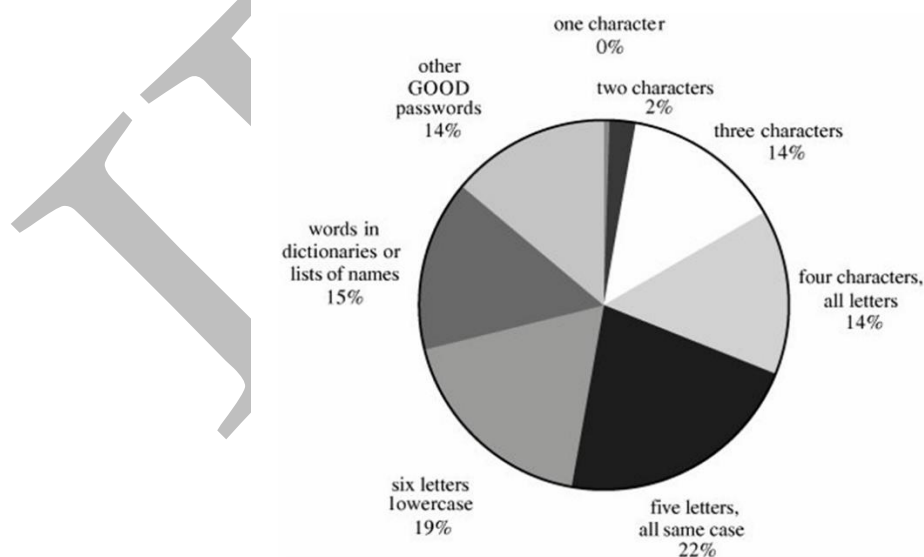


Figure 4.15. Users' Password Choices.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Lest you dismiss these results as dated (they were claimed in 1979), Klein frequent the experiment in 1990 and Spafford in 1992. Each gathered roughly 15,000 passwords. Klein claimed that 2.7 per-cent of this passwords have been guessed in mere quarter-hour of machine moment and 21 percentage were guessed within a week! Spafford located the average password length was initially 6.8 heroes, and 28.9 percent consisted of simply lowercase alphabetic heroes. Observe that both these analyses were done after the Web worm (defined in Section 3) succeeded, partly by breaking poor passwords.

Even in 2002, the British online bank Egg found users nonetheless choosing poor passwords. A full 50 pct of passwords for his or her online banking service were family customers' labels: 23 percentage children's names, 19 percent a husband or wife or companion, and 9 percent their own. Alas, pets arrived in at only 8 percent, while stars and soccer (soccer) stars tied at 9 percent each. And in 1998, Knight and Hartley reported that about 35 percent of passwords are usually deduced from syllables and initials from the account owner's name.

Two friends we know have told us their passwords once we aided them administer their methods, and their passwords would both have been among the first we would contain guessed. But, you claim, these are amateurs unacquainted with the security risk of a weakened password. At a recently available meeting, a security and safety expert associated this experience: He thought he had picked a solid security password, so he asked a category of learners to request him a few questions and offer some guesses concerning his password. He was surprised that they questioned only a few questions before they had deduced the password. And this was a security specialist.

Several news posts have claimed which the four most typical passwords happen to be "God," "sex," "love," and "money" (the order among those is unspecified). The possibly apocryphal set of prevalent passwords at geodsoft.com/howto/security/password/common.htm appears at other places on the Internet. Or start to see the default password checklist at www.phenoelit.de/dpl/dpl.html. Whether these are seriously passwords we have no idea. Nevertheless, it warrants a peek because similar lists are bound to be built into some hackers' equipment.

Several network internet sites posting dictionaries of phrases, technology fiction characters, spots, mythological names, Chinese language words, Yiddish phrases, and other specialised lists. All these lists are uploaded to help webpage administrators identify customers who have chosen weak passwords, but the same dictionaries can also be used by attackers of sites that do not need like attentive administrators. The COPS ,Break , and SATAN utilities allow an administrator to scan something for weak passwords. But these very same utilities, or various other homemade ones, let attackers to accomplish the same. Nowadays Internet sites present so-called password

recuperation software program as freeware or shareware for under \$20. (These are password-cracking courses.)

People think they could be clever by picking a simple password and replacing certain characters, such as for example 0 (zero) for letter O, 1 (one) for letter I or L, 3 (three) for letter E or @ (at) for letter A. But consumers aren't the only real individuals who could come up with these substitutions. Knight and Hartley record, in order, 12 tips an attacker might attempt to be able to determine a security password. These steps are in increasing amount of difficulty (number of guesses), so they indicate the quantity of work to that your attacker must go to derive a password. Listed below are their password speculating steps:

- no password
- exactly like an individual ID
- is, or is derived from, the user's name
- common word list (for example, "password," "hidden knowledge," "non-public") plus common names and designs (for instance, "asdfg," "aaaaaa")
- short university dictionary
- complete English term list
- common non-English terminology dictionaries
- short college or university dictionary with capitalizations (PaSsWoRd) and substitutions (0 for O, and so forth)
- complete English with capitalizations and substitutions
- popular non-English dictionaries with capitalization and substitutions
- brute force, lowercase alphabetic characters
- brute force, complete character set

Although the final step will always succeed, the ways right away preceding it are so frustrating that they can deter all however the devoted attacker for whom period isn't a limiting component.

Plaintext System Security password List

To validate passwords, the machine must have a way of comparing entries with actual passwords. Rather than trying to speculate a user's password, an attacker may as an alternative target the system password document. Why think when with one table you can ascertain all passwords with complete accuracy?

On some systems, the password listing is a data file, organized essentially being a two-column stand of person IDs and equivalent passwords. This information is certainly also obvious to leave out in the wild. Various security methods are used to conceal this stand from those that should not view it.

You might defend the desk with strong admittance controls, limiting usage of the operating-system. But perhaps this tightening up of control is looser than it should be, because don't assume all operating system component needs or deserves usage of this table. For instance, the operating-system scheduler, accounting exercises, or storage supervisor have no need to know the table's material. Unfortunately, in a few systems, there are $n+1$ known customers: n regular users along with the operating-system. The operating system isn't partitioned, consequently all its modules have access to all privileged info. This monolithic view of the operating-system implies that a end user who exploits a flaw in a single portion of the operating system has access to all the system's deepest secrets and techniques. A better strategy is to control table usage of the modules that require access: the user authentication module and the parts connected with installing new consumers, for example.

If the stand is stashed in plain eyesight, an intruder can merely dump memory with a convenient time and energy to access it. Cautious timing may allow a user to dump the contents of all of recollection and, by exhaustive lookup, find worth that appear to be the password stand.

System backups could also be used to obtain the password table. In order to recover from system errors, system administrators regularly back up the file room onto some auxiliary channel for safe storage space. In the unlikely event of a problem, the file system can be reloaded from the backup, using a loss simply of changes made since the last backup. Backups normally contain only record contents, without protection mechanism to control file gain access to. (Physical protection and access settings for the backups themselves are usually depended on to provide security for the articles of backup press.) If a regular customer can access the backups, perhaps ones from weeks, months, or years ago, the password tables stored inside them may include entries which are still valid.

Finally, the password file is a copy of an file saved on disk. Anyone with access to the drive or anyone who can overcome file gain access to restrictions can buy the password file

Encrypted Security password File

There is an easy way to foil an intruder seeking passwords in simple perception: encrypt them. Regularly, the password record is disguised . from view with conventional encryption or one-way ciphers.

With normal encryption, either the complete password table will be encrypted or simply the password column. Whenever a user's password is certainly received, the stored password is usually decrypted, and both are compared.

Even with encryption, there is still a slight visibility because for an instantaneous the user's security password comes in plaintext in key memory. That's, the password can be acquired to anyone who could obtain access to all of memory.

A safer approach utilizes one-way encryption. The password table's entries happen to be encrypted by way of a one-way encryption and then stored. Once the user gets into a password, additionally it is encrypted and compared with the table. If both values are similar, the authentication succeeds. Of course, the encryption must be so that it is improbable that two passwords would encrypt to exactly the same ciphertext, but this feature is true for most risk-free encryption algorithms.

With one-way encryption, the password file can be stored in ordinary view. For example, the password stand for that Unix operating system can be read by any user unless special accessibility controls have already been installed. As the contents will be encrypted, backup copies of the security password table are no more a problem.

There's always the possibility that two people might choose the same password, so creating two similar entries in the password file. Despite the fact that the entries are usually encrypted, each user will understand the plaintext equal. For example, if Costs and Kathy both select their passwords on April 1, they might choose APRILFOOL as a password. Bill might read the password record and observe that the encrypted type of his security password is equivalent to Kathy's.

Unix+ circumvents this vulnerability by using a password extension, named the salt. The salt is really a 12-bit number shaped from the machine time and the procedure identifier. Hence, the salt is likely to be unique for every user, and it can be stored in plaintext within the password data file. The salt is certainly concatenated to Bill's password (pw) when he selects it; $E(pw + salt_B)$ is stored for Costs, and his salt value can be kept. When Kathy chooses her password, the salt differs because the time or the process number is different. Call this different one saltK. On her behalf, $E(pw + salt_K)$ and saltK will be placed. When either individual tries to log in, the machine fetches the correct salt in the password desk and includes that while using password before accomplishing the encryption. The encrypted versions of (pw+sodium) are very different for both of these users. When Costs looks down the security password list, the encrypted edition of his security password will not take a look at all like Kathy's.

Storing the password file in the disguised contact form relieves much of the pressure to safeguarded it. Better still is to limit access to operations that legitimately will need access. In this manner, the password data file is secured to an even commensurate while using protection provided by the security password itself. Someone who has broken the control buttons of the file system has access to data, not only passwords, which is a serious menace. But if an attacker effectively penetrates the external security covering, the attacker nevertheless must see through the encryption of the password file to access the useful data in it.

Indiscreet Users

Speculating passwords and bursting encryption can be tedious or challenging. But there is a simple way to obtain a security password: Get it directly from an individual! People frequently tape a security password aside of a terminal or write it over a card just inside the top workplace drawer. Users are afraid they will forget their passwords, or they can not be bothered attempting to remember them. It really is particularly tempting to write the passwords down when customers have several addresses.

Users sharing function or data can also be tempted to talk about passwords. If an individual needs a record, it is better to say "my security password is x; obtain the file yourself" than to arrange to share the file. This example is a result of user laziness, nonetheless it may be caused or exacerbated by way of a system that makes sharing inconvenient.

Within an admittedly unscientific poll performed by Verisign, two-thirds of individuals approached on the road volunteered to reveal their password to get a coupon best for a cup of coffee, and 79 percentage admitted they applied the same password for several system or site.

Password Selection Criteria

On the RSA Security Seminar in 2006, Bill Gates, head of Microsoft, explained his perspective of a global where passwords would be obsolete, having long gone the way in the dinosaur. In their place innovative multifactor authentication technology would offer much larger security than passwords ever could. But that is Bill Gates' watch of the future; despite generations of articles or blog posts about their weakness, passwords are usually with us nonetheless and will be for quite a while.

So what can we conclude about passwords? They must be hard to imagine and difficult to find out exhaustively. But the degree of issues should be correct to the safety measures needs of the problem. To these edges, we present different guidelines for password selection:

Use characters apart from only AZ. If passwords happen to be chosen from letters AZ, you can find only 26 alternatives for each figure. Adding digits expands the number of options to 36. Employing both uppercase and lowercase letters plus digits expands the number of possible figures to 62. Although this change seems small, the result is large when someone is testing a full space of most possible combos of characters. It requires about 100 hours to check all 6-letter words selected from letters of 1 case only, but it takes about 2 years to test all 6-mark passwords from top- and lowercase letters and digits. Although 100 hours is reasonable, 2 years is oppressive enough to make this attack far less attractive.

Choose long passwords. The combinatorial explosion of passwords begins at length four or five. Choosing much longer passwords helps it be less likely a password will undoubtedly be uncovered. Understand that a brute power penetration can quit as soon as the password is found.

Some penetrators will try the easy cases known words and quick passwords and move ahead to another concentrate on if those episodes fail.

Avoid actual titles or text. Theoretically, there are 26^6 or about 300 million 6-letter "words", but you can find only about 150,000 text in an excellent collegiate dictionary, overlooking length. By choosing one of the 99.95 per-cent nonwords, you power the attacker to use a longer brute drive search instead of the abbreviated dictionary look for.

Choose an unlikely password. Password option is a dual bind. To keep in mind the password easily, you want one which has special significance to you. On the other hand, you don't wish someone else in order to imagine this special meaning. One easy-to-remember password will be 2Brn2B. That improbable looking jumble is really a simple transformation of "for being or not to be." The first letters of phrases from a music, a few characters from different thoughts of an exclusive phrase, or perhaps a memorable basketball score are types of realistic passwords. But don't be too noticeable. Password-cracking tools in addition test replacements of 0 (zero) for o or O (notice "oh") and 1 (one) for l (notice "ell") or \$ for S (letter "ess"). Consequently I10veu has already been in the research file.

Change the password regularly. Even though there is absolutely no reason to think that the password has been jeopardized, change is advised. A penetrator may split a password system by obtaining an old list or working exhaustively on an encrypted list.

Don't create it down. (Note: This time-honored tips is relevant only when physical security is a serious risk. People who have accounts on many different machines and machines, not to mention bank and bank card PINs, could have trouble remembering all of the access codes. Setting up all codes exactly the same or employing insecure but easy-to-remember passwords may be more dangerous than composing passwords on the reasonably well guarded list.)

Don't tell anyone else. The easiest attack is **social engineering**, where the attacker contacts the system's administrator or a customer to elicit the password for some reason. For instance, the attacker may phone a user, case to end up being "system supervision," and have the user to verify the user's security password. Under no circumstances should you ever give out your private security password; genuine administrators can circumvent your password if need be, and others are merely trying to deceive you.

To help users select good passwords, some techniques present meaningless but pronounceable passwords. For instance, the VAX VMS technique randomly creates five passwords from which the user selects one. They're pronounceable, so the user can duplicate and memorize them. However, the user may misremember a security password because of getting interchanged syllables or characters of your meaningless string. (The noise "bliptab" is no easier misremembered than "blaptib" or "blabtip.")

Yan et al. performed experiments to find out whether consumers could keep in mind passwords or passphrases far better. First, they found that users are terrible at remembering arbitrary passwords. And directions to users about the importance of choosing good passwords possessed little effect. However when they asked users to select their very own password based on some mnemonic word they selected themselves, the consumers selected passwords that were harder to guess than normal (not predicated on a saying) passwords.

Other systems motivate users to change their passwords frequently. The regularity of password change is usually a system parameter, which may be changed for your characteristics of confirmed installation. Suppose the frequency is defined at 1 month. Some systems begin to warn the user after 25 times that the password is about to expire. Others hold out until 1 month and inform the user that the security password has expired. Some methods nag without end, whereas other devices take off a user's access if a security password has expired. Still others force the user immediately in to the password change power on the initial login after 1 month.

Grampp and Morris claim that reminder process is not necessarily good. Choosing passwords isn't difficult, but under great pressure a user may choose any password, merely to fulfill the system's need for a fresh one. In addition, if this is the only moment a password could be changed, a negative password choice cannot be changed before next scheduled moment.

One-Time Passwords

A one-time password will be one that adjustments every time it is used. Instead of assigning a static expression to a customer, the machine assigns a static numerical function. The system provides an debate to the function, and the user computes and comes back the function value. Such systems are also called challengerresponse techniques because the technique presents a challenge to an individual and judges the authenticity of an individual with the user's response. Here are some simple examples of one-time password features; these functions are overly simplified to make the explanation easier. Highly complex functions may be used in place of these simple kinds for coordinator authentication in a network.

$f(x) = x + 1$. With this particular function, the machine prompts using a benefit for x , and an individual enters the worthiness $x + 1$. The forms of mathematical functions used are limited only by the power of the user to compute the reaction efficiently. Other similar options happen to be $f(x) = 3x^2 - 9x + 2$, $f(x) = p_x$, where p_x is the x th prime variety, or $f(x) = d * h$, where d may be the time and h may be the hour of the current time. (Alas, several users cannot execute simple arithmetic within their heads.)

$f(x) = r(x)$. For this function, the receiver uses the argument because the seed for your random amount generator (available to both the recipient and coordinator). An individual replies with the value of the initial random number developed. A variant of this scheme utilizes x as a number of random numbers to generate. The receiver produces x random figures and sends the x th of these to the variety.

$f(a_1a_2a_3a_4a_5a_6) = a_3a_1a_1a_4$. With this particular function, the system provides a character string, that your user must transform in some predetermined manner. Again, many different character operations can be used.

$f(E(x)) = E(D(E(x)) + 1)$. In this particular function, the laptop directs an encrypted price, $E(x)$. An individual must decrypt the value, perform some numerical performance, and encrypt the result to come back it to the machine. Clearly, for individual employ, the encryption performance must be a thing that can be done easily by hand, unlike the good encryption algorithms. For machine-to-machine authentication, on the other hand, an encryption algorithm such as for example DES or AES is appropriate.

One-time passwords are very very important to authentication because an intercepted security password is useless since it cannot be reused. However, their effectiveness is limited with the complexness of algorithms people should be expected to keep in mind. A password-generating machine can implement more technical functions. Several types are readily available at reasonable prices. They are quite effective at countering the risk of transmitting passwords in plaintext across a network.

The Authentication Process

Authentication usually works as described formerly. However, users once in a while mistype their passwords. A end user who receives a note of INCORRECT LOGIN will thoroughly retype the login and access the system. A good user who's a terrible typist can log in successfully in several tries.

Some authentication procedures are intentionally sluggish. A legitimate individual won't complain when the login process can take 5 or 10 a few moments. To some penetrator who's striving an exhaustive search or a dictionary search, however, 5 or 10 moments per trial can make this school of attack usually infeasible.

A person whose login makes an attempt continually fail may not be an authorized person. Systems commonly detach a user following a few failed logins, forcing an individual to reestablish a connection with the machine. (This step will slow down a penetrator who is trying to permeate the machine by telephone. Aft In more secure installations, ending penetrators is even more significant than tolerating consumers' mistakes. For example, some technique administrators assume that legitimate customers can form their passwords appropriately within three tries. After three successive security password failures, the account for that user is certainly disabled in support of the safety administrator can reenable it. This action identifies accounts that may be the prospective of attacks by penetrators.

Fixing Flaws inside the Authentication Process

Password authentication assumes that anyone who is aware of a password may be the individual to whom the password belongs. As we have seen, passwords can be guessed, deduced, or inferred. Some people hand out their passwords for the asking. Some other passwords have already been obtained simply by someone observing a end user typing inside the password. The password can be considered as a preliminary or first-level piece of evidence, but skeptics will need more convincing proof.

There are several ways to provide a second level of protection, incorporating another round of passwords or perhaps a challenge-response interchange.

Challenge Response Systems

As we contain just noticed, the login is usually period invariant. Except when passwords happen to be evolved, each login appears like every other. A more sophisticated login requires a user Identification and password, accompanied by a challenge-response interchange. In such an interchange, the system prompts the user for an answer which will be different each time an individual logs in. For instance, the machine might exhibit a four-digit variety, and the user would have to correctly enter a function like the sum or product or service on the digits. Each customer is assigned a different challenge work to compute. Because there are many possible challenge functions, a penetrator who catches the user Identification and password cannot always infer the proper function.

A physical device similar to a calculator may be used to implement a far more complicated response purpose. The user gets into the challenge number, and these devices computes and displays the response for the user to enter order to log in. (er a small number of problems, the penetrator must reconnect, which takes a couple of seconds.)

Impersonation of Login

In the systems we have defined, the proof will be one-sided. The machine demands certain recognition of the user, but the consumer is supposed to trust the system. However, a programmer can easily write a program that displays the standard prompts for person ID and security password, captures the pair entered, retailers the pair inside a file, exhibits SYSTEM Problem; DISCONNECTED, and exits. This assault is a type of Trojan horse. The perpetrator pieces it up, departs the terminal unattended, and waits for an innocent victim to try a login. The naive sufferer may not perhaps suspect a security breach has occurred.

To foil this sort of attack, the user should be sure the road to the system is reinitialized each and every time the system is used. On some systems, turning the terminal on / off again or pressing the BREAK major generates a clear signal for the computer to prevent any running method with the terminal. (Microsoft selected <CTRLALTDELETE> because the way to the safe authorization mechanism because of this.) Don't assume all computer recognizes power-off or

Separate being an interruption of the current method, though. And processing systems tend to be accessed through systems, so physical reinitialization is impossible.

Alternatively, the user can be suspicious of the processing system, in the same way the system is suspicious of the user. The user won't enter confidential files (like a security password) until persuaded that the processing system is legitimate. Needless to say, the laptop acknowledges the user only after passing the authentication process. A computing method can display some information acknowledged only by an individual and the system. For example, the system might read the user's label and reply "YOUR Final LOGIN Had been 10 APRIL AT 09:47." An individual can verify the date and period are accurate before stepping into a secret password. If higher security is desired, the system can mail an encrypted timestamp. An individual decrypts this and discovers that enough time is current. The user then replies with an encrypted timestamp and password, to convince the machine that a harmful intruder has not intercepted a security password from some prior login.

Biometrics: Authentication Not really Using Passwords

Some sophisticated authentication devices are actually available. The unit consist of handprint detectors, speech recognizers, and identifiers of patterns in the retina. Authentication with such devices uses unforgeable physical features to authenticate customers. The cost is constantly on the fall as these devices are implemented by major market segments; the devices are useful in very high security situations. In this particular section we consider a several approaches available.

Biometrics are natural authenticators, predicated on some physical feature of our body. The set of biometric authentication systems is still developing. Now there will be devices to recognize the following biometrics: fingerprints, hand geometry (shape and size of fingers), retina and iris (parts of the eye), tone, handwriting, blood vessels in the finger, and encounter. Authentication with biometrics features benefits over passwords because a biometric can't be lost, stolen, forgotten, lent, or forged and is definitely available, always accessible, so to speak.

Id versus Authentication

Two concepts are easily confused: id and authentication. Biometrics have become reliable for authentication but significantly less reputable for authentication. The reason is mathematical. All biometric viewers work in two phases: First of all, a consumer registers with the reader, where time a feature of an individual (for example, the geometry in the hand) is taken and reduced to a template or style. During registration, an individual may be asked to present the hand many times so that the registration software program can adjust for variations, such as for example how the

hands is positioned. Next, the user in the future looks for authentication from the system, during which time the machine remeasures the hand and compares the brand new measurements together with the stored template. If the new measurement is nearby enough to the template, the system accepts the authentication; in any other case, the machine rejects it. Every template is therefore a routine of some amount of measurements.

Unless every design template is unique, that's, no two people have exactly the same measured hand geometry, the system cannot uniquely determine subjects. However, so long as it is improbable that an imposter could have exactly the same biometric template as the real user, the system can authenticate. The distinction is between a system that talks about a side geometry and says "this is Captain Hook" (recognition) versus a man who claims "I, Captain Hook, provide my palm to show who I'm" and the machine confirms "this side fits Captain Hook's template" (authentication). Biometric authentication can be feasible today; biometric identification is basically still a study topic.

Problems with Biometrics

There are many problems with biometrics:

Biometrics are fairly new, and some people get their employ intrusive. Palm geometry and deal with recognition (which can be done from a camera over the room) are usually scarcely invasive, but people have real worries about peering into a laser beam or sticking a hand into a slot. for some examples of persons resisting biometrics.)

Biometric recognition gadgets are expensive, although as the devices are more popular, their costs go down. Even now, outfitting every user's workstation with a reader can be expensive for a big company with many employees.

All biometric visitors apply sampling and set up a threshold for when a match is near enough to accept. The device has to sample the biometric, measure often hundreds of tips, and compare and contrast that set of measurements having a template. There's usual variability if, for example, your face is tilted, you press one side of the finger more than another, or your speech is suffering from an infection. Variant reduces accuracy.

Biometrics can become a single point of failure. Consider a retail application in which a biometric recognition is certainly associated with a payment design: As one user puts it, "If my credit card fails to enroll, I can usually pull out another card, but if my fingerprint is not recognized, I've only that certain hand." Forgetting a password is a user's fault; failing biometric authentication is not.

Although equipment is improving, there are still incorrect readings. We content label a "false good" or "false accept" a studying that is recognized when it should be rejected (that's, the authenticator will not match) as well as a "false adverse" or "false reject" one which rejects when

it should accept. Often, reducing a false good rate increases fake negatives, and vice versa. The results for a incorrect negative are usually less than for any false positive, so an acceptable technique may have a false positive charge of 0.001 per-cent but a incorrect negative rate of just one 1 percent.

The speed of which a recognition must be done limits accuracy and reliability. We might ideally like to acquire several readings and merge the outcomes or measure the closest match. But authentication is performed to allow a user to accomplish something: Authentication isn't the end objective but a gate maintaining an individual from the goal. An individual understandably really wants to see through the gate and becomes frustrated and annoyed if authentication takes too long.

Although we prefer to think of biometrics as exceptional parts of an individual, forgeries are feasible. The most renowned example was an synthetic fingerprint produced by research workers in Japan . Although tricky and unusual, forgery will undoubtedly be an issue whenever the praise for a bogus positive is high enough.

Sometimes overlooked inside the authentication discussion is that credibility is a two-sided problem: The system needs guarantee that an individual is authentic, however the user desires that same guarantee about the method. This second concern has led to a new school of computer scams called phishing, where an unsuspecting customer submits sensitive info to a malicious program impersonating a trustworthy one. Common goals of phishing disorders are banks along with other financial institutions because fraudsters use the sensitive info they get from customers for taking customers' cash from the real institutions.

Authentication is vital for an operating system because accurate individual identification is the key to specific access rights. Just about all operating systems and computing program administrators have applied reasonable but stringent security steps to lock out illegal users before they can access system methods. often an inappropriate mechanism is pressured into use as an authentication device.

4.6 Review Question

1. Give an example of the usage of physical parting for security in a computing environment.
2. Give an example of the usage of temporal separation for security inside a computing environment.
3. Give an example of an thing whose security stage may transform during execution.
4. Respond to the allegation "A great operating system needs no protection for its executable program code (in memory) because that program code is a duplicate of code maintained on disk."
5. Explain what sort of fence register can be used for relocating a user's program.
6. Can any number of concurrent processes get protected in one another by just one couple of platform/bounds registers?
7. The talk of foundation/bounds registers means that program code can be execute-only and this data areas are usually read-write-only. Will be this ever false? Explain your solution.
8. A design employing tag parts presupposes that adjacent storage area locations hold dissimilar points: a type of code, a bit of data, a type of code, two bits of data, and so forth. Most programs do not look like that. How can tag bits turn out to be appropriate in a situation in which courses have the extra conventional set up of program code and data?
9. What are some other levels of safety that users should apply to program code or data, in addition to the common read, write, and execute agreement?
10. If two customers share access to a segment, they must do so by exactly the same name. Must their defense rights into it be the similar? Why or why not?
11. An issue with either segmented or paged target translation is timing. Assume a user wants to read some files from an input device into memory space. For proficiency during data transport, often the actual memory address at which the data should be placed is furnished to a I/O device. The real address is passed in order that time-consuming target translation does not have to be done during a extremely fast data move. What security difficulties does this approach bring?
12. A directory is also an subject to which entry should be handled. Exactly why is it not appropriate to allow users to change their own directories?
13. Why should the directory of one user not end up being generally accessible (for read-only gain access to) to other users?

14. Describe each of the following four kinds of access control mechanisms in terms of (a) ease of determining authorized accessibility during execution, (b) ease of adding access for a new subject, (c) simple deleting access by way of a subject, and (d) simple creating a fresh thing to which all themes by default have got access.

per-subject access command list (that's, one list for every subject tells all of the items to which that subject has admittance)

per-object access management list (that's, one list for every object tells all the subjects who have access to that thing)

access control matrix

capability

15. Assume a per-subject entry control list is used. Deleting an subject in such a system can be inconvenient because all modifications must be made to the control listings of all subject matter who did get access to the object. Recommend an alternative, less expensive means of controlling deletion.

16. File gain access to control relates mainly for the secrecy aspect of security. What's the relationship between an accessibility control matrix plus the integrity of the items to which entry is being operated?

17. One characteristic of an capability-based protection method is the potential of one method to move a copy of your capability to another process. Describe a situation in which one process can transfer a capacity to another.

18. Describe a system by which an operating system can enforce limited transfer of functions. That is, procedure A might send a capability to method B, but A wants to stop B from transferring the capability to any other processes.

Your design should include a explanation of the actions to be carried out by A and B, as well as the activities conducted by and the information maintained because of the operating system.

19. Listing two disadvantages of using real separation in a computing system. Record two drawbacks of making use of temporal separation in the computing system.

20. Explain why asynchronous I/O activity is a difficulty with many memory space protection schemes, like basic/bounds and paging. Suggest a solution to the issue.

21. Suggest an efficient scheme for sustaining a per-user defense scheme. That is, the system keeps one website directory per user, and that directory lists all the objects to that your user is authorized access. Your design and style should address the needs of a system with 1000

customers, of whom only 20 are lively at any time. Each user has an regular of 200 permitted objects; you can find 50,000 full objects in the system.

4.7 References

1. Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger - RAND Corporation Publisher: Prentice Hall
2. Cryptography and Network Security - Principles and Practice fifth edition Stallings William Publisher: Pearson
3. Cryptography And Network Security 3rd Edition behrouz a forouzan and debdeep mukhopadhyay 3/E Publisher: McGraw Hill Education
4. Cryptography and Network Security, 3e Atul Kahate Publisher: McGraw Hill

Chapter 5. Designing Trusted Operating Systems

5.0 Introduction

5.1. What Is a Trusted System?

5.2. Security Policies

5.3. Models of Security

5.4. Trusted Operating System Design

5.5. Assurance in Trusted Operating Systems

5.6 Review Question

5.7 References

5.0 Introduction

In this particular chapter

What makes a great operating system "secure"? Or perhaps "trustworthy"?

How are respected systems designed, and which in turn of those design rules carry over naturally in order to other program development responsibilities?

How do we produce "assurance" of the correctness of any trusted operating program?

Operating systems are the main security suppliers of computer systems. they support several programming capabilities, allow multiprogramming and resource sharing and implement system and user behavior restrictions. Because they have that control, operating systems are also targets for attack, since breaking through an operating system's defenses allows access to computer system secrets.

From a user perspective, we looked at the operating system to determine what general operating systems offer basic security services. Such four services have been studied:

- memory protection
- file protection
- general object access control
- user authentication

They conclude that an operating system is trusted if we have confidence in the consistent and efficient delivery of these four services. We will look at a trustworthy operating system in terms of the design and function of safety services components. The four first parts of this chapter are the four core pillars of a trustworthy operating system:

Policy. That system can be defined by its specifications: what the system ought to do and how it ought to do it. The safety requirements of an operating system are a collection of clearly and unambiguously defined, valid, and implementable rules. To order to comply with these specifications, the operating system follows the user's requirements. The OS usually follows a specified security policy, which is a set of rules that sets out what to protect and why, to ensure that the requirement is simple, consistent and efficient.

Model. The developers must be sure that the proposed system meets its requirements while safeguarding appropriate objects and relationships to establish a trustworthy operating system. We usually start by creating an environment model that can be protected. In reality, the model reflects the policy implemented by the operating system. And make sure the overall system functions are not affected or weakened by security requirements, engineers equate the model with the system requirements. They will then research different ways to enforce this protection.

Design. Designers choose a way to enforce it after the use of a security model. The design, therefore, covers both the confidence of the operating system (that is, its intended functionality) and its structure (its implementation).

Trust. Because the OS plays a major role in security compliance, we (as developers and users) need a reason (assurance) to assume that it meets our standards. Our trust in the system is based on two factors, namely features (the operating system has all the features required to implement the expected security policy) and assurance (the operating system was designed to make sure that the security policy is properly and efficiently enforced). We should address in the fourth part of this segment what makes a specific design or implementation trustworthy.

5.1. What Is a Trusted System?

Let us look at the terms used in trust understanding and explaining trust carefully before discussing a trustworthy operating system in detail. What is it that would take to be secure for us? The term "security" represents a dichotomy: something is protected or not safe. If we claim to be safe, you either support our statement or you deny it (or don't use or use it but don't trust it). How does security differ from quality? When we say something is fine, you are less interested in our arguments and more interested in an objective assessment as to whether it meets your requirements for efficiency and functionality. Security is only one element of excellence or consistency in that context; you might choose a system that is best, in the context from your choices, through integrating security with other attributes (such as speed or user-friendliness). In particular, the system you construct or choose might be safe as you would like to be, even though it may not be as secure.

If we are aware that the program has been intensively designed and evaluated, we conclude that the code is trusted which is reason to believe that the software does what is expected nothing more than that. In fact, trusted code can be the backbone upon which, untrustworthy, code functions. In other terms, the consistency of the untrustworthy system relies partially on the trusted code as well as the trustworthy code puts down the foundation of system security. The software can, in general, trust an operating system if there is a foundation to trust that access to components and systems running from it is properly controlled.

We depend on rigorous analysis and tests to find certain key characteristics in order to trust each program:

Functional correctness. The program does whatever it should and works perfectly.

Enforcement of integrity. Even if incorrect commands or instructions through illegitimate users are provided, the software retains the integrity of the data.

Limited privilege: The program has secure data, but access is reduced but neither access rights nor data are transferred to or returned to other untrustworthy programs.

Appropriate confidence level. A level of trust suitable for the types of data and context wherein the program has tested and scored has been created.

A trustworthy program is frequently used as a safe means of accessing confidential data for ordinary users. Trustworthy programs, without letting users exclusive access to critical data, are being used to conduct minimal (secure) tasks for users

Rather than secure operating systems, security experts prefer to speak regarding trustworthiness. A secure system achieves its safety standards, is reliable enough to support the user's trust in that system. In other words, the user or application of the system not their developer designer or manufacturer recognizes trust.

Secure	Trusted
.Either-or: Either something is secure or not	Graded: "Trustworthiness" levels are eligible
Property of presenter	Property of receiver
Claimed on the basis of product features	Determined by proof and examination
Absolute: not how, where, when, or by whom it is used	Relative: seen in use context
A goal	A characteristic

Table 5.1. Qualities of Security and Trustedness

Above Table taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Instead of secure operating systems, security experts prefer to talk of trustworthy. A trustworthy system satisfies the safety standards, is of sufficient quality and demonstrates the user's trust in this quality product In other words, trust is interpreted by the receiver or user of the system not its developer designer or manufacturer. You may not explicitly measure this trust as an user You may depend on the trust, an evaluation or a respected colleague's views. But in the end, the level of confidence you need must be approved.

It is important to understand that there may be trust; trust is not contradiction, unlike security. For instance, with fundamental secrets, you trust other friends, but some others only believe you to provide you daytime. Trust, according to proof and knowledge, is a characteristic that often develops over time. Securities with high-confidence records for example, can borrow the money when lenders repay loans as they plan. In the end, trust is earned, not claimed or conferred.

The trustworthy term is often used in this chapter as a trustworthiness procedure (a process that can impact system security or process whose inappropriate or deceptive operation breaches the system security policy), trustworthy software (an examined and authorised product), trustworthy software (the software part of a system used to implement security policies), The trusted computer base (set of any computing system protection systems that, together, implement a common security policy across a product or system, including hardware, firmware, and software) or a trustworthy system (a system that employs sufficient hardware and software integrity measures to enable its use to process sensitive information).sufficiency of actions and mechanisms.

The principles of are similar to these definitions

- Security policy enforcement
- Adequate steps and processes
- evaluation

In studying respected os's, we examine tightly why is trustworthy.

5.2. Security Policies

They must be able to define the security policies in order to ensure how an operating system provides the security that they expect. A security policy is a security assurance which we intend the system to implement Only in association with its security policy could an operating system (or another trustworthy system) be trusted; the system is set to meet its security requirements.

We start our security policy review through investigating national defence policy, as it was the foundation of a trusted creation of operating systems and was relatively straightforward to state. After which we migrate to security policies that can be introduced by commercial institutions.

Military Security Policy

The Military Security Policy is based on the confidentiality security of classified information.

•Each information piece is ranked at a particular level of sensitivity such as :

–unclassified

- restricted
- confidential
- secret
- top secret

The ranks or levels construct a hierarchy which, as shown in figure 5-1, represent an increasing level of sensitivity. That is, at a specific level information is more sensitive than at a lower level and less sensitive than at a higher level. Limited information, for example, is more sensitive than unclassified, but less sensitive than confidential. The sensitivity of an object O can be described by rankO.

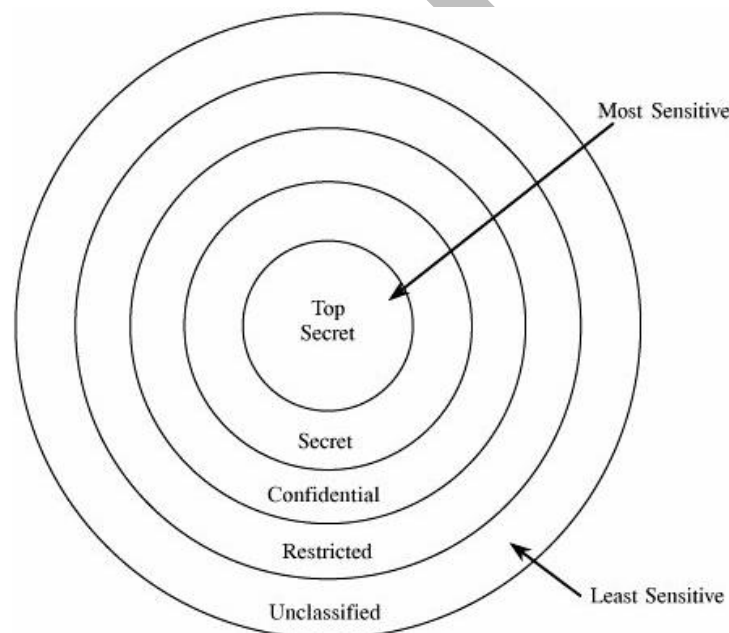


Figure 5.1. Hierarchy of Sensitivities.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The need to know rules restrict the access to the information: only subjects who need to know the data to do their job are given access to sensitive data. Each piece of classified data may be linked to one or more projects that are called compartments that describe the information subject. The Alpha Project, for instance, can use secret information like the beta project, but alpha personnel do not have access to beta information. In other words, both projects use secret information, but only secret information is needed for each one. Compartments thus help enforce limits on need for information in order to allow people to only access information related to their work. A compartment may contain only one level of sensitivity or may contain information on multiple levels of sensitivity. Figure 5-2 shows the relationship between the compartments and the level of sensitivity.

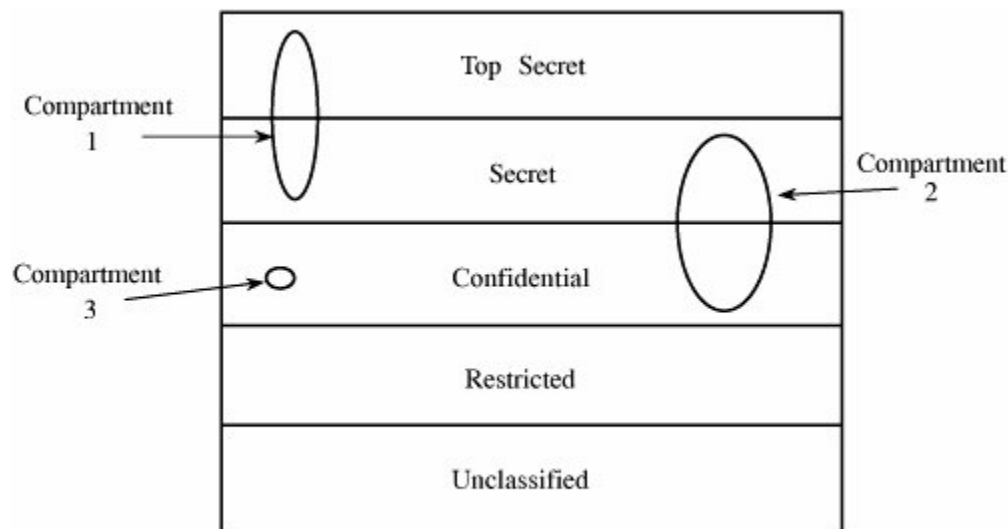


Figure 5.2. Compartments and Sensitivity Levels.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

One (or more) projects, called compartments, that relate to each piece of information. • The $\langle \text{rank}; \text{compartments} \rangle$ combination is called an information piece's classification or class.

Person seeking access must be cleared of sensitive data. A clearance is an example of a person having trust in getting access to information at a certain sensitive level and that an individual needs to know certain sensitive information categories. The clearance is an $\langle \text{rank}; \text{compartments} \rangle$ combination of a subject. This combination is the same way as a piece of information is classified.

Now we have a tendency to introduce a relation \leq known as dominance, on the sets of sensitive objects and subjects. For an subject s and an object o ,

$$s \leq o \text{ if and only if } \begin{aligned} &rank_s \leq rank_o \text{ and} \\ &compartments_s \subseteq compartments_o \end{aligned}$$

We say that o dominates s (or s is dominated by o) if $s \leq o$; the relation \geq is the opposite. Dominance is being used to restrict sensitivity and information content which can be accessed by a subject. A subject could only read a subject if the level of clarity of the subject is at least as high as the level of the information and the subject has a need to learn about all the compartments in which the information is classified.

Sensitiveness and information-based standards are both imposed by military security. Sensitivity needs are known as hierarchical requirements since they are representative of the hierarchy of sensitivity levels; need-to-know constraints do not necessarily reflect a hierarchical structure in compartments. The combination model is ideal for an environment in which a central authority strictly regulates the access. Someone, often referred to as a security officer, controls clearances and classifications that do not normally require people to modify.

Commercial Security Policies

Commercial enterprises have major safety issues. They care about the fact that industrial spy would provide competitors with information about the new emerging products. Moreover, companies also look forward to securing corporate finance details. Although the corporate world is inherently less bureaucratic and less bureaucratically organized than the military, in corporate security policies we also see many of the same ideas. For example, a major company could be split into groups or departments, every accountable for a number of disparate projects, such as a corporation or university. Other administrative tasks, including accounting and personnel activities, can also exist. Data items at each point may have various sensitivity levels such as the public the holders or the internal; in this situation, the names that vary between organizations and there is no standardized hierarchy.

Suppose personal information is less vulnerable than private information, which, in effect, is less vulnerable than internal information. Projects and departments are typically quite well-differentiated, some of which overlap as people work on two or more projects. Corporate roles continue to cross projects and departments as individuals might need accounting or private data across the organization. Nonetheless, even company data can also be critical. Projects could also be vulnerable: Outdated-standby project workers do not need to know about new-product projects, whereas new-product staff could have access to all outdated-stand-by data.

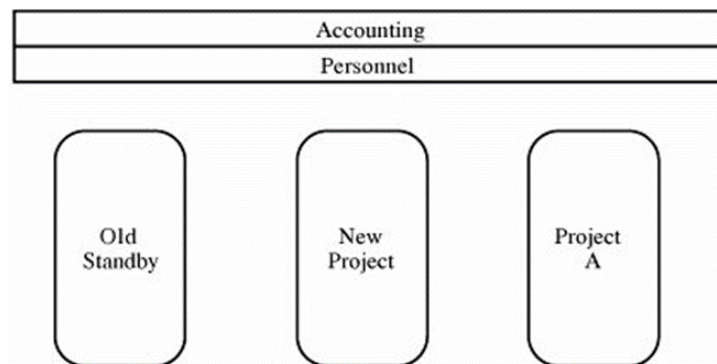


Figure 5.4. Commercial View of Sensitive Information

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

There are two substantial differences between the protection of commerce and military information. Firstly: a person operating on a private project doesn't need the permission of a central security officer for the MARS project accessibility, usually without the military permission. Useful access to proprietary data usually may not grant an employee another degree of privacy. Secondly, the guidelines for access are less standardised since there is no formal approval concept. For instance, if the senior executive determines that the person deserves access to some MARS internal information, the executive orders somebody to give access either once or continuously. Therefore, most corporate access to information is not regulated by the structured definition of commercial approval.

To date, many of our discussions have only concentrated on read access which discusses security confidentiality. That simplistic view probably applies to most of the computer security research still in progress. In many cases, though, integrity and availability are at least as effective as confidentiality. Integrity and availability regulations in both commercial and military sectors are significantly less well developed than those for confidentiality. We discuss certain examples of integrity concerns in the following two examples.

ClarkWilson Commercial Security Policy

In 1987, David Clark and David Wilson built a completely different Integrity Model from earlier models. This model uses transactions as the basic operation which would more accurately model most commercial systems than previous models. As discussed earlier, one of the primary concerns of a commercial environment is the quality of the data in the system and of the actions carried out on that data. The data is said to be in a stable (or consistent) state if it satisfies the properties in question. For example, let D be the amount of money deposited so far today, W the amount of money withdrawn so far today, YB the amount of money at the end of yesterday in all accounts, and TB the amount of money in all accounts so far today. Then the consistency property is

$$D + YB - W = TB$$

The conditions of continuity must be fulfilled before and after each operation. A well-formed transaction is a sequence of transactions that change the system from one coherent state to another. For instance, if a depositor transfers money from one account to another, the transaction is the transfer; this transaction is made up of two operations: a first deduction and an addition to the second account. The activity may leave data inconsistently, but it must maintain consistency with the well-formed transaction.

The integrity of the transactions itself is the second characteristic of a commercial context relevant to an integrity policy. What tests and approves the right conduct of transactions? For instance the procurement company needs multiple steps to pay for an invoice if a company receives the invoice. First, somebody had to apply for a service and created the account that would pay for the service. The invoice has to be checked by someone (was it really paid for?). You must debit your account approved to pay for the service, and write and sign the check. If one individual takes these

steps, that individual may pay fake invoices easily, but both must collude in defrauding the company if at least two different individuals conduct such steps. The requirement that more than one person be involved in this process is an example of the separation of duties concept.

There is no distinction in computer-based transactions. Someone has to ensure the appropriate implementation of transactions. The concept of division of duties takes a different person for the certifier and implementers. In order to change the data, two different people must make similar mistakes or coordination with each other to validate the well-formed transaction as right in order to be able to modify the data or leave the data in an inconsistent state.

The model

To order to ensure confidentiality, the Clark-Wilson model describes data items as restricted data objects or CDIs. Unconstrained data items or UDIs are referred to as data that are not subject to the integrity controls. The balance of accounts in a bank is, for example, CDI since its integrity is vital to the bank's operation while the gifts that the account holders select when their banks are opened are UDI, because their integrity does not have a decisive role in the bank's activity. The CDI set and the UDI partition set are the set of data modeled within the device.

The values of CDIs are restricted by a number of integrity constraints (simulated in spirit to the constraints mentioned above). The consistency constraint previously raised in the bank example would also be a constraint of integrity. Two sets of procedures are also included in the model Integrity check procedures or IVPs test that the CDIs comply with the integrity restrictions when the IVPs run. The system is apparently in a good state in this situation. Procedures of transformation or TPs, modify data status in the system between legitimate and other state; TPs carry out well-formed transactions.

Back to the bank account instance. The accounts balance are CDIs; as mentioned above it is an IVP to ensure that the accounts are balanced. Cash deposit, cash withdrawal, and money transfer between accounts are TPs. A bank investigator should certify that the bank uses the correct procedures to verify that the accounts are balancing, to deposit cash, to withdraw money and to transfers of money to ensure adequate account administration. In fact, these proceedings can only cover savings and checking accounts; they may not extend to other types of accounts, such as small cash. Within two certification rules, the Clark-Wilson model contains these requirements:

Certification rule 1 (CR1): When any IVP is running, all CDIs must be in a valid condition.

Certification rule 2 (CR2): A TP will convert these CDIs into a valid (possibly different) state for some related sets of CDIs.

CR2 defines a connection which links a collection of CDIs with a specific TP as certified.

$$(\text{balance}, \text{account1}), (\text{balance}, \text{account2}), \dots, (\text{balance}, \text{accountn}) \in C$$

CR2 means that if a TP is not certified to work with that CDI, a CDI may become corrupted. For example, if the TP investment in the stock portfolio of the Bank is allowed to work on the fund, the TP would corrupt the balance of the bank's assets, because the activities of the TP do not make sense on the bank account. The system must therefore prevent TPs from operating on CDIs not certified for.

Chinese Wall Security Policy

The Chinese Wall model is a security policy model that equally refers to confidentiality and integrity. It describes policies which involve business conflicts of interest and is as important for situations like the military model of Bell-LaPadula. For example, British law requires the application of a policy similar to that and the proper implementation, in cases involving certain criminal charges, of parts of the model provides the defense. This model is the natural environment for a stock exchange or trading house. The security insurance plan develops on three degrees of abstraction. The purpose of the model here is to avoid a conflict of interest between the two customers served by the trader and the best interests of the dispute between clients, so that the trader can contribute to one profit, at the cost of the other.

Informal Description

Consider taking the investment house database into account. This consists of investment records of companies and other data likely to be sought by investors. These records are used by analysts to direct both the business and the individual's investments. Suppose that in their savings Rajesh counsels the Bank of India.

When Citibank is also endorsed, he may have an interest conflict since the investments of both banks can clash. Rajesh can not therefore advise all banks

The following definitions capture this:

The database objects are company-related pieces of information.

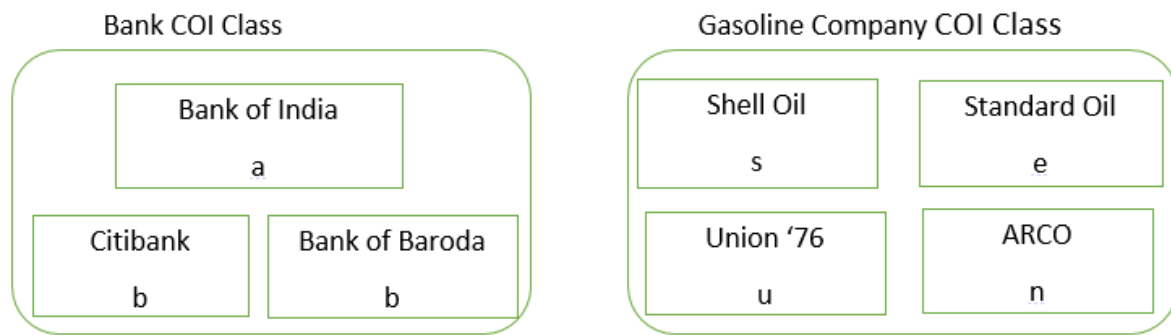
An company data set (CD) includes objects related to a single company

Both classes of items are grouped at the highest level for competing companies.

Let COI(O) represent an object O COI class, allowing CD(O) to be an object O company dataset. Each object belongs to the same COI class, the model assumes.

The items in the Bank of India CD are available to Rajesh. Due to the fact that the CD at Citibank is in the same COI class as the CD at Bank of India, Rajesh can not get the object on the CD at Citibank. This database structure provides the necessary skills

Figure : The Chinese Wall model database. There are two groups of COI. Three CDs are included in the Banks ' collection. The other one includes four CDs for gas companies. A lowercase letter (for instance, (Bank COI, Citibank) c) is represented for each pair (COI, CD). In each COI, Rita can access no more than one CD, so that it can access the CD and ARCO, but not the CD and Bank of India's CD of Citibank.



This involves an element of time. Suppose Rajesh served on the portfolio of the Bank of India, and then was moved to the portfolio of Citibank. While he is just working on a CD in the COI class, most of the information that he has learnt from the portfolio of the Bank of India is up to date. It can also direct Citibank's investments by Bank of India information— a conflict of interest. This leads to a law where $PR(S)$ is the set of objects interpreted by S .

CW-Simple Security Condition, Preliminary Version : S can read O only if one of these is valid.

1. Object O' has accessed S O' and $CD(O')=CD(O)$. Object O' is there.
2. For all objects O' , $O' \in PR(S) \Rightarrow COI(O') \neq COI(O)$.

Initially, $PR(S) = \emptyset$,

In the above case, both the Bank of India's CD and ARCO CD are able for Rajesh to read objects Condition 1 is therefore fulfilled. Unless, however, Bank of India's CD involves unauthenticated objects (a fair assumption), then condition 2 is incorrect as Rajesh can read those objects. Rajesh can therefore not write to ARCO's CD objects

5.3. Models of Security

Models always use to define, study or analyze a specific situation or relationship in security . McLean gives a good insight into security models. Security models in particular are used

check the completeness and consistency of a specific policy

Policy document

support to plan and conceptualize implementation

Check if an application complies with its requirements

We presume that a single user is able to gain access a certain object under a certain access control policy. We always take this approach for granted outside of any plan. This is to say, a policy decision defines whether a particular individual should have access to a particular object; the model

is a policy-enforcing mechanism alone. Therefore, we continue to study models with easy ways to control user access.

Multilevel Security

Theoretically we would like to construct a model which will represent a number of sensitivities and the importance of rigorously separating subjects from object they should not have access to. Consider, for example, the choice and sensitivity of the data in the voting process. Perhaps not sensitive are the name of the candidates. The names of the winner are kind of sensitive if the results have not been announced. Whether the number of votes earned by one candidate is embarrassingly small, the vote count may be more sensitive Finally, it is highly sensitive for a single person to vote. The sensitivity of the information that users can access can also be rated.

The army has created comprehensive methods of information security for evident purposes. As a data security standard in an operating system, a generalization of the army model of data security has also been adopted. The features of the military model were first defined by Bell and La Padula in mathematics, and Denning initially formalized the structure of this model. In 2005, Bell restored its contribution to computer security on the original model. He noted that the model demonstrated that before the system was designed, security had to be recognized and not included in the program. Developing and creating a security toolbox to secure yourself. The simplified model is referred to as the security's lattice model, as its mathematical structure are called a lattice.

A more general scheme called a lattice, is the military security model In the military model, the dominant relation \leq is described as the relationship for the lattice. The relationship \leq is antisymmetric and transitive. The biggest component of the lattice is the classification < top secret; ; all compartments >, the smallest element is <unclassified; no compartments >; these two elements dominate and are dominated by all elements. The military model is therefore a lattice.

The lattices are many other structures. For instance, in the natural order of public data that are less than proprietary and less vulnerable than internal data, a commercial security policy can include sensitivities data such as public, proprietary and internal. Such three levels are also a grid.

Security experts have opted to base security systems on a lattice because of its increasing degrees of course. In a military context, a security system built to incorporate lattice models can be used. It can also be used for sensitivity levels in commercial contexts with different labels. Therefore, lattice representation of awareness levels pertains to many computing circumstances.

What Is a Lattice?

The LBAC (Lattice-based access control) computer security system is a complicated method of controlling access to information based on any combination of objects (e.g. resources, computers and application programs) and topics (e.g. people, groups or organisations)

A lattice is used in this type of control model to describe the security levels an object may have and for which a subject may access. This is to suggest that at the security levels a partial order is defined so that each two security levels have a lower (meet) and a lower (join) lower bound at all times. Once two objects A and B are joined together as another object C, the level of security is

allocated to the object as the "join" of A and B levels. When two subjects have to access a certain secure data together, the level of access of that object is specified as the "meet" of the level of the subjects.

A lattice is a math structure of elements ordered by the connection of a relational operator between them. We use the notation \leq to indicate this relation and state that $b \geq a$ is the same as $a \leq b$. A relationship is called a partial ordering if it is transitive as well as antisymmetric. For all three elements a, b, and c, these terms mean that there are two rules:

transitive: In case $a \leq b$ and $b \leq c$, $a \leq c$

antisymmetric: In case $a \leq b$ and $b \leq a$, $a = b$

Not every pair of element should be equal in a lattice; that is, elements (a) and (b) for which neither $a \geq b$ nor $b \leq a$ are valid. Nevertheless, each pair of elements has a upper bound which is an element that is at least as large as (\geq) both a and b. In other words While a and b may not be compared under \leq , a upper bound element u such that $a \leq u$ and $b \leq u$ exist in a lattice. -pair of elements in a lattice has a lower bound, an element l that is both a and b, that is, $l \leq a$, and $l \leq b$.

Consider Figure 5 as a lattice that displays all of the numbers 60. The relation operator is the relationship of "is a factor of." Thus, the notation $a \leq b$ implies that a divide (b) or, equivalently, b is a multiple of a. The lattice indicates that the 60 is a dominant factor for the other elements;; 12 predominates 4, 6, 2, 3, and 1; 20 dominates 4, 10, and 5; and so on. We can also see that other elements are unlike them. For example, 2 and 5 are not comparable and are therefore not directly linked in the diagram by lines.

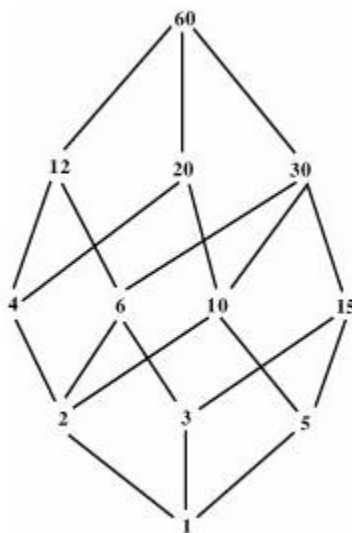


Figure 5.6. Sample Lattice.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Lattices are useful for the representation of relationships, and they most often occur when a relation shows a difference in power, substance or value. Nonetheless, many typical relationships only form half a lattice. There is a particular upper bound (for example, a common ancestor) but not the greater lower bound for each pair in relationships. "less than," is a subset of 'or' reports to (for employees) 'or' is a descendant of '.

BellLa Padula Confidentiality Model

The Bell La Padula concept fits the classification scheme of the military style to distinguish secure and non secure states

It incorporates mandatory (compulsory system-based) and discretionary controls (user-set) of access. If there was no compulsory control S would be able to read O if it was discretionary read / write access to O means.

Characteristics of the Model

1. The model includes a set of subject S security clearance levels and Object set O security classification.
2. Let $L(s) = l_s$ be subjects s level security clearance and $L(O) = l_o$ be object o security classification, and only if $l_o \leq l_s$ and s have discretionary read access to o . (simple security condition)
3. S can write O only if and only when $l_s \leq l_o$ and s has access to o (*-property).
4. Let Σ be an secure initial state Q_0 program, and T be a series of transformations. When each T element retains the simple security and *-property condition, then every Q_i state is secure

This property states in the military model that the contents of a critical object are at least as high as possible.

One of the meanings of the *-property in the military model is that the information can be passed on by an individual on some level only to others not less than the information level. The *-property avoids write-down, when a subject with high-level data access transfers the data to a low-level entity.

Basically, the *-property needs that a person receiving information on an individual level does not talk to people at levels less than the weather level! This example shows that this property is stronger than required for security but in computer systems the same goes for it. The concept of BellLa Padula is highly conservative: even at the expense of usability or other properties it ensures protection

Biba Integrity Model

Biba's model addresses integrity issues i.e. the likelihood of corruption of information. For gauge integrity a new label is used. If a high-security object comes in contact with information at a low

level or is handled by a low-level system, the level of integrity can be decreased. For example, the software could hiddenly copy a secure document to another part of the systems if you used an insecure program for viewing a secure document.

Integrity is generally characterised by a three following goals: that is

The data is protected from unauthorized user modification

Data are protected by authorized users from unauthorized modification (which raises the issue of unauthorized modification; e.g. logs are removed or altered records whereas adding records is permitted)

The data is consistent internally and externally. Once, an example of log integrity

The Biba model (also referred to as the dual) is the equivalent of the BellLa Padula model. Biba describes "integrity levels," similar to the BellLa Padula model sensitivity levels. Objects and objects denoted $I(s)$ and $I(o)$, are sorted by a scheme of integrity classification. The apps are

Simple Integrity Property. Subject s can alter (writing access to) object only if $I(s) \geq I(o)$ object

Integrity *-Property . If subject s reads the object at integrity level $I(o)$, s can only write the object p if $I(o) \geq I(P)$ has read the object.

Both these rules obviously cover unreliable information. Assume that Rajesh is sometimes considered to be dishonest. If Rajesh is able to create or modify a document, people in this document may doubt the validity of the statements Thus the integrity of this object is decreased by an unreliable person with written access to an object. Similarly, a report based on unsound evidence is quite reasonably suspicious. The low integrity of a source object means that every object centered on the source object has a low integrity.

Graham Denning Model

The idea of a formal system of protection rules was introduced in Lampson and Graham and Denning. A model with generic protections was constructed between Graham and Denning. Two later security system models will be based on this model.

A collection of S objects, O objects, R rights and A access control matrix function on the Grahamdenning model. Each subject and object has one row and one column in the matrix. The contents of an element of the matrix indicate the rights of the subject to another subject or object Each object shall hold special rights to a subject named "owner;" a further subject designated as a "controller" shall hold special rights for each subject.

The Graham-Denning model deals with some of these issues and lays out a collection of fundamental rights in commands a particular subject may perform on an object This model has eight basic rights of protection or rules on how such functionalities should be performed securely:.

Create object Allows the subject command to enter a new system object.

Create subject, delete object, and delete subject The effect of the creation or destruction of a subject or object is similar.

Read access right Enables a subject to determine a subject to an object's current access rights.

Grant access right Allows an object's owner to convey to another subject certain access rights for an object.

Delete access right Enables a subject to delete a right to an object for another subject, if that delete subject is either the object's owner or the subject from whom the access is supposed to be deleted controls.

Transfer access right Enable a subject to transfer to another subject one of its rights to an object Every right may or may not be transferable. If a subject is given a transferable right, that entitlement (or not transferable) can be transferred to other subjects by the subject. Where a subject is granted a non-transferable right, the right may be used but may not be transferred to other subjects.

These rules happen to be shown in Table 5.2 which ultimately shows prerequisite situations for performing each command and its own effect. The accessibility control matrix is really a $[s,o]$, where s is really a area of interest and o can be an object. The topic executing each control is certainly denoted x . A transferable best suited will be denoted r^* ; a nontransferable ideal is prepared r .

Table 5.2. Protection System Commands.

Command	Precondition	Effect
Create object o		Add column for o in A ; place owner in $A[x,o]$
Create subject s		Add row for s in A ; place control in $A[x,s]$
Delete object o	Owner in $A[x,o]$	Delete column o
Delete subject s	Control in $A[x,s]$	Delete row s
Read access right of s on o	Control in $A[x,s]$ or owner in $A[x,o]$	Copy $A[s,o]$ to x
Delete access right r of s on o	Control in $A[x,s]$ or owner in $A[x,o]$	Remove r from $A[s,o]$
Grant access right r to s on o	Owner in $A[x,o]$	Add r to $A[s,o]$
Transfer access right r or r^* to s on o	r^* in $A[x,o]$	Add r or r^* to $A[s,o]$

Above Table taken from the book **Security in Computing, Fourth Edition** By **Charles P. Pfleeger** - Pfleeger Consulting Group, Shari Lawrence Pfleeger

This set of rules provides the necessary characteristics to model the security system's access control mechanisms. This process may be, for instance, a benchmark or a sharing system between two mutually untrustworthy subsystems.

HarrisonRuzzoUllman Results

The variant on the GrahamDenning model was suggested by Harrison, Ruzzo, and Ullman. A few questions concerning the security of a given system were addressed by this revised model. Suppose you're going to use a specific operating system and want to know whether a certain user can ever have a certain type of access. For instance, in Windows or MVS you may be setting security levels. You set the access controls and inquire whether user X will ever be able to access object Y. The three scientists have developed their model so that we can answer questions like this.

The HarrisonRuzzoUllman model (the HRU model) is built on commands that involve primitive operations and conditions for each command. The command structure is like this.

```
command name( $o_1, o_2, \dots, o_k$ )  
  if       $r_1$  in  $A[s_1, o_1]$  and  
           $r_2$  in  $A[s_2, o_2]$  and  
          ...  
           $r_m$  in  $A[s_m, o_m]$   
  then  
     $op_1$   
     $op_2$   
    ...  
     $op_n$   
end
```

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

This command is designed as a procedure with o_1 to o_k parameters. The HRU model's notation differs slightly from the GrahamDenning model, and each subject in HRU is also a subject. Therefore, all subjects and objects not subjects are the columns of the access control matrix. Therefore, all command parameters are called o although they can either be objects or objects that are not subject to it. As in the GrahamDenning model, every r is a generic right. Every op operation is a primitive, described in the list below. The access matrix is shown in Table 5.3.

Table 5.3. Access Matrix in HRU Model.

Subjects	Objects					
	S ₁	S ₂	S ₃	O ₁	O ₂	O ₃
S ₁	Control	Own, Suspend, Resume		Own	Own	Read, Propagate
S ₂		Control			Extend	Own
S ₃			Control	Read, Write	Write	Read

Above Table taken from the book **Security in Computing, Fourth Edition** By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

According to the GrahamDenning model, the basic operations op:

- create subject s
- create object o
- destroy subject s
- destroy object o
- enter right r into A[s,o]
- delete right r from A[s,o]

Such operations are defined by their names. There are a variety of subjects, objects rights and commands in the protection system.

Harrison et al. show that these operations are appropriate for modeling a number of examples of protection systems, including the Unix protection mechanism and the Graham and Denning indirect access mode. The HRU model can therefore reflect reasonable protection interpretations as with the GrahamDenning model.

Two important effects produced by Harrison et al. own key implications for makers of protection techniques.

The first derive from HRU implies that

In the modeling system where commands are limited to a single operation, it is possible to determine whether a specific object is ever allowed to obtain a particular right.

Therefore, we are able to decide (that's, we can learn beforehand) whether a low-level subject can ever get read usage of a high-level object, for example.

The second result is less encouraging. Harrison et al. show that

It's not always determined, if commands are not limited to one operation each, whether a certain protection system will grant a certain right.

Therefore, in general we can not decide whether a subject can obtain a specific right to an object.

Consider protection in the Unix OS as an example. It is relatively simple to use the Unix protection system, other protection systems are more complicated. As more than one operation per command in the HRU model is required for the Unix protection scheme, no general method can be applied to decide whether a particular access right can be granted to a subject.

The result of HRU is significant, but dreadful. Indeed, it is possible to extend the HRU result. An algorithm can be used to evaluate the access right question for a particular set of protection systems, but the correct access question for all protection systems can not be determined by even an infinite number of algorithms. The negative results however do not indicate that there is no protection system decision-making process. Indeed it is decidable whether a particular right of access can be granted on certain unique protection systems.

Therefore, in general procedures the HRU results are negative but do not preclude decision-making on specific protection systems.

TakeGrant Systems

The Take-Grant system is a model for assessing the rights of protection in a computer system, e.g. reading or writing. Jones, Lipton and Snyder developed Take-Grant system to demonstrate that the protection of a computer system can be determined even if there is a very large or unbound number of subjects or objects. It can be achieved in linear time depending on the system's initial size.

A protection system containing a variety of States and State transitions is based on the collection system. A directed graph shows the connections between the system nodes. Such nodes represent the subjects or objects of the model. The directed edges between the nodes are the rights of the connected node.

Denotation: (x,y) is the set of access rights on the edge from node x to node y . If r is an element of (x,y) , then node x has the right r for node y .

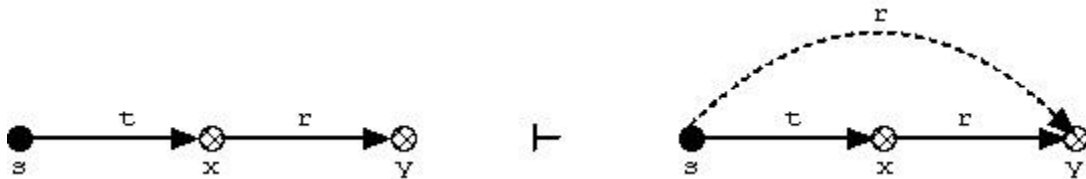
Take Right: The take right t on an edge is a special right. For a subject s to have the right t on an object x , it means that subject s can take any rights that x possesses.

Grant Right: Similar to the take right, the grant right g on an edge is also a special right. For a subject s to have the right g for a subject/object x , it means that subject s can grant (share) any of the rights it possesses to subject/object x .

State Transition Rules

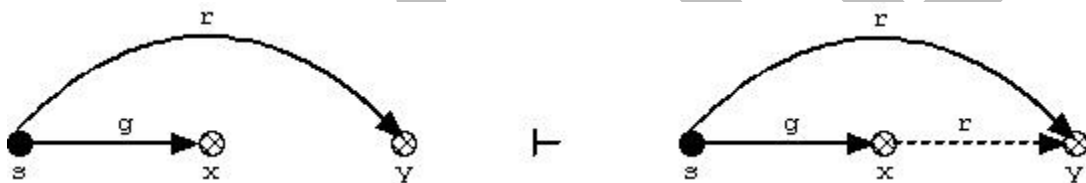
Rule 1 (Take):

Let s be a subject with t being an element of (s,x) and r being an element of (x,y) for a right r and the nodes x, y . To add r to (s,y) use: s **take** r for y from x . This is shown in the following picture where nodes x and y can be either subjects or objects.



Rule 2 (Grant):

Let s be a subject with g being an element of (s,x) and r being an element of (s,y) for a right r and the nodes x, y . To add r to (x,y) use: s **grant** r for y to x . This is shown in the following picture where nodes x and y can be either subjects or objects.



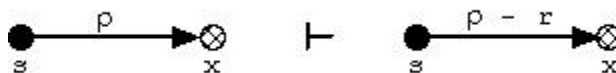
Rule 3 (Create):

If s is a subject and p is a set of rights, then the command: s **create** p for **new** {subject or object} x will add a new node x and sets $(s,x) = p$. This is shown in the following picture where node x can be either a subject or an object.



Rule 4 (Remove):

If s is a subject and x is a node, then the command: s **remove** r for x will remove the right r from (s,x) . This is shown in the following picture where node x can be either a subject or an object.



Summary of Models of Protection Systems

For two reasons, we are studying computer security models. First, models are necessary to determine the policies to be implemented by a secure system. The BellLa Padula and Biba models, for example, identify specific concepts we have to implement to guarantee confidentiality and integrity. The second thing is to understand the properties of protection systems by studying abstract models. The HRU model, for example, sets out characteristics that an arbitrary protection system can or can not agree on. For designers of protective systems, these characteristics are important to know.

5.4. Trusted Operating System Design

Operating systems are very difficult to design on their own (regardless of their security restrictions). You perform several tasks, are disrupted and contextually shifted and need to reduce overhead so that user computations and interactions are not delayed. The introduction of security enforcement responsibility into the operating system raises the complexity of developing an operating system considerably.

Trusted System Design Elements

Good design principles are always good for security. Nonetheless, some important design concepts are very important for security and are necessary for building a solid, secure operating system. Saltzer and Saltzer and Schroeder have clearly explained these concepts.

Least privilege. The fewest possible rights should be used by each individual and each program. It minimizes the damage caused by an accidental or malicious attack.

Economy of mechanism. The protection system architecture should be small, simple and straightforward. Such a system of security is analyzable, thoroughly tested, maybe reviewed and depended upon.

Open design. The security system must not rely on the ignorance of potential attackers; the mechanism should be available, based on the confidentiality of relatively few key items, such as a password table. The open design is also available for extensive public review and offers objective design security confirmation.

Complete mediation. It is necessary to check every attempt at access point. The system should be set up in order that it couldn't be bypassed. All attempts to direct access (requests) and attempts to bypass the access checking mechanism.

Permission based. A denying access is the default condition. A conservative designer defines what is to be accessible instead of what is not to be accessible.

Separation of privilege. User authentication plus a cryptographic key will preferably be based on multiple conditions. Thus, anyone who violates a security system won't have full access.

Least common mechanism. Shared structures deliver potential information for flow channels. Physical or logical separation systems reduce the sharing risk.

Ease of use. It is impossible to be avoided if a protective mechanism is simple to use.

Security Features of Ordinary Operating Systems

The operating system multi-programming carries out a variety of security related functions. Examine how Figure 5-10, which shows how an OS operates for users, offers services and resource allocation.

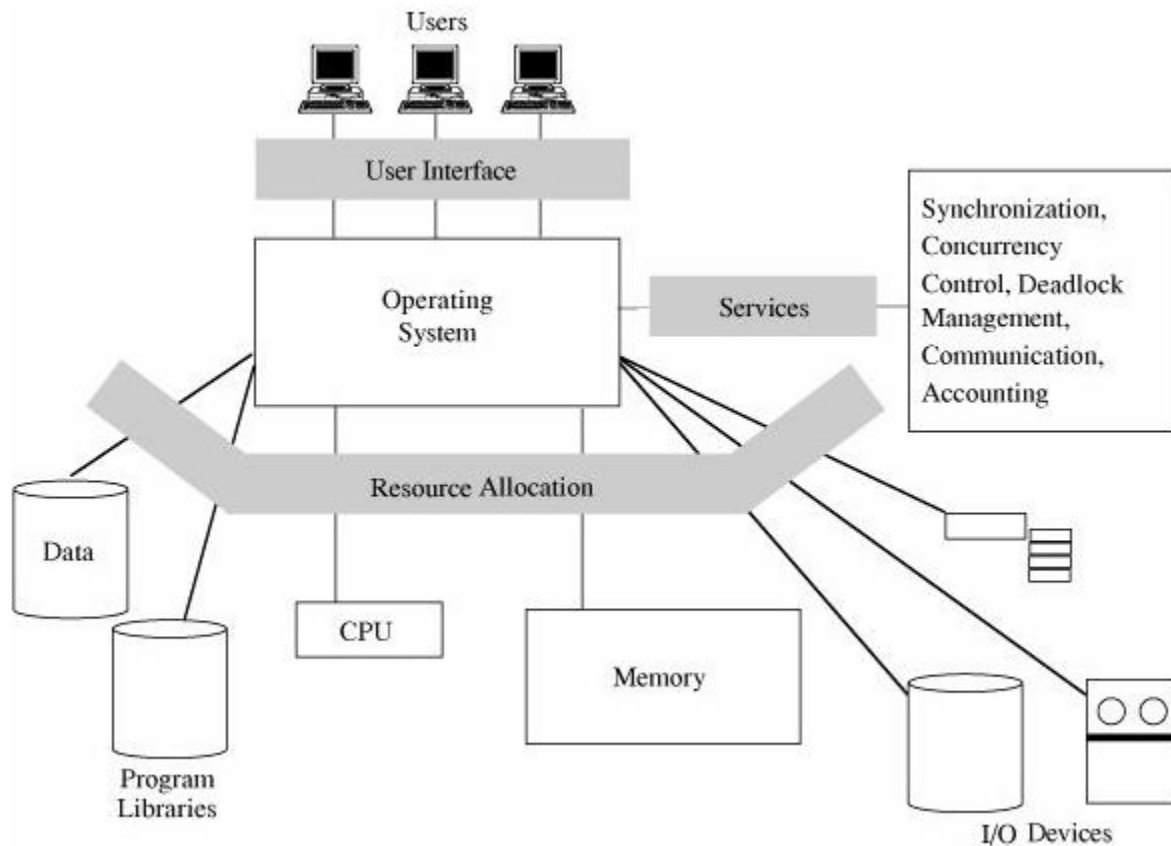


Figure 5.10. Overview of an Operating System's Functions.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

We can see that some different roles concerning computer security are handled by the system:

User authentication. The user granting access must be identified by the OS and the user must be sure that he or she is indeed the one to be. Password comparison is the most popular authentication method.

Memory protection. Each program user must run on an unauthorized access-protected portion of the memory. Such security can definitely prevent access by third parties and can also regulate the access of the user to a certain part of the program space. Variable security, such as reading, writing, and executing, can be applied to user memory space parts. Computer mechanisms such as paging or segmenting typically provide memory protection..

File and I/O device access control. The os will restrict unauthorized users from accessing user and system files. Such usage must also be protected of I / O devices. As with the access control matrix, data protection is usually accomplished through table lookup.

Allocation and gain access to control to basic objects. Users want specific artifacts, like constructs to allow concurrency and synchronization. However, it must be controlled to access certain objects, so that a user has no adverse influence on other users. Table lookup is once again the standard way to provide protection.

Enforced sharing. Users should have access to resources wherever appropriate. Sharing ensures that integrity and consistency must be assured. Table lookups are also used to allow managed sharing, coupled with integrity controls such as monitors or transaction processors.

Guaranteed fair services. Both users demand the use of CPUs and other resources such that no one is starving forever from receiving service. Hardware clocks are paired with fairness scheduling techniques. Control is paired with hardware and data tables.

Interprocess communication and synchronization. Execution processes can need to coordinate or synchronize access to shared resources with other processes. These services are provided by operating systems serving as a bridge between processes and reacting to the asynchronous process communication requests or to other processes. Communication between interprocesses is facilitated by access control tables

Protected operating-system protection data .In order to implement security, the operating system needs to keep data. Of course, the operating system is not able to implement such data if it is not protected from unauthorized access (read, modify, and delete). Different methods, including encryption, hardware control and isolation, enable operating system support data protection separation.

Security Features of Trusted Operating Systems

Contrary to regular operating systems, trusted systems use both features and protection technologies. The design of a trustworthy system is responsive and includes choosing an appropriate and compatible collection of features and ensuring that all features are controlled and properly implemented. Figure 5-11 indicates a distinction between a trusted and a common operating system. Figure 5-10 compare it. Note how the objects are preceded by or surrounded, with far more protection and separation than a typical operating system, by an access control system. Therefore, user-separated storage and databases of data and programs have operated sharing and parting.

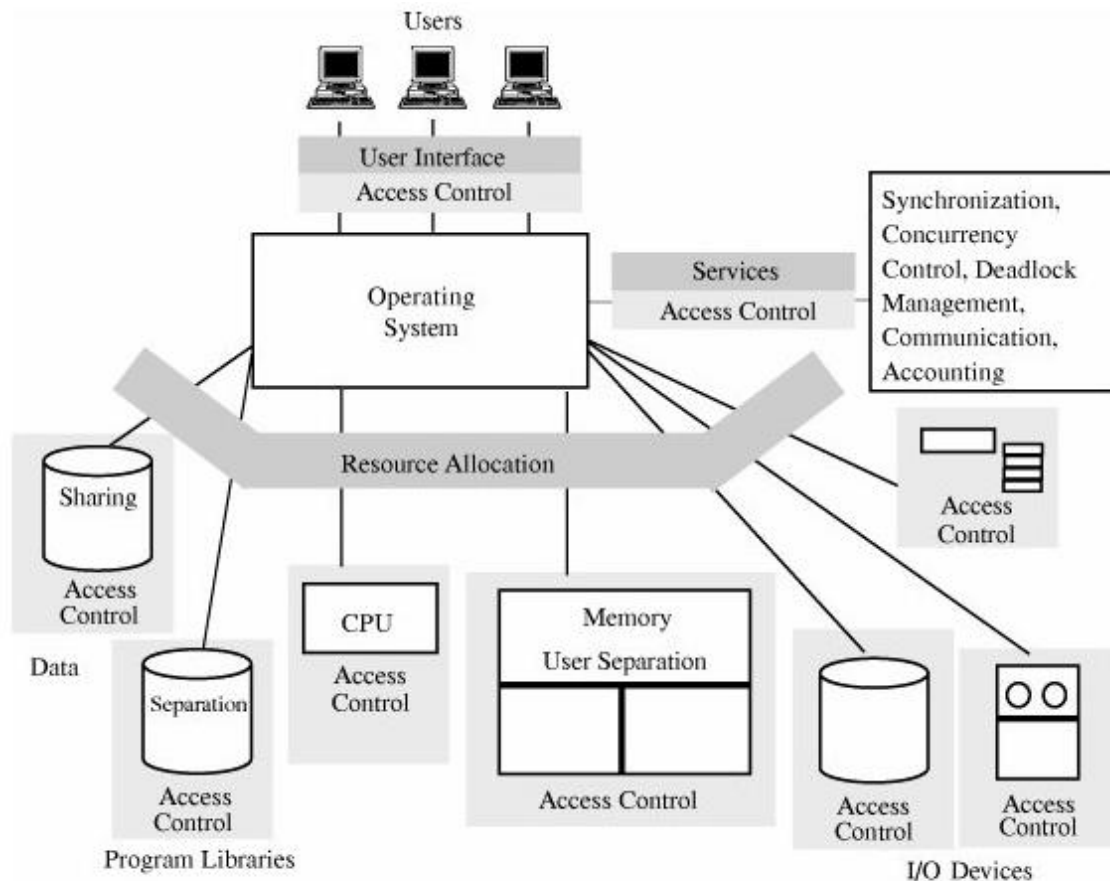


Figure 5.11. Security Functions of a Trusted Operating System.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

In this section, we consider in more detail the key features of a trusted operating system, including

- user identification and authentication
- mandatory access control
- discretionary access control
- object reuse protection
- complete mediation
- trusted path
- audit
- audit log reduction
- intrusion detection

We consider each of these features in turn.

Identification and Authentication

Most computer security is rooted in identification. We need to be able to know who wants access to an object and to check the identity of the subject. In the near future, the majority of access controls are based on correct identification, whether mandatory or discretionary.

Identification requires two steps: determining who all the access requestor is and verifying whether the requester is who it claims to be. That is, we want to have an identification that is authenticated or verified. Trusted operating systems need to be securely identified and people must be identified uniquely.

Mandatory and Discretionary Access Control

Mandatory access control (MAC) means that decisions on access control policies are created outside the control of an object's owner. A central authority decides who accesses information and can't change the user's access privileges. A military security example of a MAC takes place in situations where a single data owner does not determine who has the highest security clearance; nor can the owner adjust the category of an object from high secret to secret.

In comparison, The **discretionary access control (DAC)** leaves the owner or anyone else authorized to control the access to the object, as its name implies, a degree of access control. The owner may decide who should and should be allowed to have access to an object. Commercial environments typically use DACs to enable anyone and sometimes additionally named people to change access within a designated group. For example, a company can set up access controls so it can access the personnel files by the accounting group. The company can also allow Ram and Ganesh, as Directors of the Inspector General, to access these files. Access privileges to DACs are usually subject to dynamic changes. The accounting file owner can add Suraj and remove Mohan from the list of permissible accessors, as needed by business needs

The same object can be applied to both MAC and DAC. MAC takes priority over DAC, which ensures that only those who pass DAC will actually have access to the object from all those who are eligible for MAC access. For instance, a file may be classified as secret, which allows only individuals allowed to have secret access to the file. Yet only people in the project "National park" or in the environmental group or in "Karla Fort" are currently allowed to have access to millions of people who have restricted access through the government.

Object Reuse Protection

One way to preserve the performance of a computer system is by reuse of artifacts. The OS controls the allocation of resources and the operating system permits the next user or program to use the resource, since a resource may be accessed by other users or programs. Nonetheless, reusable items must be managed carefully in order to prevent a severe vulnerability. See why when a new file is generated, Normally file space is created by a pool of free space on the disk or another storage device previously used. Released space is returned to the "dirty" pool, which still contains the previous user's data. Because many users are writing to a file before attempting to read it, data from the new user delete the previous owner, so the previous user's information is not revealed improperly. A malicious user will demand a great deal of disk space and scavenge sensitive data

after that. The attack is referred to as the reuse of the object. It can occur with the main memory, processor registers and storage, another magnetic media like disks and dvds, or any other storage media that can be reused.

To avoid leakage of artifacts, operating systems erase (overwrite) all space until the next user has access to it. This hazard is especially vulnerable to magnetic media. Very accurate and costly equipment can occasionally distinguish the latest data from the previously reported data, from the previous and so on. This hazard is beyond the reach of this book, called magnetic remanence. S In any case, before allowing access to the resource, the operating system must take responsibility for "cleaning."

Complete Mediation

All accesses must be successful regulated to make mandatory or discretionary access control. Access to files is only inadequate if the attack is accessed through a memory or an external port or a Internet or a secret channel. The complexity of designing and implementing a trustworthy OS is considerably growing as more access routes have to be managed. Full mediation is carried out on highly trusted operating systems where all accesses are checked.

Trusted Path

One way to make a malicious user inappropriate is to "target" users and make them believe they interact with a legitimate security compliance system when they actually intercept and analyze by their keystrokes and commands.. A malicious spoofer can set a fake user ID and password system between the user and the legitimate system, for example. The spoofers capture the real user ID and the password when the illegal program queries the user for identity information; the spoofer may later on use the appropriate input data, potentially malicious deliberately, to access the system. Thus users want to provide an undeniable communication called a trusted path for sensitive procedures, such as setting a password or modifying their access permission, to ensure that only a valid user receives protected information. In certain trustworthy systems, a trusted user uses a specific key sequence which, by design, is directly intercepted by security enforcement software; safety-related changes can only be made on other trusted systems at system start-up before processes other than safety compliance code are performed.

Accountability and Audit

A security activity may be as simple as a individual accessing an item, such as a file, or as important as changing central access control data that affects all concurrent accesses. Accountability typically involves keeping a record of security events, listing every event and the individuals responsible for adding, eliminating, or modifying the event. Obviously, this audit log must be secured protected against external individuals, and each safety incident must be logged.

Audit Log Reduction

An audit log basic notion is desirable as it allows responsible parties to determine all activities that impact all the system's protected elements. Nonetheless, due to the volume and analysis, an audit

report can in fact be too difficult to manage. To understand why, consider gathering and analyzing details. In the extreme (as data may affect the survival of a company or the protection of a nation) we may argue that all changes or even every character read from a file are potentially safety relevant; changes could have an impact on the quality of the data, or even a single character may show the only really important part of the entire file. The series of specific instructions can also be security important as the direction of control by the programme's data is influenced by the application processes. For every connection to an individual file character, the audit log is enormous if an audit record is generated and for every instruction carried out. (In addition, each instruction should not be audited, since the audit order would then be audited itself. Such commands should be enforced in conjunction with orders which would be audited, and so on.)

In most trustworthy systems, just opening (first-access to-) and closing (last access-to-)files or related artifacts can help make the problem simpler. Objects including individual memory locations, hardware registries and instructions are also not reviewed. Even with such limitations, audit logs are generally quite big. Only a simple word processor can open a dozen or more support modules as it starts, build and delete hundreds or more temporary files while running. It can also open several other driver so that other tasks like complex formatting or printing can be done. So, one simple program could easily cause to the opening and closing of 100 documents, and complex structures can lead to the access of thousands of files in a fairly short period of time. At the other side, some systems read or update a single file on a continuous basis. The bank transactions can be handled by the bank account's personal account system all day long; what may be important is that the Bank's teller did not have access to the account file, but rather accessed the entries in the file. There is also too much data and in some situations inadequate auditing at the opening and closing level to meet security needs.

Intrusion Detection

The capacity to identify security deficiencies throughout the execution is closely linked to the audit reduction. As we saw in the the State Department for example, it is likely that the audit report includes too much information for a individual to examine. Technology for intrusion detection creates normal system usage patterns that cause an alert whenever the use becomes abnormal. Products are already widely available, following a decade of promising research results for intrusion detection. Many trustworthy operating systems have such a primitive device detection of intrusion. For more explanations of intrusion detection systems.

Kernelized Design

A kernel belongs to the functionality of the lowest level of the operating system. The kernel performs operations like synchronization, communication between processes, message passing and interrupt handling in the standard operating system architecture. The nucleus or core is also known Kernel. Lampson and Sturgis and Popek and Kline explain the concept of designing an operating system across a kernel.

The security mechanisms of the whole operating system shall be implemented by a security kernel. The security kernel includes the security interfaces between the hardware, the operating system and other computer system components.

A **security kernel** is definitely responsible for enforcing typically the security mechanisms of typically the entire operating system. The particular safety kernel provides typically the security interfaces among typically the hardware, operating system, along with other parts of the processing system. The operating system is usually designed to contain the security kernel inside the OS kernel. Ames is discussing security kernels in depth

There are several great design reasons why security functions might be isolated inside a security kernel.

- **Coverage.** The security kernel should pass through any access to a protected object. The OS will use the security kernel to make sure any access is tested in a system designed to use it.
- **Separation.** The separation of the security mechanisms from both the rest of the operating system and the user's space makes it even easier to prevent the operating system or users penetration.
- **Unity.** All security features are implemented with a single code collection, so monitoring the cause of any issues with such features is easier.
- **Modifiability.** Modifications in the security systems can be easily rendered and checked.
- **Compactness.** The security kernel is likely to be extremely small because it conducts mainly security functions.
- **Verifiability.** The security kernel is fairly small and it can be intensively examined. For example, formal methods may be employed to assure that somehow the design is protected by all security conditions (such as states and policy changes).

Reference Monitor

A reference monitor, which regulates the access of objects, is the most essential element of the security kernel. It's more important to obtain access control for devices, files, memory, communication between interprocesses and other objects, rather than the single piece of code. A reference monitor behaves like a concrete wall around the operating system or trusted software as shown by Figure 5-12.

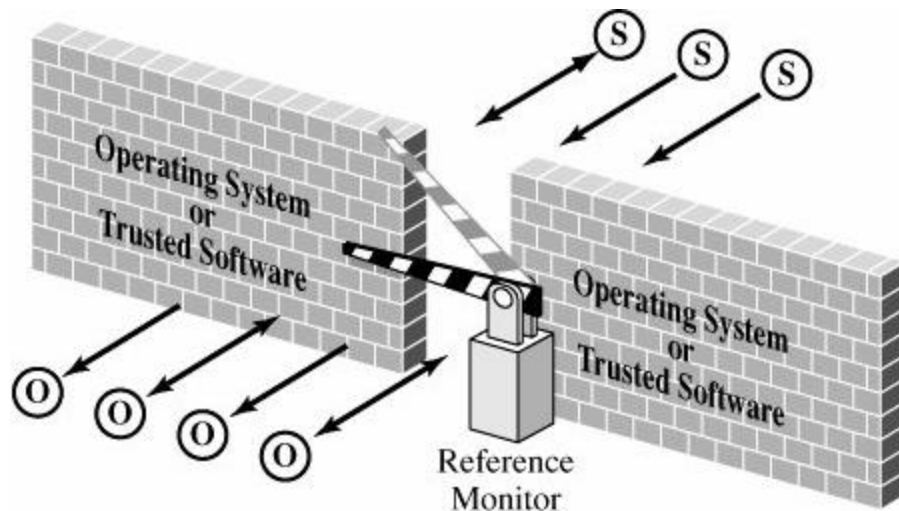


Figure 5.12. Reference Monitor.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

A reference monitor has to be

- **tamperproof**, that will be, impossible to weaken or even disable
- **unbypassable**, that will be, always invoked when accessibility to any object is needed
- **analyzable**, that is, compact enough to be put through to analysis and tests, the completeness which could be ensured

The reference monitor is not just a sensitive operating system's only security mechanism. The security package involves audit, identifying and authenticating and also the description of enforcement criteria, such as who the subjects allowed are and which objects they may access. Some other security components communicate with a reference monitor, collect data or provide it with the information it needs to function from a reference monitor..

Trusted Computing Base

The trusted computing base (TCB) is the name we call everyone necessary to implement the security policy in the trusted operating system. Conversely, we state the TCB is made up of the trusted operating system that dependent on with the proper execution of the policy. The TCB can be viewed in the following way as a complete unit. Suppose the modules in the TCB and the parts that are not segregated into a trusted operating system and allow most competent malicious programmers to develop all the non-TCB parts. Since the TCB deals with all of the security concerns, the malicious non-TCB parts are unable to prevent the proper implementation of the TCB security policy.

TCB Functions

Where is the TCB like? We can address this question by listing system elements that can rely on security enforcement:

- **hardware**, including cpus, memory, registers, and I/O devices
- **some notion regarding processes**, so that we are able to separate and protect security-critical processes
- **primitive files**, including the security access control databases and identification/authentication data
- **protected memory**, so that typically the reference monitor can end up being protected against tampering
- **some interprocess communication**, In order to transfer data to and disable certain pieces in various areas of the TCB. For example, an audit routine can be securely invoked and passed by the reference monitor.

It may appear that this list covers much of the operating system, but the TCB is only only a small part of it. For instance, while the TCB requires access to enforcement data files, it does not require a whole hierarchical directory file structure, virtual devices, indexed files or multidevice files. Therefore, a basic file manager could only handle the small, simple files that are required for the TCB. The most complex file management can be outside the TCB to supply externally accessible files. Typical TCB and non-TCB sections are shown in Figure 5-13..

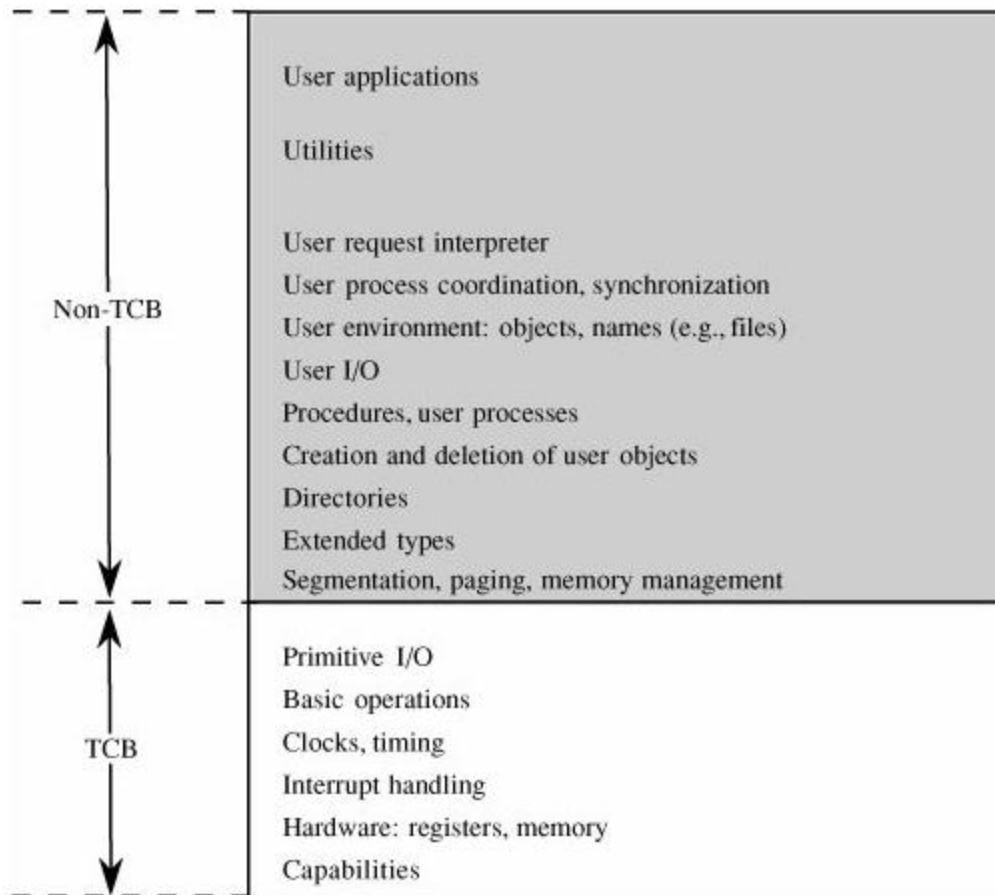


Figure 5.13. TCB and Non-TCB Code.

Above Image taken from the book **Security in Computing, Fourth Edition** By **Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger**

The TCB monitors four basic relationships that must protect the secrecy and integrity of each domain.

- **Process activation.** Processes are activated and deactivated frequently occur in a multi programming context. Moving from one process to another would entail the complete change of registers, relocation maps, lists of accesses to files, details on process status and other indicators which are mainly security critical information.
- **Execution domain switching.** Within one domain processes are also used to receive more sensitive data or information from systems in other domains.
- **Memory protection.** Since each domain contains memory code and records, the TCB must control memory references to make sure each domain's confidentiality and integrity.-

I/O functioning. Software is used for each character transmitted during an I / O process in certain systems. In the outermost domain this software links to the innermost (hardware) domain an I / O device. Therefore, all domains can be crossed by I / O operations.

TCB Design

It's useful to designers and developers to split the operating system in TCB and non-TCB aspects as this ensures every security code is put in one (logical) portion. However, the discrepancy is more than just factual. TCB code must run in a safe environment that separates the security compliance from non-TCB code. The TCB and non-TCB structuring must therefore be performed consciously. Nevertheless, after this structuring is done, code outside the TCB can, without compromising the ability of the TCB to implement protection, be modified at will. The ability to modify allows developers because it ensures that large parts of operating system applications, device drivers, interface managers and the like can be modified or replaced at any time. Finally, the separation into TCB and non-TCB greatly simplifies assessment for someone testing the security of a trusted operating system as non-TCB code does not have to be considered.

TCB Implementation

Activities related to security may be carried out in different locations. Security may apply to all memory accesses, each I / O operation, each file or system access, each user initiation or termination, and each communication between processes. Such individual tasks can be handled in separate modules in modular operating systems. There are also all security-related functions in each of these different modules.

The functionality of the existing operating system can kill the set of all security functions in a TCB. A single TCB can be too large to be easily analyzed. Nevertheless, a designer may decide to separate the security functions of an existing existing operating system, creating a security kernel. This form of kernel is depicted in Figure 5-14.

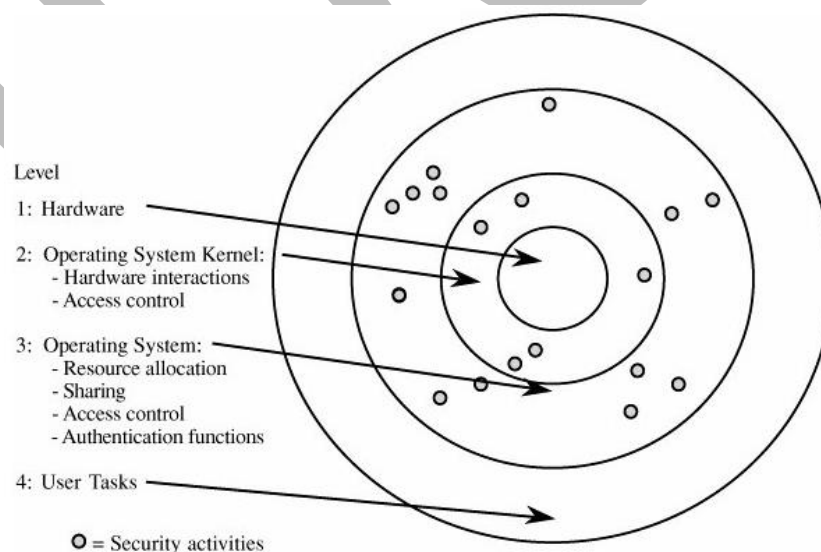


Figure 5.14. Combined Security Kernel/Operating System.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The security kernel must first be developed and the operating system is designed to be more efficient. Honeywell used this method to develop a prototype for its secure operating system, Scomp. It included only 20 modules to execute the primary security functions, with less than 1,000 lines of advanced source code. After Scomp's actual security kernel was developed, its functionality grew to about 10,000 code lines.

The security kernel is a safety-based architecture that forms an interface layer, just on top of the system hardware. The security kernel controls and executes all security functions in all hardware devices. The security kernel, based on hardware support, allows the operating system to perform most non-security functions itself. This makes a small and powerful security kernel. Computer systems have at least three executive domains as a by-product of this partitioning: security kernel, operating system and user. See Figure 5.15.

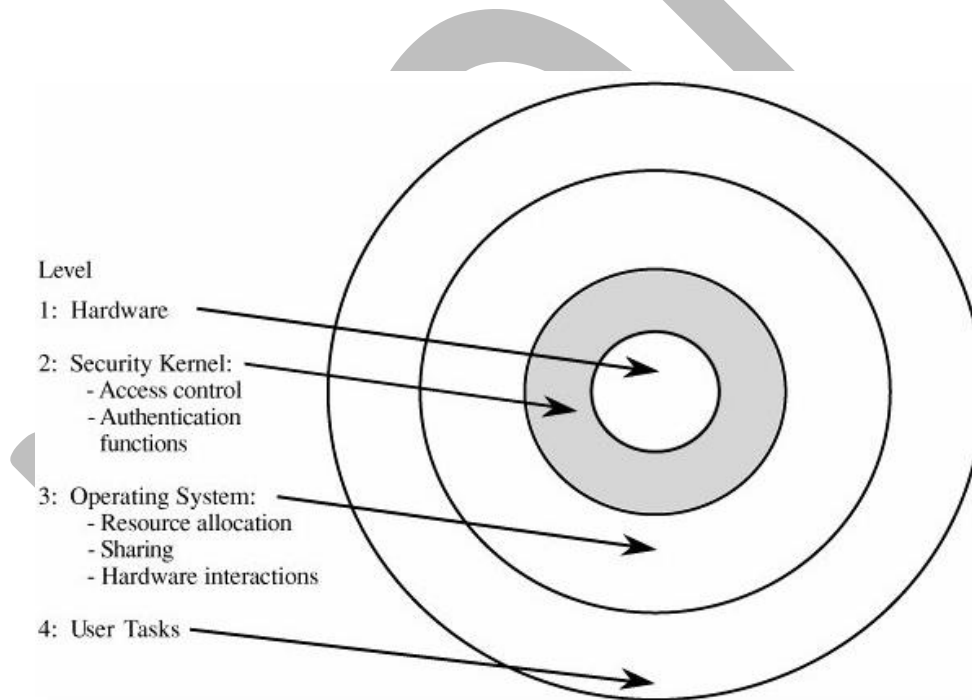


Figure 5.15. Separate Security Kernel.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Separation/Isolation

Rushby and Randell are listed four ways of separating one process from the other. Two independent processes use two distinct hardware systems of physical separation. Sensitive operations on reserved machines may, for example, be performed; unsensitive tasks are carried out on a public machine. Hardware separation includes various desirable features, including support

for several separate execution threads, memory protection, I / O mediation and at least three different execution rights. Temporary separation takes place as multiple processes take place at various times. Many military networks, for example, run unsensitive jobs from 8:00 a.m. to noon to 17:00 p.m. with sensitive computation. Encryption is for cryptographic segregation such that two processes can be performed at the same time so unauthorized users do not have readable access to sensitive data. Logical separation, also called isolation, is created when a process such as a reference monitor distinguishes between one user and another user's objects. For each of these types of separation, secure computing systems were developed.

Multiprogramming operating systems can separate each user from each other so that interactions between users are managed only cautiously. Many operating systems have been developed to provide us with an unified environment. This implies, as shown in Figure 5-16, that one copy of the OS can be used by many users. The OS is also divided into two different parts at the highest and lowest memory addresses

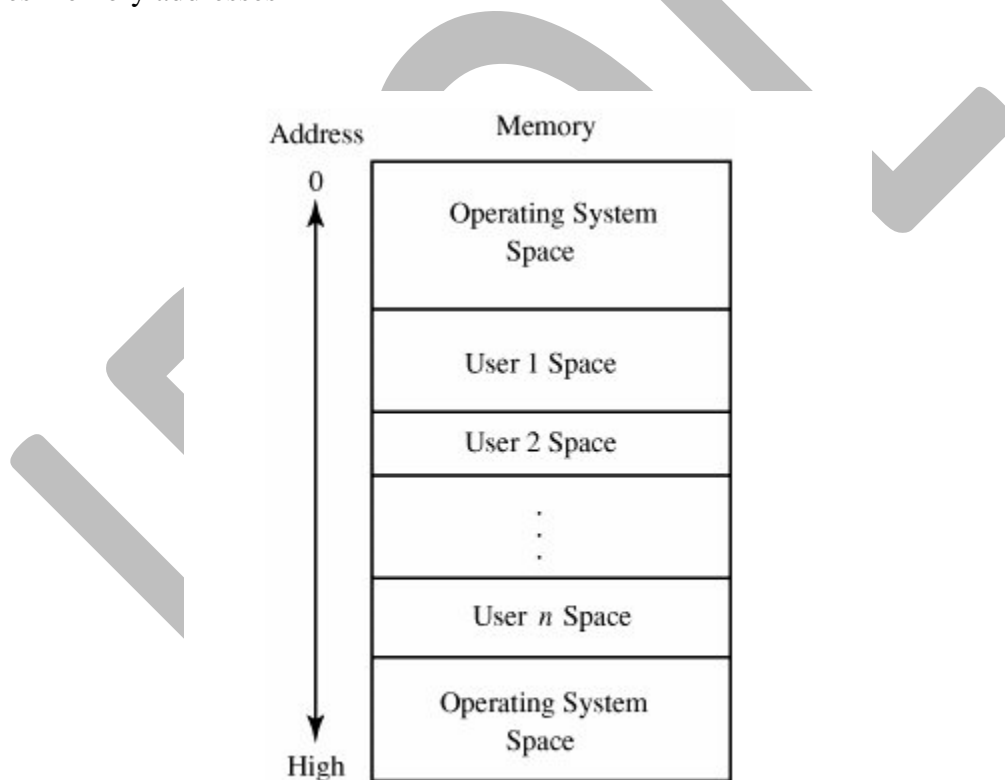


Figure 5-16. Conventional Multiuser Operating System Memory.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Virtualization

Virtualization is a powerful method for trustworthy device designers, as it provides users with carefully orchestrated access to complex items. Virtualization means that the operating system emulates or simulates a computer system resource set. We say a virtual machine is a series of actual or simulated equipment: a [central] processor that runs a set of instructions, a range of directly addressable storage devices and I / O devices. These facilities support plan execution.

Of course, physical hardware or software will support virtual resources, but the physical resources shouldn't be the same as those simulated. Several examples of such simulation are available. For example, printers are often simulated in multiuser environments on direct access devices. A large one can simulate multiple small disks. Some noncontiguous memory will accommodate a much wider virtual memory space with demand paging. And even on PCs it is popular to simulate faster-memory space on slower disks. The operating system thus provides the user with the virtual resource, while the security kernel monitors the user access precisely.

Multiple Virtual Memory Spaces

Virtualization is used to provide a logical separation of the IBM MVS / ESA operating system to allow physical separation seem to the user. IBM MVS / ESA is a paging system such that a page mapping mechanism distinguishes the logical address space of each user from that of the other users. Therefore, MVS / ESA incorporates the operating system in the logical address space of each device, so that a device is running on a completely different computer

Several paging system only contain the virtual address space of a user; the operating system is outside of the virtual address space of the user. The operating system is therefore part of every MVS / ESA user's logical space. Therefore, as shown in Figure 5-17, MVS / ESA appears to user as a single-user system.

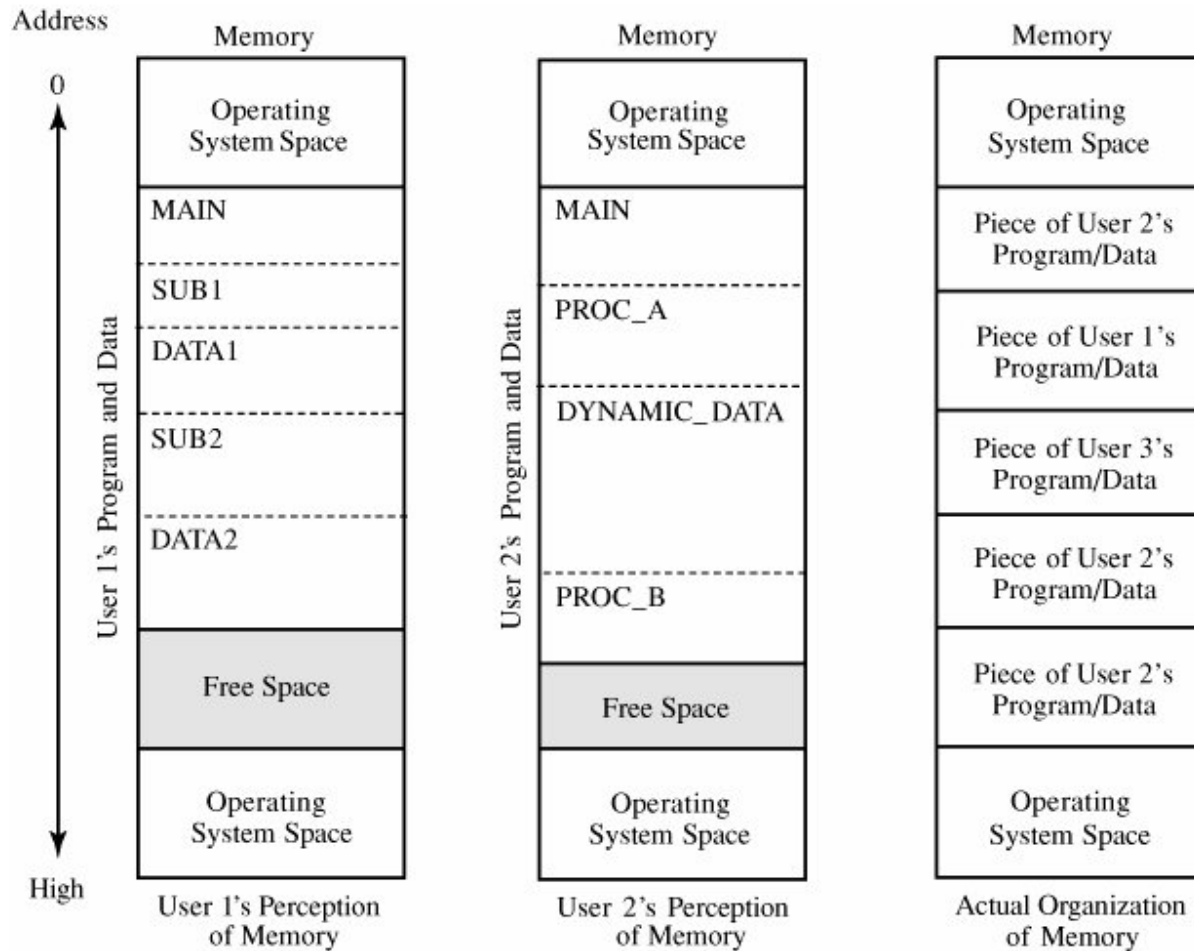


Figure 5-17. Multiple Virtual Addressing Spaces.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Memory management is a primary advantage of MVS / ESA. The virtual memory space of each user can be over 16 million bytes as high as the total addressable memory that can be handled. This memory representation provides protection as a second advantage. As the logical address space of each user contains the operating system, the user has a different impression that may be valid.

Virtual Machines

A stronger level of security is offered by the IBM Processor Resources / System Management System (PR / SM). A traditional operating system is equipped with hardware and devices directly controlled by the operating system in Figure 5-18. PR / SM provides each user with a full virtual machine that not only provides logical memory for every user but also logical I / O devices, logical files and other logical resources . This feat is carried out by PR / SM by strictly separating

resources.

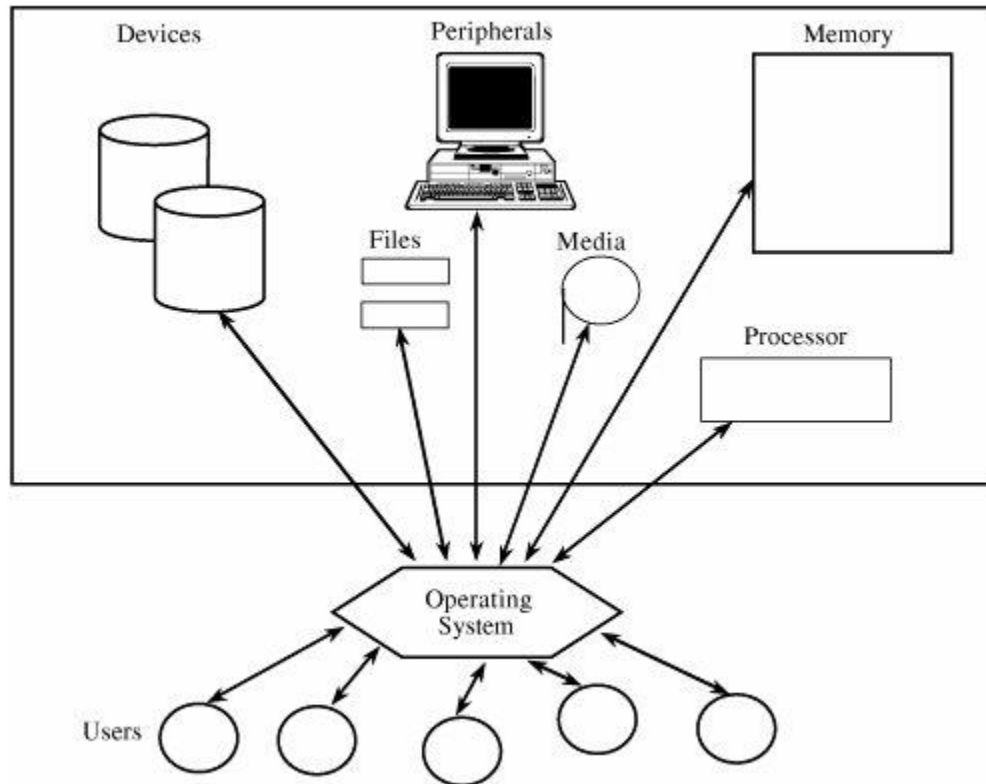


Figure 5.18. Conventional Operating System.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The system PR / SM is a fundamental part of the virtual memory concept. The virtual memory provides the user with a logically segregated memory space; the virtual memory space is usually bigger than the actual memory. A virtual machine offers the user a complete collection of features and functions, i.e. a complete computer which can be very different from the real one. Logically these virtual hardware resources are often different from other users. Figure 5-19 illustrates the relationship between virtual and actual machines.

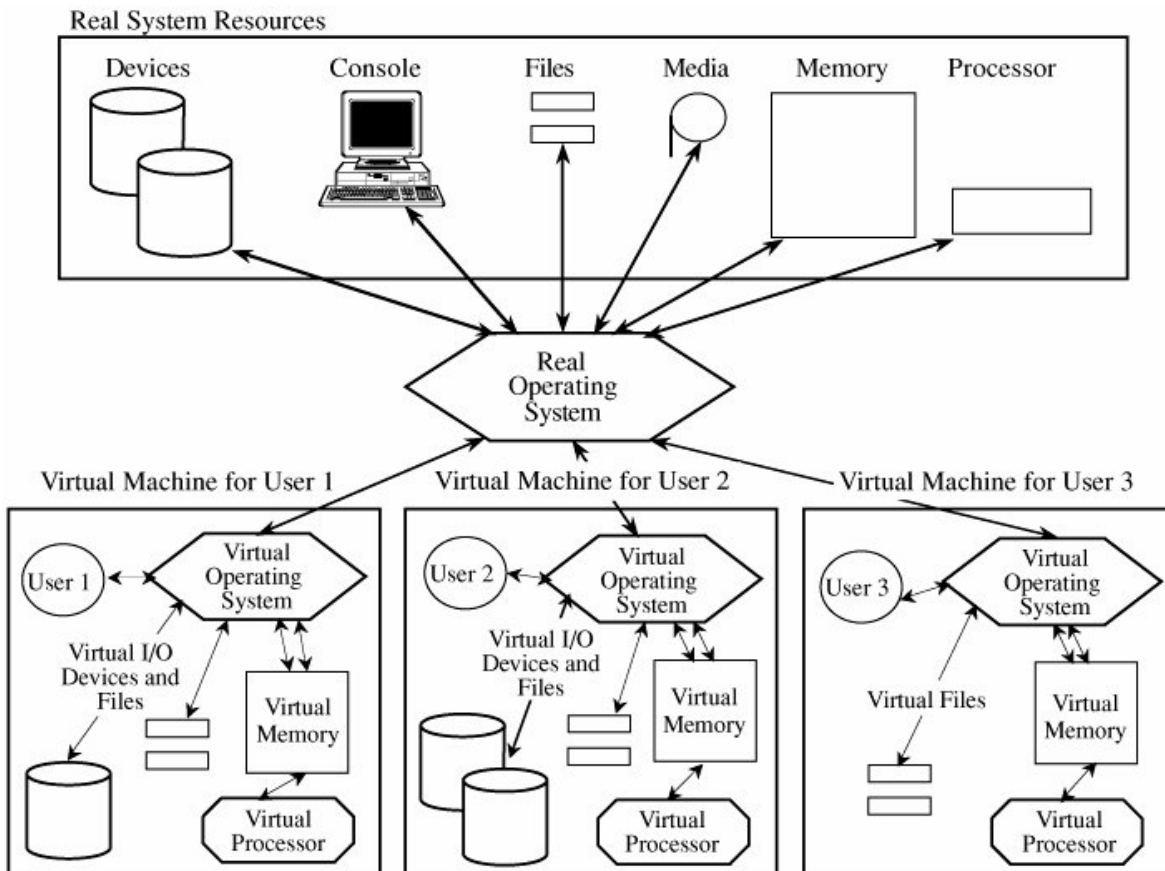


Figure 5.19. Virtual Machine.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Including MVS / ESA and PR / SM enhance individual users 'separation from other user and system hardware. This additional difficulty, of course, raises the overall burden of these translation and security levels. In the following section we discuss alternative designs that reduce the security complexity of an operating system.

Layered Design

As described above, there are at least four levels in the kernel-based operating system: hardware, kernel, operating system and user .. Such layers can contain sublayers. The kernel has five separate layers . It is not unusual for the user to have pseudo system programs, like database managers or graphical user interface shells, that are separate security layers in their own way.

Layered Trust

The secure operating system's layered view is portrayed in a sequence of clustered circles, with most critical operations in the most innermost layers. The process 'distance to the center can then be measured on the trust and access rights: the closer the processes are to the center are more

trustworthy. But we can also view the trusted operating system as a stack with the hardware's closest security functions in the layers. The Figure 5-20 indicates such a system.

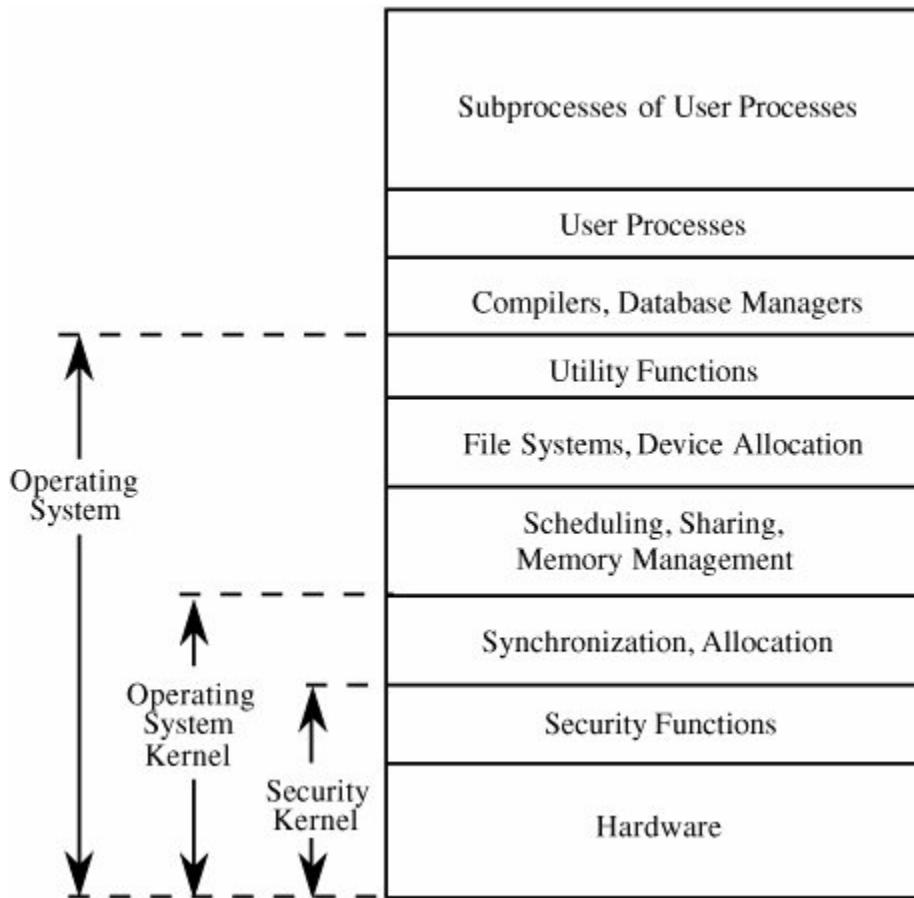


Figure 5.20. Layered Operating System.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Some protective functions beyond the security kernel are carried out in this design. User authentication, for example, can require access to a password table, challenge the user to offer a password, check that the password is correct, and so on. The downside of all those activities within the security kernel is that there is no high security prerequisite for any operations (e.g. user contact formatting and user searching on a list of identified users in a table).

Conversely, in many separate modules we can perform a single logical function; we called it a layered design. The layering is built on trust and access rights. In other words, a set of modules that work in different layers can perform a single task, as illustrated in Figure 5-21. The modules perform certain sensitivity operations of each layer

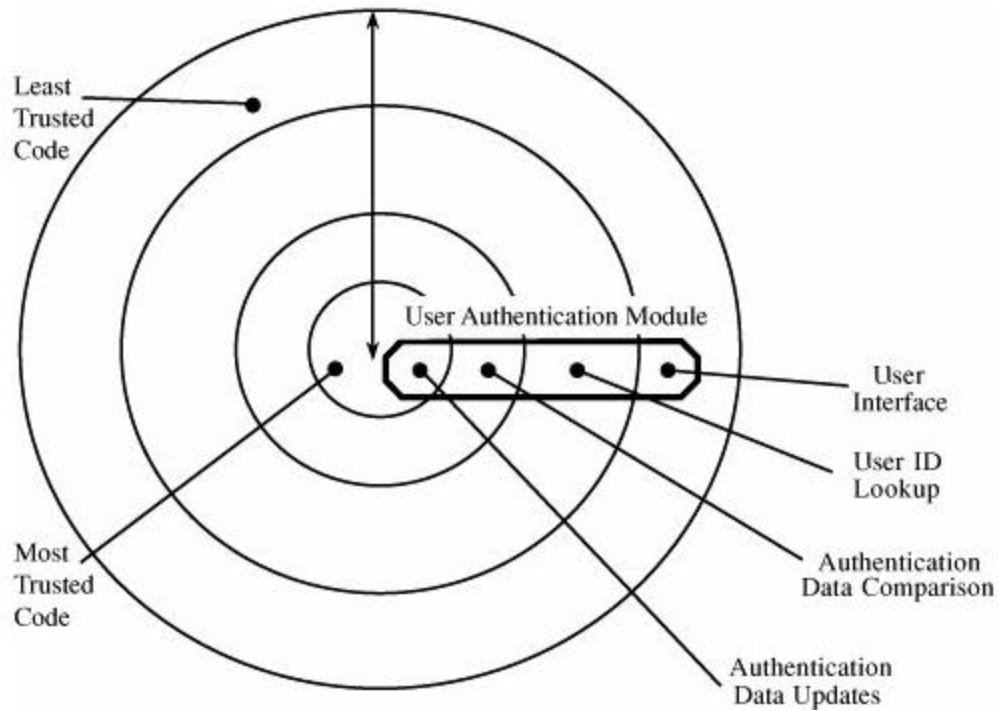


Figure 5.21. Modules Operating In Different Layers.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The layered structure of the Provably Secure Operating System (PSOS) is defined by Neumann. Table 5-4 demonstrates that some lower levels have some or all of its functionality in higher levels, but these items can be properly embodied below. Layer

Table 5.4. PSOS Design Hierarchy.

Level	Function	Hidden by Level	Visible to User
16	User request interpreter		Yes
15	User environments and name spaces		Yes
14	User I/O		Yes
13	Procedure records		Yes
12	User processes and visible I/O		Yes

11	Creation and deletion of user objects		Yes
10	Directories	11	Partially
9	Extended types	11	Partially
8	Segments	11	Partially
7	Paging	8	No
6	System processes and I/O	12	No
5	Primitive I/O	6	No
4	Arithmetic and other basic operations		Yes
3	Clocks	6	No
2	Interrupts	6	No
1	Registers and addressable memory	7	Partially
0	Capabilities		Yes

From [NEU86], © IEEE, 1986. Used with permission.

Above Table taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Another way to accomplish encapsulation is through a layered approach. Layering is well known for creating a good OS. -- layer uses the central layers as services, and each layer provides the layers further afield with a certain degree of functionality. Through this way, we can "delete" each layer and still have a lower functionality with logically complete system. Layering gives a good illustration of how design characteristics are changed and balance

The control of damage is another reason for the layering. Take two examples of risk in Tables 5-5 and 5-6 from Neumann's . To see the factors. Any hardware malfunction, software fault or unpredictable condition, in a conventional, non-hierarchically designed system (as shown in Table

5-5), can cause a catastrophe in an apparently unsecured portion, because the issue has unlimited effect, and because the system design means it can not be trusted that any feature does not (indirectly) have any security effect.

Table 5-5. Conventionally (Nonhierarchically) Designed System.

Level	Functions	Risk
All	Noncritical functions	Disaster possible
All	Less critical functions	Disaster possible
All	Most critical functions	Disaster possible

Above Table taken from the book **Security in Computing, Fourth Edition** By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Table 5-6. Hierarchically Designed System.

Level	Functions	Risk
2	Noncritical functions	Few disasters likely from noncritical software
1	Less critical functions	Some failures possible from less critical functions, but because of separation, effect limited
0	Most critical functions	Disasters possible but unlikely if system simple enough to be analyzed extensively

Above Table taken from the book **Security in Computing, Fourth Edition** By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Simply by contrast, as shown throughout Table 5.6, hierarchical structuring has two benefits:

- **Hierarchical structuring** allows the identifying the most critical elements, which then can be thoroughly evaluated for accuracy, thereby reducing the number of problems.
- **Isolation restricts** the influence of problems at and above the level of hierarchy, thereby reducing the consequences of other problems

These designs have been the foundation for several trusted system models, such as the kernel, separation, isolation and hierarchical structures. We have stood the test of time as best design and implementation.

5.5. Assurance in Trusted Operating Systems

This chapter provides moved our dialogue from the overall to this. We begun by studying the latest models of of protection devices. By enough time we reached the final section, we reviewed three principles isolation, stability kernel, and split structure used in building secure os's, and we viewed in detail in the approaches used by developers of particular os's. Now, we guess that an operating-system provider has had these considerations into consideration and claims to truly have a secure design. It really is time for all of us to consider confidence, ways of persuading others a model, style, and implementation happen to be correct.

Typical OPERATING-SYSTEM Flaws

Regularly throughout our research of operating-system security features, we've used the key phrase "exploit a vulnerability." Through the entire years, various vulnerabilities have already been uncovered in lots of operating systems. They will have gradually happen to be corrected, and your body of understanding of likely weak areas has grown.

Known Vulnerabilities

In this segment, we discuss usual vulnerabilities which have been uncovered in os's. Our goal isn't to supply a "how-to" tips for prospective penetrators of os's. Rather, we review these flaws to comprehend the careful research necessary in making and testing os's. User interaction may be the largest single way to obtain operating-system vulnerabilities, for a number of reasons:

- The interface is conducted by independent, brilliant equipment subsystems. The humancomputer program often falls beyond your safety kernel or safety measures restrictions applied by an operating-system.
- Program code to connect to users is frequently much more sophisticated plus much more dependent on the precise device components than code for just about any other element of the computing program. Therefore, it really is harder to examine this program code for correctness, aside from to validate it formally.
- User interactions tend to be character oriented. Once again, in the fascination of fast information transfer, the os's designers could have tried to get shortcuts by restricting the amount of instructions executed by operating-system during actual info transfer. Occasionally

the instructions taken away are the ones that enforce security guidelines as each identity is transferred.

A second notable weakness in operating-system security displays an ambiguity in gain access to policy. Similarly, you want to separate customers and guard their individual assets. Alternatively, users be determined by shared usage of libraries, utility plans, common information, and system dining tables. The differentiation between isolation and posting is not constantly clear on the policy level, therefore the distinction can't be sharply attracted at implementation.

A third potential difficulty area is imperfect mediation. Recall that Saltzer advised an operating-system design where every requested entry was inspected for right authorization. Even so, some systems verify access only one time per interface operation, method execution, or device interval. The system can be acquired to implement total protection, however the policy choice on when to invoke the system is not finished. Therefore, within the lack of any explicit need, system designers take up the "most effective" enforcement; that's, one that will result in the least usage of machine resources.

Generality is really a fourth safety weakness, specially among commercial os's for large processing systems. Implementers make an effort to provide a opportunity for users to personalize their operating-system installation also to allow installing software packages compiled by other companies. A few of these plans, which themselves use within the operating-system, must perform with exactly the same access privileges because the operating system. For instance, there are plans offering stricter access handle than the normal control available from your operating-system. The "hooks" where these packages happen to be installed may also be trapdoors for just about any user to permeate the operating-system.

Types of Exploitations

Earlier, we reviewed why an individual interface is really a weak point in lots of major os's. We start out our illustrations by discovering this weakness in more detail. On some devices, after access may be checked to start a user procedure, the operation proceeds without following checking, resulting in basic time-of-check to time-of-use imperfections. Checking access authorization with each identity transferred is really a substantial overhead for any protection technique. The command usually resides within the user's storage. Any user can transform the foundation or destination deal with of the control after the functioning has got commenced. Because gain access to was already checked once, the brand new address will undoubtedly be used without even more checking it isn't checked whenever a piece of files is moved. By exploiting this flaw, consumers have been in a position to transfer files to or from any recollection address they really want.

Another exemplary case of exploitation will involve a procedural issue. In one technique a particular supervisor function was initially reserved for installing other security deals. When carried out, this supervisor contact returned handle to an individual in privileged method. The functions allowable for the reason that mode weren't monitored closely, therefore the supervisor call could possibly be used for gain access to control or for just about any other high-security technique access. This supervisor call expected some work to execute, nonetheless it was fully on

the system. Extra checking must have been utilized to authenticate this program performing the supervisor submission. As a substitute, the access protection under the law for any subject matter coming into under that supervisor need might have been limited by the objects essential to perform the event with the added program.

The time-of-check to time-of-use mismatch can add security problems, as well. In an episode predicated on this vulnerability, admittance permission is examined for a specific user to gain access to an object, like a buffer. But between your time the gain access to is approved along with the access actually arises, the user shifts the designation of the thing, so that rather than accessing the permitted object, an individual today accesses another, undesirable, one.

Assurance Methods

Once we appreciate the possible vulnerabilities in something, we can put on assurance ways to look for the vulnerabilities and mitigate or eradicate their effects. In such a section, we think of three such strategies, showing how they provide us confidence in a very system's correctness: screening, confirmation, and validation. Nothing of these is certainly entire or foolproof, and each features benefits and drawbacks. However, used in combination with knowing, each can participate in an important position in deriving general assurance on the systems' security.

Testing

Testing, , may be the most widely acknowledged assurance method. As Boebert observes, conclusions from tests derive from the actual merchandise being evaluated, definitely not on some abstraction or precursor of the merchandise. This realism is really a security advantage. Even so, conclusions predicated on testing are always limited, for the next reasons:

- Assessment can display the lifetime of an issue, but passing testing does not illustrate the lack of problems.
- Testing sufficiently within reasonable moment or effort can be difficult as the combinatorial explosion of inputs and interior states makes screening very complex.
- Testing based simply on observable outcomes, not on the inner structure of something, does not make sure any amount of completeness.
- Testing in line with the internal composition of something involves modifying the merchandise by adding program code to draw out and display inner states. That more functionality impacts the product's actions and will itself be considered a way to obtain vulnerabilities or cover up other vulnerabilities.
- Testing real-time or intricate systems presents the issue of monitoring all says and triggers. This issue makes it difficult to replicate and analyze challenges described as testers continue.

Ordinarily, we think about testing with regards to the creator: unit examining a component, integration testing to make sure that modules function correctly together, function tests to track correctness across all areas of a given work, and system assessment to combine components with software. In the same way, regression testing is conducted to be sure a change to 1 part of something will not degrade any functionality. But also for other tests, adding acceptance tests, an individual or consumer administers tests to find out if that which was ordered is what's delivered.

Thus, a significant aspect of confidence is considering if the tests run work for the application form and degree of security. The type and forms of testing echo the developer's assessment approach: which studies address what concerns.

Similarly, you should recognize that evaluation is almost generally constrained by way of a project's spending plan and routine. The constraints generally mean that assessment is incomplete for some reason. Because of this, we look at notions of check coverage, evaluation completeness, and assessing effectiveness within a testing strategy. The greater complete and helpful our testing, a lot more confidence we've in the program. More info on testing are available in Pfleeger and Atlee

Penetration Testing

A testing strategy usually used in laptop security is named penetration trials, tiger team evaluation, or honest hacking. In this process, a workforce of specialists in the utilization and design and style of os's tries to split the system getting examined. The tiger staff knows well the normal vulnerabilities in os's and computing techniques, as explained in previous areas and chapters. With this particular knowledge, the workforce attempts to recognize and exploit the system's certain vulnerabilities. The task of penetration testers carefully resembles what a genuine attacker might perform

Penetration testing can be both a skill and a research. The artistic area requires careful research and creativeness in selecting the test conditions. But the technological side involves rigor, order, perfection, and group. As Weissman observes, there's an organized technique for hypothesizing and verifying defects. It isn't, as some might suppose, a arbitrary punching contest.

Using penetration screening is similar to asking a auto mechanic to look more than a used car over a sales great deal. The mechanic has learned potential weak places and checks as much of them as you possibly can. Chances are that a great mechanic will see significant troubles, but getting a problem (and repairing it) is not any promise that no additional problems are usually lurking in other areas of the machine. For instance, when the mechanic investigations the fuel method, the coolant system, along with the brakes, there is absolutely no assurance that the muffler can be good. Just as, an operating-system that fails a penetration test out may own faults, but something that will not fail isn't guaranteed to turn out to be fault-free. Even so, penetration testing pays to and often detects faults that may have been disregarded by other styles of screening. One possible reason behind the results of penetration evaluation is its employ under real-life circumstances. Users often working out a system with techniques that its makers never expected or intended. Therefore penetration testers can exploit this real-life atmosphere and knowledge to be sure kinds of challenges visible.

Penetration testing is certainly favored by the commercial neighborhood who think knowledgeable hackers will check (attack) a niche site and find issues in hours or even days. These folks don't realize that finding defects in complex program code can take 2 or 3 weeks if not calendar months. Indeed, the initial military red groups to test stability in software methods had been convened for 4- to 6-30 days workout routines. Anderson et al. explain the restriction of penetration tests. To get one flaw in an area of just one 1 million inputs may necessitate screening all 1 million options;

unless the area is reasonably constrained, this search can be prohibitive. Karger and Schell explain that even with they well informed testers of a bit of malicious program code they put in something, the testers were not able to get it. Penetration assessment isn't a magic way of finding fine needles in haystacks.

Formal Verification

The most strenuous method of examining security is certainly through conventional verification,. Formal confirmation uses regulations of mathematical reasoning to demonstrate a system has selected security houses. In formal confirmation, the operating-system is modeled plus the operating system ideas are referred to as assertions. The assortment of products and assertions can be regarded as a theorem, that is then confirmed. The theorem asserts which the operating system is certainly correct. That's, formal confirmation confirms the fact that operating system supplies the security features it will and little or nothing else.

Proving correctness of a whole operating system is really a formidable task, typically requiring months and even years of work by several individuals. Computer programs known as theorem provers can help in this work, although much individual activity continues to be needed. The quantity of work expected and the techniques used are properly beyond the range of this reserve. However, we demonstrate the general basic principle of confirmation by presenting a straightforward example that utilizes proofs of correctness.

Consider the move diagram of Figure 5.22, illustrating the reasoning in an application to look for the smallest of a couple of n ideals, $A[1]$ through $A_{\text{good}}[n]$. The circulation chart includes a single identified starting point, an individual identified ending stage, and five interior blocks, consisting of an if-then composition including a loop.

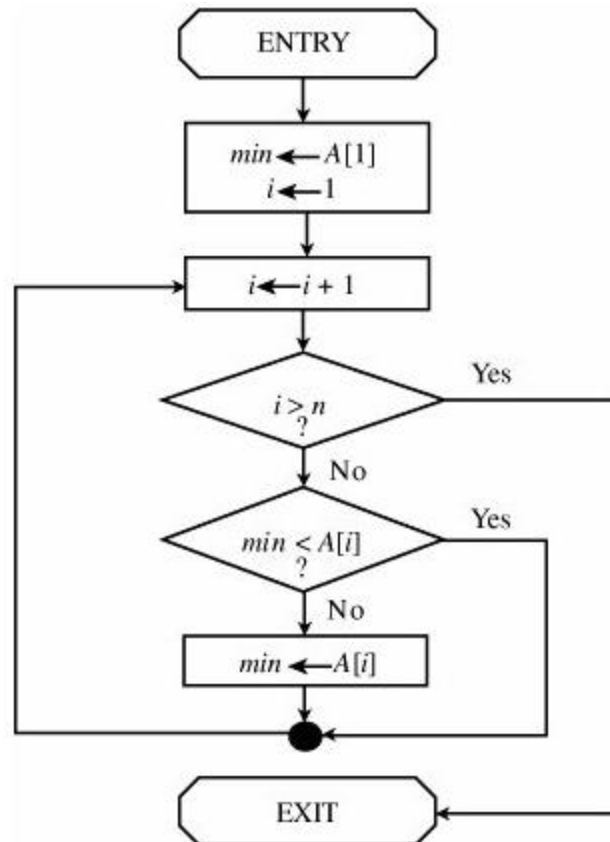


Figure 5.22. Flow Diagram for Finding the Minimum Value.

Table

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

In program confirmation, we rewrite this program as some assertions concerning the program's factors and values. The original assertion is really a statement of ailments on entry towards the module. Up coming, we identify some intermediate assertions from the work on the module. We likewise determine an concluding assertion, a declaration of the anticipated result. Ultimately, we demonstrate that the original assertion turns logically for the intermediate assertions that subsequently lead logically towards the ending assertion.

We can officially verify the illustration in Shape 5-22 through the use of four assertions. The initial assertion, P, is really a statement of original conditions, assumed to become true on access to the task.

$n > 0$ (P)

The next assertion, Q, may be the result of using the initialization program code in the initial box.

$n > 0$ and (Q)

$1 \leq i \leq n$ and

$\min \leq A[1]$

The 3rd assertion, R, may be the loop assertion. It asserts what's true in the beginning of every iteration from the loop.

$n > 0$ and (R)

$1 \leq i \leq n$ and

for several j , $1 \leq j \leq i - 1$, $\min \leq A[j]$

The ultimate assertion, S, may be the concluding assertion, the assertion of conditions accurate at that time the loop leave occurs.

$n > 0$ and (S)

$i = n + 1$ and

for several j , $1 \leq j \leq n$, $\min \leq A[j]$

These four assertions, found in figure 5.23, catch the essence from the flow chart. The next phase in the confirmation process involves demonstrating the logical development of the four assertions. That's, we must demonstrate that, presuming P holds true on entry to the procedure, Q holds true after conclusion of the initialization segment, R holds true the very first time the loop will be entered, R holds true each time with the loop, and the reality of R means that S holds true with the termination from the loop

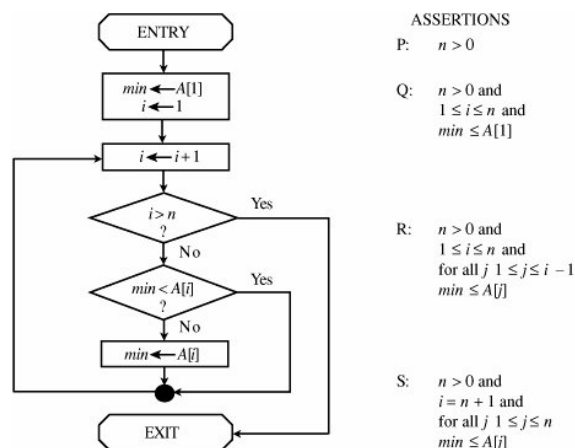


Figure 5.23. Verification Assertions.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Clearly, Q practices from P plus the semantics of both statements in the next box. Whenever we enter in the loop for the very first time, $i = 2$, therefore $i - 1 = 1$. Hence, the assertion about min applies limited to $j = 1$, which comes after from Q. To confirm that R remains to be legitimate with each execution with the loop, we are able to use the concept of numerical induction. The foundation with the induction is the fact R was real the very first time from the loop. With each iteration in the loop the worthiness of i rises by 1, so it's necessary to display simply that $\text{min} \leq A[i \text{ actually}]$ because of this new price of we. That proof comes after from this is of the evaluation and replacement claims. Therefore, R holds true with each iteration with the loop. Eventually, S employs from the ultimate iteration worth of R. This task completes the conventional verification that flow graph exits with the tiniest price of $A[1]$ through $A[n]$ in min.

The algorithm (certainly not the confirmation) shown here's frequently used for example in the initial couple of weeks of introductory encoding classes. It really is quite simple; actually, after researching the algorithm for a short while, most students encourage themselves how the algorithm is appropriate. The confirmation itself takes a lot longer to explain; in addition, it takes far more lengthy to write compared to the algorithm itself. Consequently, this proof-of-correctness instance highlights two primary difficulties with elegant verification strategies:

- Time. The techniques of formal confirmation are frustrating to perform. Proclaiming the assertions at each move and verifying the reasonable flow in the assertions happen to be both slow functions.

Complexity. Formal confirmation is a sophisticated process. For a few systems with many says and transitions, it really is hopeless to attempt to state and confirm the assertions. This example is especially real for systems which have not been made with formal verification at heart.

Validation

Formal verification is really a particular instance of this more general method of assuring correctness: confirmation. As we have observed in Section 3, there are lots of ways to demonstrate that each of an system's functions performs correctly. Validation may be the counterpart to confirmation, assuring that the machine developers have executed all requirements. Hence, validation makes certain that the developer is definitely building the proper product (based on the specs), and confirmation checks the grade of the execution .There are many various ways to validate an operating-system.

Open Source

A debate has opened up in the program development area over so-called open up source os's (along with other programs), ones that the source program code is freely produced for public evaluation. The arguments happen to be predictable: With open up source, several critics can peruse the program code, presumably finding imperfections, whereas shut (proprietary) source helps it be more challenging for attackers to get and exploit imperfections.

The Linux operating-system is the best example of open up source software, even though way to obtain its predecessor Unix seemed to be also accessible. The open resource idea is finding on: In accordance with a study by IDG Study, reported within the Washington Blog post, 27 per-cent of high-end machines now manage Linux, instead of 41 percent for your Microsoft operating-system, and the open up source Apache internet server outruns Microsoft Web Details Server by 63 percentage to 20 percentage.

Lawton lists extra benefits of available source:

- **Expense:** As the source code can be acquired to the general public, if the dog owner charges a higher fee, the general public will trade the program unofficially.
- **Top quality:** The program code can be examined by countless reviewers that are unrelated for the development energy or the company that developed the program.
- **Help:** Because the public finds imperfections, it may in addition be in the very best situation to propose the fixes for all those flaws.
- **Extensibility:** The general public can readily body how to increase code to meet up new needs and will reveal those extensions with various other users.

Opponents of open release dispute that presenting the attacker understanding of the look and execution of a bit of code enables a seek out shortcomings and a blueprint for his or her exploitation. Many industrial vendors have compared open source for a long time, and Microsoft happens to be being really vocal in its opposition. Craig Mundie, mature vice us president of Microsoft, claims open source computer software "puts at an increased risk the continuing vitality with the independent software industry". Microsoft favors a design under which it could share source program code of a few of its goods with selected spouses, while still keeping intellectual property privileges. The Alexis de Tocqueville Establishment argues that "terrorists attempting to hack or disrupt U.S. laptop networks will dsicover it easier if the government attempts to change to 'wide open resource' as some communities propose," citing risks against air visitors control or monitoring systems.

But noted personal computer security researchers dispute that wide open or closed supply is not the true issue to look at. Marcus Ranum, leader of Network Airline flight Recorder, has mentioned, "I don't believe making [computer software] open resource contributes to rendering it better by any means. What makes very good software is definitely single-minded concentrate." Eugene Spafford of Purdue College agrees, declaring, "What really can determine whether it's trustable is high quality and care. Was basically it designed properly? Was it designed using proper resources? Did individuals who constructed it use self-control and not squeeze in a lot of characteristics?" Ross Anderson of Cambridge College argues that "you can find more pressing protection challenges for the open up source area. The connections between security and safety and openness is usually entangled with efforts to use safety mechanisms for industrial benefits, to entrench monopolies, to regulate copyright, and most importantly to regulate interoperability."

Evaluation

Most system buyers (that's, customers or system buyers) aren't security experts. They want the security capabilities, but they aren't usually with the capacity of verifying the accuracy and

reliability or adequacy of check coverage, examining the validity of the proof correctness, or figuring out in any some other way a system appropriately implements a safety measures policy. Thus, it really is useful (and often essential) with an independent alternative party assess an operating system's safety. Independent authorities can review certain requirements, design, execution, and proof assurance for something. Because it is effective to truly have a standard technique for an assessment, several schemes have already been devised for structuring an unbiased review.

5.6 Review Question

1. A principle of typically the BellLa Padula model seemed to be not mentioned in this specific chapter. Called the tranquillity principle, it states that this classification of a subject matter or object does not really change although it is being referenced. Explain the objective of the tranquillity principle. Do you know the implications associated with a model where the comfort principle is not genuine?

2. Subjects can obtain objects, but they could also access other subject matter. Describe what sort regarding reference monitor would handle access in the circumstance of a subject doing work on another subject. Explain what sort of research monitor would control gain access to in the case involving two subjects interacting.

3. List the original source and conclusion of all information runs in each of typically the following statements.

- a. `sum: sum= a+b+c;`
- b. `if a+b < c+d then queen: =0 else q: =1;`
- c. `write (a, b, c);`
- d. `read (a, b, c);`
- e. `case (k) of`

`0: d: = 10;`

`1 , 2: d: = 20;`

`other: d: = 30;`

`end; /* case */`

- f. `for i: =min to max do k: =2*k+1;`

- g. `repeat`

`a[i]: =0;`

`i: =i-1;`

`until i ≤ 0;`

4. Does the particular system of all subsets of a finite arranged under the operation "subset of" (\subset) form a new lattice? Why or the reason why not?

5. Can the user cleared for $\langle \text{secret}; \{\text{dog, cat, pig} \} \rangle$ have accessibility to documents classified inside of each of the next ways within the military safety model?

1. $\langle \text{top secret}; \text{dog} \rangle$
2. $\langle \text{secret}; \{\text{dog} \} \rangle$
3. $\langle \text{secret}; \{\text{dog, cow} \} \rangle$
4. $\langle \text{secret}; \{\text{moose} \} \rangle$
5. $\langle \text{confidential}; \{\text{dog, pig, cat} \} \rangle$
6. $\langle \text{confidential}; \{\text{moose} \} \rangle$

6th. According to the BellLa Padula model, what constraints are placed on a couple of active subjects (for instance, two processes) that have to have to send and get signals to and by one another? Justify your response.

7. Write an established of rules combining typically the secrecy controls from the BellLa Padula model together with the honesty controls of the Biba model.

8. Demonstrate a way for limited transfer regarding rights in the GrahamDenning model. A limit associated with one is adequate. Of which is, give an approach by which A can easily transfer to B appropriate R, with the accessibility that B can exchange that right to virtually any one other subject. Typically the subject to which M transfers the right are not able to transfer the right, neither can B transfer that again.

9. Explain just what is necessary to supply temporal separation. That is usually, what conditions must become met to ensure two process to be adequately divided?

10. Does the regular Unix operating system work with a nondiscretionary access command? Explain your answer.

10. Why is labeling involving objects a security necessity? That is, why are unable to the trusted computing basic just maintain an accessibility control table with articles for every single object and every single subject?

12. Label honesty is a technique that will ensures that the content label to each object is altered only by the respected computing base. Suggest a new method to implement content label integrity for an information file. Suggest a technique to implement label sincerity for a callable process.

13. Describe a condition when you might desire to allow the safety kernel to violate one particular of the security attributes of the BellLa Padula model.

14. Explain this specific is of the phrase granularity in comparison with access management. Discuss the tradeoff among granularity and efficiency.

15. Explain what sort regarding semaphore could be employed to implement a hidden channel in concurrent running. Explain how concurrent running primitives, for example fork and even join, might be used to be able to implement a covert funnel in concurrent processing.

16. The Unix main system constructions files by using a new tree. Each file will be at a leaf associated with the tree, plus the data file is identified with the (unique) path from the underlying to the leaf. Each and every interior node is the "subdirectory, " which identifies the names with the pathways leading from that client. A user can stop access through a client by restricting access to be able to the subdirectory. Devise the method that uses this kind of structure to implement some sort of discretionary access policy.

17. In the Unix data file system described in this kind of chapter, could a nondiscretionary access policy be described so that an end user has access to some sort of file as long as the consumer has access to most subdirectories higher (closer towards the root) in the record structure? What would get the effect of this particular policy?

18. I/O looks as the source associated with several successful methods involving penetration. Discuss why I/O is hard to safeguarded within a computing system.

5.7 References

1. Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger - RAND Corporation Publisher: Prentice Hall
2. Cryptography and Network Security - Principles and Practice fifth edition Stallings William Publisher: Pearson
3. Cryptography And Network Security 3rd Edition behrouz a forouzan and debdeep mukhopadhyay 3/E Publisher: McGraw Hill Education
4. Cryptography and Network Security, 3e Atul Kahate Publisher: McGraw Hill

Chapter 6. Database and Data Mining Security

6.0. Introduction

6.2. Security Requirements

6.3. Reliability and Integrity

6.4. Sensitive Data

6.5. Inference

6.6. Multilevel Databases

6.7. Proposals for Multilevel Security

6.8. Data Mining

6.9 Review Question

6.10 References

6.0 Introduction

Protecting data is an essential part of the systems being secure, and many of the users are dependent on a database management system (DBMS) for the management of data for protection. In this chapter, we will look at the security mechanism applied to the database management systems, for instance how we can design the essential security for the software and execute an essential task in the software. DBMS protection is a newer concept for a new programmer rather than an operating system. Hence Databases are important for an organization and government organization, for holding important data and information that leads to an organization for competence with another organization. When the business process is revamped for creating a new model of the systems and the new innovative thing has emerged, one of the main things is systems should be carefully scrutinized whether the databases are suitable for the business process. The database is more like a software thing. The organizational data and contents are a crucial resource for the corporation for investment so their data must be protected.

6.1. Introduction to Databases

In this topic, we will start with the database and their terminology that is used in the database. We will be drawn as an example of a relational database management system (RDBMS) that is widely used in the world. However, this concept is applicable to any kind of database. Hence we will describe an important concept and use this concept for database security related.

Concept of a Database

A database is an organized collection of data that stored and accessed in an electronic form from the computer systems. The database management software is used to store the data in a related form. There are sets of rules that organized the data in a relationship among the data. The record in the database is stored in the file but in a physical format where there is no concern for the user where it is stored. A database administrator is a person who defines the rules and regulations for data for an organized database and gives the right for accessibility for the users what content can be accessed by the user or not. The interaction of the users with the database through a specialized program called Database Management Systems (DBMS). This is an interface where the user writes the queries in Structured Query Language (SQL) to fetch the data from the database.

Components of Databases

The files of the database contain the records, each of the record are related to the collection of other records in the database. Consider an example given in Customers Table 6.1. Customer table consists the fields name (CUST_FIRST NAME ,CUST_LAST NAME,CUST_AGE , CUST_ADDRESS , CUST_SALARY) . Here the database is viewed as two- dimensional table in which the fields are considered as column and records is considered as a row in the table and each field of the record is the data item of the table.

Table 6.1 Customer Table

CUST_FIRST NAME	CUST_LAST NAME	CUST_AGE	CUST_ADDRESS	CUST_SALARY
Pramod	Mali	23	Ahmedabad	2000
Pramod	Rathod	43	Ahmedabad	2000
Sanket	Kadam	35	mumbai	4000
Sanket	Singh	27	Delhi	1500
Sanket	Rao	28	hyderabad	1500
Ramesh	Mukerjee	25	Kolkata	3000
Ramesh	Badal	25	Kota	6000
Ramesh	Thakkar	25	Raipur	5000
Suresh	Mittal	28	Kolkata	4000
Ganesh	Talwar	32	Bhopal	6000

The above example of the table is the logical structure of the database known as a schema . the particular user will have access to some part of the table or database that is called subschema. The table is divided into two parts consider an example of the subschema. Table 6.1 is complete customer table, in which left part of the table (I.eCUST_FIRST NAME, CUST_LAST NAME) are part of subschema and another part of subschema is (CUST_AGE , CUST_ADDRESS,CUST_SALARY), the administrator have given the privileges for access only the customer id and customer name and prevent the user from accessing customer age, customer address and customer salary.

Here the rules of the database are identified with column names. The name of the column in the databases is called an attribute. A relation is a set of a column. In the table theCUST_NAME CUST_ADDRESS is relation formed by taking the name CUST_NAME and CUST_ADDRESS . Table 6.2 shows the relationship in the database

Table 6.2 Relation in a Database.

CUST_FIRST NAME	CUST_LAST NAME	CUST_ADDRESS
Pramod	Mali	Ahmedabad
Pramod	Rathod	Ahmedabad
Sanket	Kadam	mumbai
Sanket	Singh	Delhi
Sanket	Rao	hyderabad
Ramesh	Mukerjee	Kolkata
Ramesh	Badal	Kota
Ramesh	Thakkar	Raipur
Suresh	Mittal	Kolkata
Ganesh	Talwar	Bhopal

Queries

The user can retrieve, modify, add, or delete the record of the database through the command by interacting with the database management systems. The Database command is called a query. Database management systems have rules and regulations for writing the queries in the form of SQL. The notation of SQL is the same as and English the full of SQL is a structured query language was developed by IBM. we have written the syntax of an SQL is same as the English language which is easy to understand

SELECT CUST_FIRSTNAME = 'Sanket'

The above queries will retrieve all details of Promad in the CUST_NAME field.

We can get subschema by executing the SQL queries by applying a certain condition to retrieve the record from the database. For example select record where the customer address is Ahmedabad, producing the result in table 6.3.

Table 6.3. Result of Select Query

CUST_FIRST NAME	CUST_LAST NAME	CUST_AGE	CUST_ADDRES S	CUST_SALAR Y
Pramod	Mali	23	Ahmedabad	2000
Pramod	Rathod	43	Ahmedabad	2000

Other complicated selection criteria can be possible with the help of operators such as **and**(\wedge) and **or** (\vee), and significantly less than ($<$) operator. For an example select query is

SELECT (CUST_ FIRST NAME='Ramesh') \wedge (CUST_AGE='25')

Table 6.4 . Results of Select-Project Query.

CUST_FIRST NAME	CUST_LAST NAME	CUST_AGE	CUST_ADDRESS	CUST_SALARY
Ramesh	Mukerjee	25	Kolkata	3000
Ramesh	Badal	25	Kota	6000
Ramesh	Thakkar	25	Raipur	5000

If we want to combine the record of the two tables in the database .we can use the SQL Join means for combining the records of the two tables

Consider the following table:

Table 6.5 – CUSTOMERS Table

CUST_ID	CUST_Name	CUST_AGE	CUST_ADDRESS	CUST_SALARY
1	Ramesh	32	Ahmedabad	3000
2	Suresh	25	Delhi	4000
3	Pawan	34	Mumbai	6000
4	Pramod	26	Raipur	8000
5	Kailash	22	Panji	10000
6	Mandar	30	Kota	5000
7	Nikhil	35	Jaipur	1500

Table 6.6 – ORDERS Table

ORDERID	DATE	CUST_ID	AMOUNT
102	10/1/2019	3	3000
101	12/11/2019	3	1500
103	10/20/2019	2	1300
104	5/20/2019	2	1400

Now, let us join the two tables by using the SELECT Clause of SQL Statement as follows

SELECT CUST_ID, CUST_Name, CUST_AGE, AMOUNT

FROM CUSTOMERS, ORDERS

WHERE CUSTOMERS.CUST_ID= ORDERS.CUST_ID;

Result of the query is as follows:

CUST_ID	CUST_Name	CUST_AGE	CUST_ADDRESS	AMOUNT
3	Pawan	34	Mumbai	3000
3	Pawan	34	Mumbai	1500
2	Suresh	25	Delhi	1300
4	Pramod	26	Raipur	1400

Advantages of Using Databases

The main aim of the database is: A database is a central repository of data, whether data is stored in a one location where too many people were accessing the data as they needed, the actual storage is done at physical level but the user is unaware the physical location of data where it reside on the hard disk. So the user is able to see the logical arrangement of data in the database. Due to that, a database gives several benefits over the file system:

- **shared access**, many users can use it as a central repository of data.
- **minimal redundancy**, individual users not required to collect and maintain their own set of data.
- **data consistency**, user change the value of data at one location should affect the data value of all the users.
- **data integrity**, all the records in the table will be protected against the unintentional or malicious user.
- **controlled access** only authorized users to have permission for viewing, modifying and deleting data in the files

6.2. Security Requirements

The basic requirement for database security is as follows :

- **Physical database integrity.** The data stored in the database physically have a problem, like a power outage, and due to power outage, the systems administrator has to construct the database from the crash.
- **Logical database integrity.** The structure of the database is maintained, due to which the logical integrity of the database is intact, for example, if a user has modified the value of one field in the database will affect other fields of the database.
- **Element integrity.** Each element in the database is kept accurate.
- **Auditability.** The systems administrators are able to track who or what resources are accessed by the users.
- **Access Control.** A user is able to access the only authorized data, and different users have different privileges for accessing the different parts of the database.
- **User authentication.** Each user is properly authenticated by providing the credential and the different users have different rights for accessing the data such as read and write.
- **Availability.** Users are able to access the database by 24*7 *365 days which they are authorized for accessing the data in the database.

We briefly verify each one of these requirements.

The integrity of the Database

The data in the database are stored in the central location, so the user has to ensure the data in the database is accurate and maintain integrity. The database administrator has ensured that data in the database have to access only to authorized users. It also has to protect the data from

corruption like illegal access by a certain user in which they are not authorized or from power failure.

There is the two situation the data in the database will lose it integrity: one situation is the whole database can be corrupted due to power failure (physical damage of stored media i.e hard disk) or individual data items are unable to read or access.

Element Integrity

The integrity of the database element maintains the accuracy of the data. Legitimate users are responsible for entering the values in databases so the user can do the mistake while entering the value, collecting the values or executing the results.so DBMSs take special precautions to catch errors and correct errors after the user is inserted in the database.

The corrective measurements of the data in the database are taken in three ways. The first corrective measurement is the DBMS has to check the value of the field such as appropriate value is put in the respective position of the fields. The fields can be numeric, character, Boolean, etc. it also ensured that the field values are fall in the specific range.

The second corrective measure is to provide access control. The different users are located at a different place in the different geographical locations .some of the users are modifying, accessing and writing the data in the database. Some of the users are unaware of the location of files in the database so sometimes accidentally the users can change the data in the other location of the data values in the database so the integrity of the data in the database can lose. Database management systems are able to solve the above problem. The data in the database are stored in the central location so database administrator has given the different access control mechanisms to a different user for writing, deleting or executing the data.

The third corrective measurement is database integrity maintain in the changelog of the database. A change logs consist list of every day's changes that are made in the database so it contains original data and modified data. Using the log files, the database administrator can undo or redo any change in the database if the error occurs in the data values.

Auditability

Some programs are able to generate an audit report of all users who have an access database. These records can help the database administrator for maintaining the particular database's integrity, or find out the affected value is made by the user at what time.

A second advantage, users can obtain protected data incrementally; that means no single accessibility reveals protected data, yet a set of continuous accesses viewed together reveals the data, just like finding the clues inside a private investigator novel.

Fine-grained audit policies can be to create audit records in the database log files, maintaining the record, when a table is accessed during specific periods or specific columns are being accessed.

Access Control

Databases often logically separated simply by user access privileges. For instance, all users are granted access to the common data which they are privileged for access that data, for example, the salary department will acquire salary data and the marketing department will acquire sales information. Hence the data are stored central manner and maintained the accuracy of the data. Limited privileges for the user for accessing the database provides benefits to maintain the data integrity.

The database administrator provides different accessibility for the different users to allowed which data, fields, records, as well as a low-level elements in the database. The DBMS needs to enforce the policy for accessing all prescribe data or no gain access to where the user is prohibited. There are different modes of access in the database provided to users. A user or software might have privileges for reading, changing, deleting, or appending the data values in the database.

User Authentication

The DBMS may require a different level of authentication. For example, a DBMS might insist users for passing credential and day-time. This authentication is particularly performed by the operating system. Hence the DBMS software runs on the top of the operating systems. The overall systems design of operating systems does not provide full-fledged security as there is no trusted path from DBMS to the operating system if the DBMS receives any suspicious files from the users, the DBMS is enforced to used own mechanism for authentication.

Availability

A DBMS is a combination of hardware and software. It is usually a software that uses other hardware and computer software resources, for many users it just an application run over the operating systems. Therefore the users often take the DBMS for granted and use it as an essential tool for performing the specific tasks. Hence the database system is busy processing giving task so other users are not able to access the database. For example, if two users are requesting the same record so DBMS must arbitrate; one particular user is denied access to the record and other users are permitted. The DBMS system may uphold the unprotected data for avoiding protected data to reveal this make the user unhappy.

Integrity/Confidentiality/Availability

The three important features of computer security integrity, confidentiality, and availability are related to database management systems.

integrity is applicable to each element of a database as well as to the entire database. Due to this integrity is a crucial part of designing the database management systems.

Confidentiality is an important concern for the database due to the inference problem, where users can directly access the sensitive data.

6.3. Reliability and Integrity

In the database, data is coming from various sources and collated in a uniform format. The user of the database is except that the data provided by the database should be in reliable ways software programmers are except the software should have reliability. The software should run along the amount of time without the exception. In an organization data is plays an important role to meet the competition in the market with a different organization.so the user is more concerned about the data that is, the data should be lost and corrupted so the data management system should provide the ensure about the protection of the data.

Reliability and integrity of the data have been categories in a three-dimension way such as:

- **Database integrity:** The user much worries about the database protection as a whole, as the disk drive failure and master database index get corrupted to do the power failure. An operating system provides integrity control and various recovery mechanism for this issue.
- **Element integrity:** It more concerned about the value of the desired data items is modified by the illegitimate users. An appropriate access control mechanism has to be provided by the database management system from unauthorized users.
- **Element accuracy:** In the database, the accurate value of data elements should be written. The user has to check the values of elements prior to written in the data this can prevent the user from entering the wrong values in the database. The constraints condition must be applied to an element.

Protection Features from the Operating System

A system administrator should take the back up the files associated with a database periodically together with other useful documents. The execution of the files should be protected from the operating system's standard access control mechanisms. The operating system should check all the data elements while writing and reading operations on the I/O devices.

Two-Phase Up-date

A critical problem with database administration is the failure of systems in the middle of the transaction over the database. In this case, some of the data items have waited for modification, half of the values converted to a new value and another half of value persists as the old value. This is a critical problem in database transactions some time it can easily detect, sometimes not. The solution to the problem was by proposed by Lampson and Sturgis and mostly DBMS systems in term of two-phase update.

Update Technique

In the update Technique, the first phase is known as the intent phase where the database management system gathers the essentials from the various resources to be needed for an updating task. The database system might gather data, create a temporary record, open files, lock the other users for modifying it, that is required- red an initial plan for updating, but it not making any changes to the database. The first phase is repeatable for an unlimited number of

times because it does not take any permanent action in the database system. If a system fails during the process of the first phase stage, there is no harm in completion here all the actions will be restarted from the initial phase after the system process is resumed.

In the first phase, the last event is called committing, which entails the writing of a new commit flag for the database. In the DBMS, the commit flag is used when there is no point of return: after the commit flag, the permanent changes are going to happen in the database.

The permanent changes happen in the second phase. In the second phase, no action occurs from before the commit phase can be repeated, the update activities involving phase two when to be repeated as usually as needed. In the second phase, if the system failure occurs, the incomplete data is persisted in the database, during that the system can repair the incomplete data by performing all transaction activities in the second phase. After the second phase has been completed, the database will be again complete.

Redundancy/Internal Consistency

Inconsistency data in the database can be detected by the additional features provided by the database management system. The database management systems check duplication fields or shadow fields in the database depending upon the importance of the data.

Error Recognition and Correction Codes

In Database, One of the types of redundancy is error detection and error correction rules. There are different techniques for error detection and error correction such as parity parts, Hamming rules, and cyclic redundancy investigations.

In this technique, certain codes could be applied to single fields, records, or the complete database. Whenever a data item is positioned in the database, the appropriate check codes will be computed and placed; whenever a data item will be retrieved, a similar check code is executed and their comparative values are calculated. In this case, if the values happen to be unequal, that indicates towards the DBMS an error has happened in the database. A few of these codes point to the place where the error has occurred; others will precisely demonstrate what the right value should be. If the more information is provided, the greater space is required to store the value.

Shadow Fields

In shadow fields, the entire attribute or entire record could be duplicated kept inside a database. In some situations, an error is detected in the database. If the data are not reproducible, then the duplicate copy can be provided as an immediate replacement, redundant fields require additional space for storage.

Recovery

A DBMS can contain a log of user accessibility information, particularly change is occurring in the database. In case of failure, the database can be reloaded from the backup copy from the database and all later changes will be then applied from the audit log.

Concurrency/Consistency

In concurrency mechanisms, multi-users are simultaneously accessing the database. If the same database is accessed by the two users at the same time there should be no conflict neither the two users are conflict with each other. Therefore, there should be a locking mechanism applied to the database. If two user tries to access the same database, there should be no conflict for accessing the same value from the database.

If the two users try to modify the same file in the database, we usually assume that there is no conflict between each other because they know each other what to write the content in the same file; here the value being written will not depend on the prior value of the data item. Even though, the assumption isn't quite precise.

Let see how concurrent modification will get us into trouble, consider an example that the database includes seat reservations for a specific railway train. Agent A, book a ticket for Ramesh, submits a query on the railway reservation website for getting seats no which is available. The Agent knows that Ramesh prefers the side window seat and plus the agent finds that side window seat S1/3, S1/5, and S1/23 are available. Exactly at the same time, Agent B is trying to book side window seats for a group of three members of the family. In response to a query, the databases show that S1/37/38/40 and S1/57/58/59 will be the two remaining sets of three adjacent unassigned side window seats. Agents A submits the update command

```
SELECT (SEAT-NO = 'S1/5')
```

```
ASSIGN 'Ramesh' TO PASSENGER-NAME
```

while Agent B submits the update command

```
SELECT (SEAT-NO = 'S1/5')
```

```
ASSIGN 'Suresh' TO PASSENGER-NAME
```

As well as send commands for seats no S1/57 and S1/58. After that two passengers have already been booked into the same seats which may below uncomfortable.

Both agents work properly by looking at a summary of empty seats, decided which one seat is empty from the list, and kept up to date the database showing to whom the seating was assigned. The issue in this example is the moment delay between reading a value from the database and updating a modification of this value. Through the delay period, another user provides access exactly the same data.

To solve this issue, the entire operation of the query update routine is done through a single atomic procedure by database management systems. The command from agent indicates that " read the current value of seat PASSENGER-NAME for seat no S1/5; if it's 'UNASSIGNED', change it to ' Ramesh' (or 'Suresh')." The read modifies operation must be accomplished as it considers continuous cycles without letting any other users access to the PASSENGER-NAME field for seating S1/5. In this cycle, the second agent will not book the ticket until unless first

agents have been completed the booking operation; at that time, the PASSENGERNAME's value would no more be 'UNASSIGNED'.

A final problem in concurrent access is read-write operation. Assume one user is trying to updating a value in the database at that time a second user wishes to learn it. When the read operation is performed while a write operation is progressing, here the reader may receive files that are simply partially updated. As a result, the DBMS locks any read request until a write operation is been completed.

Monitors

The monitor is a specialized component of DBMS which is responsible for the logical structural integrity of the database. A responsibility of a monitor is to check the values being entered to make sure there should be consistency with all of that other database or with characteristics of the data field. For example, alphabetic characters for a numeric field might be rejected by the monitor.

Range Comparisons

A range comparison monitor will check each new value to ensure that the value has come under the acceptable range. If the data value is beyond the acceptable range, the monitor will reject the new value rather than entered into the database. For instance, the range of dates maybe 131, "/", 112, "/", 19002099

Range comparisons may also be applicable to numeric values. For example, the salary field of an employee may be limit to Rs 50,000, or how big house may have that constrained being between 500 and 5,000 square feet

State Constraints

State constraints outline the condition of the entire database. The database value should adhere to the guideline of this constraint. For instance, if the constraint is not meet according to constraint, there will be some error in the database

For the state constraints, consider an example of an employee's classification database where one employee in the organization considers as a "CEO" and each employee has employee id were different from another employee id. if there is a mechanical failure or software failure which causes a certain portion of a database record is duplicated, these are one of the unique constraints has been violated, so the state of a database must be tested to identify the record in the database that employee number or two records are classified as "CEO"

Transition Constraints

State constraint outline the condition of entire database whereas transition constraint outline the condition which is essential before the change can be made in the database. For an example prior to adding the newly appointed employee record in the database, In the database, there must be a post of an employee with the status as "vacant", after that newly appointed employee record in the database so the record change from "vacant" to the newly appointed employee.

6.4. Private Data

Most databases include sensitive information. Let's first presume, as just a working definition, that confidential information are not publicly accessible data. The classification of private data items and fields depends on the specific database as well as on the underlying relevance of data. Clearly, there is no confidential information contained in certain databases, such as an university library archive; other databases, namely defense were completely sensitive. Such two scenarios aren't very sensitive and all sensitive, since access controls could be used to cover the entire database.

Regular interactions among students, faculty and staff are carried out. It is important to know not all data can be treated equally as regards our storage sharing and disposal. In three ways, the MU classifies data:

Confidential data is the most sensitive and is required by law for the protection of students of MU, faculty and staff. Examples of confidential data include:

- Social Security Numbers
- Credit Card Numbers
- Health Records
- Financial Records
- Student Records

Private data is not deemed private, but reasonable efforts must be made to ensure that it is not ready for public access. Examples of private data include:

- Research Data
- Personal Contact Data
- Proprietary information
- MU ID (i.e. 91 number)

Public data is for public use and data protection is required at the owner's discretion. Examples of public data include:

- Public budget data
- Employee contact data
- Departmental Websites

There are many factors that make sensitive data.

- **Inherently sensitive.** The value itself revealing about the data is sensitive. Examples location of military equipment such as defensive missiles or the median income of doctors in a city with only one doctor
- **From a sensitive source.** The source from where the data are generated required confidentiality. For example .the information getting from a spy his identity is compromised if the information of the spy is disclosed.

- **Declared sensitive.** The database administrator or the owner of the data has declared the data is very sensitive. Examples are classified military data.
- **Part of a sensitive attribute or a sensitive record.** an entire attribute or record is classified as sensitive. For example, the salary attribute of the employee database or record describing the secret space mission.
- **Sensitive in relation to the previously disclosed information.** Some data becomes sensitive in the presence of other data. For instance, the longitude coordinate of the oil reservoir unveils a small amount of information, however, the longitude coordinate with the latitude coordinate pinpoints the reservoir.

Many of these factors should be considered to ascertain the sensitivity of the data.

Access Decisions

A database administrator is a person who determines what data should be present in the databases and who should access the content of the database. The database administrator considers the different requirements of the users to know specific details of the user and it will decide who must access what data in the database. The decision of the database administrator is based on the access policy of an organization.

The database administrator or DBMS is a software that is operated on data in the database and initial control information is required for making a decision for access policy. For example database administrator has privileges for allowing user A to access a data B. An application or software cannot do the instruction as it owns; With the implementation of the policy by the database administrator, the program follows the policy to perform the instruction where user A accesses B's data.

Several factors are considered by the database administrator while permitting for accessing the database. These factors consist, availability of the data, acceptability of access, and authenticity of a user. We grow on these three aspects below.

Availability of Data

In the database, one or more fields are not accessible. For instance, if a user is updating numerous fields in the database, other users are accessing these same fields of the database at the same time blocks temporarily. The blocking of fields indicate that the other users are not receiving inaccurate information while updating the database, a user may be blocked for the usage of several field and records of a database which leads to guarantee the consistency of data will maintain for others user means inaccurate information will not gain by the other user.

Acceptability of Access

In the database, one or more values can be sensitive so it is not accessible to ordinary users. A DBMS should no allow accessing the sensitive data to unauthorized users.

Deciding what's sensitive data should be accessible to which user is difficult .some time the fields may not directly accessible by the users. A user may have access to several fields that

contain sensitive data, however, the users might have projected to from particular fields which aren't sensitive.

Assurance of Authenticity

Prior access to the database, certain external characteristics about the users should be considered. For example, a user may allow by database administrator for accessing a database during working hours. The systems must able to track the previous queries to ensure that the combination of queries does not reveal the sensitive data

Types of Disclosures

Data may be sensitive, so their attributes may also be sensitive .we will see the descriptive information about the data is a type of disclosure.

Exact Data

The exact data is the most serious type of disclosure, it contains the exact value sensitive data itself. A user may be known about the sensitive data are been requested, or a user may request ordinary data without understanding that some of its data are sensitive. A sensitive data, by accidentally, could even deliver by faulty database administrator without the user are requested to it. In all the above cases, the result is been conclude be the same means the security and safety of the sensitive data are breached.

Bounds

Another type of disclosures is revealing sensitive data or values between certain bounds. That is, indicating a sensitive data, such as y , can be between two values, L and H . By taking a narrowing technique, a user may first decide that $L \leq y \leq H$ and then it decide whether $L \leq y \leq H/2$, etc, thereby allowing a user to determine y value to the desired accuracy. In another case, merely revealing that a value such as a scholarship budget or that the number of police agents exceeds a certain amount may lead to a serious breach of security. Sometimes, however, bounds are a useful way to present sensitive data It is not uncommon to release upper and lower bounds without identifying their specific records. For example, a company may announce that its salaries for programmers range from Rs 20,000 to Rs32,000. If you are a programmer earning Rs29,700. you can presume that you are fairly well paid, so you have the information you want; however, the announcement does not disclose who are the highest and lowest paid programmers.

Negative Result

Sometimes the negative result can be determined by querying one word, that is, that z is not the value of q . For example, In the Honours list, students name does not appear, you can conclude that the student's grade point is below average than a certain level. but this information is not too much revealed about the performance of the students, however, since the range of grade point averages is rather wide.

Existence

In some instances, the presence of data can be itself a sensitive data piece of information, whatever the actual value is. For instance, an employer does not want to know the usability of remote telephone lines by the employees which have been monitored by the employer because discovering a remote DISTANCE field in the personnel files might be revealing sensitive data.

Probable Value

Finally, it might be possible to look for the probability a certain element includes certain values. Suppose you intend to find out if doctor X is registered as a practitioner. Knowing that the doctor X record can be in the data, you post two queries towards the database:

How many people have 12 Doctor house, Marine Line Road as their official residence? (Response: 4)

How many people have 12 Doctor houses, MarineLine Road as their official residence and also have YES as the value of PRACTITIONER? (Response: 1)

From these inquiries you conclude, there's a 25 percent possibility that the doctor is president is a registered practitioner.

Security versus Precision

In the above examples, we have concluded that how hard it is to find out which data are usually sensitive and how to protect the data from unauthorized users. The situation will become complicated when the database administrator is the desire to share the nonsensitive data. For the secrecy reason, we want to reveal only the non-confidential data.

Security: the database administrator makes sure that data should be kept safe from corruption and data should be available to the users by some suitable access control. In order to provide security, it should only disclose the non-sensitive data and reject the queries that references to sensitive data.

Precision: In order to protect all the sensitive data, the database administrator should always as much as non-sensitive data possible.

We are able to depict the partnership between safety and accuracy with concentric circles. As Figure 6.3 displays, the sensitive info in the middle circle ought to be carefully concealed. The exterior band represents information we willingly disclose in reaction to queries. But we realize that an individual may come up with bits of disclosed info and infer various other, more deeply invisible, data. The amount displays us that under the outer layer could be yet extra nonsensitive information that an individual cannot infer.

6.5. Inference

The inference problem indicates that in which way we can infer or derive sensitive data from nonsensitive data. The inference problem is the most critical vulnerability in database security.

Consider a database of students with the following schema:

Table 6.7 database of students

student ID	student name	Student gender	Students fees	exam 1 score	exam 2 score	final exam score	Hostel Name
1	Ramesh	M	10000	40	30	70	Oval
2	Suresh	M	0	42	45	87	Pearl
3	Rita	F	11000	45	50	95	Diamond
4	Maya	F	8000	17	20	37	Lake
5	Praveen	M	10000	42	45	87	Opal
6	Suraj	M	12000	30	35	65	Diamond
7	Pankaj	M	7000	42	45	87	Pearl
8	Raveena	F	0	36	34	70	Oval
9	Komal	F	0	25	25	50	Opal
10	Aarti	F	8000	45	45	90	Diamond

In this Example the Suppose that students are not authorized to execute a query that will reveal exam grades of other students. But students are authorized to execute a query Q that will return results of the average grade for an exam of all students.

Here the attacker wants to access the exact score of another student for a particular exam and an attacker may be able to infer the grade of the student via Q.

In Inference attacks, an attacker wants some additional data from outside fields of a student's database. In this case, lets the attacker knows that the particular student appears late in the examination. Then the attacker can deduce the exam grade of this student by executing Q before the student takes the exam and then again after the student takes the exam.

For example, if there are 10 students in the class and the Q returns 100 before the student takes the exam, and 99 after, we can calculate the student's score as:

$$99 = (9 * 100 + x) / 10;$$

$$990 = 900 + x;$$

$$x = 90;$$

Here the attacker marks the score grades 90 for the particular student.

Direct Attack

An attacker attack directly on the targeted data, it is called a direct attack. when the database does not contain any protection mechanism so the attacker again accesses to the database successfully.

A sensitive query might be

List Student name where

Student gender=M \wedge final exam score= 87

This query discloses that for results of Praven, final exam score= 87. However, it is a direct attack because the attacker is fired query where students for whom final exam score =87, and the DBMS may reject the query of the attacker because it selects records for a specific value of the sensitive attribute final exam score

A less noticeable query is

List Student name where

(Student gender =M \wedge final exam score= 87)

(Student gender \neq M \wedge Student gender \neq F)

(Hostel Name =Silver)

At first glance, this query seems as if it will conceal the final exam score by selecting additional no final exam score records as well. On the other hand, this query nonetheless retrieves only 1 record, disclosing a Student name that corresponds to the very final exam score. The DBMS must know that the Student gender has simply the second clause will select no records. Even though that was achievable, the DBMS would also know that no records are present with Hostel Name =Silver, despite the fact that Silver might actually be an acceptable value for field Hostel Name.

Indirect Attack

These attacks do not directly attack the target but gather information regarding the data using other objects. There are Combinations of different types of queries are used but are difficult to track.

Sum

An attack by sum tries to infer value from a reported sum. For example, with the above database, It might seem safe to report students' fees total by students' gender and Hostname. Such a report is shown in Table 6.8. This report reveals that no female living in opal is having fees. Thus, we can infer that any female living in oval (such as Raveena) is certainly not having fess. This approach often allows us to determine a negative result.

Table 6.8 Sum of students fees by Hostel Name and Students gender

Sums of students fees by Hostel Name and Students gender.				
	Oval	Diamond	Opal	Total
M	10000	12000	10000	32000
F	7000	19000	0	26000
Total	17000	31000	10000	58000

Count

The count can be added to the fields with the help of sum statements to produce some more revealing results. Often these two statistics are often released for a database for allowing users to determine average values.

Table 6.9 Count of Students by Hostel Name and students gender. Combined with the sum table, however, this table demonstrates that the 1 males in oval, Diamond and opal are having fees in the amount of Rs 10,000 and 12,000 and 10,000 respectively. We can obtain the students' names by selecting the subschema of Student name, Hostel Name, this field is not sensitive because it delivers only low-security data on the entire database.

Table 6.9 Count of Students by Hostel Name and students gender

Table Count of Students by Hostel Name and student gender.				
	Oval	Diamond	Opal	Total
M	1	1	1	3
F	1	2	1	4
Total	2	3	2	7

Mean

The exact disclosure can be done by the arithmetic mean (average), if the attacker temper or manipulate the subject population . for example, consider the example of salary. Given the number of employees in the company, the mean salary for a company and the mean salary of all employees except the CEO, it is easy to compute the CEO's salary.

Median

This is a slight bit complicated process, Here we have to conclude a single value from the median, the number is sorted in ascending or descending order and find the midpoint number in the sorted number. Here The attacker wants to find midpoint number in the sorted list which point of intersection that happens to be exactly in the middle.

$q = \text{median}(\text{student fees where student gender} = M)$

$p = \text{median}(\text{student fees where final student score} = 90)$

reveal the precise final student score sum for Aarti.

Tracker Attacks

A tracker can fool the database administrator into seeking the desired data through the use of additional queries that produce smaller results. The tracker offers additional data being retrieved for just two different queries; both sets of records are canceled one another, leaving simply the statistic or desired results.

For example, if a user is identified uniquely by the characteristics C, then this can be represented by

then this can be represented by

$$C = C_1 \wedge C_2$$

If the database system will respond to queries for both (C_1) and $(C_1 \wedge C_2)$ then $(C_1 \vee C_2)$ is called the tracker of the individual

Table 6.10 A database giving salaries

Name	Profession	Male	Female	Salary
Mr A	Editor	Yes	----	Rs25,000
Ms B	Reporter	----	Yes	Rs 20 ,000
Mr C	Reporter	Yes	----	Rs 19,000
Mr D	Reporter	Yes	----	Rs 16,000
Mr E	Journalist	Yes	----	Rs 16,000
Ms F	Journalist	----	Yes	Rs 17,000
Ms G	J ournalist	----	Yes	Rs 17,000
Mr H	Engineer	Yes	----	Rs15,000
Ms 1	Engineer	----	Yes	Rs 14,000
Mr J	Reporter	Yes	----	Rs 13,000

Consider an example of above table which reveals confidential salary details of senior staffs in a newspaper business and an interloper want to find the salary of the reporter Ms. B. This can be done by using $C_1 = \text{reporter}$ and $C_2 = \text{female}$ so that the tracker is

$(\text{reporter} \wedge \neg \text{female}) = (\text{reporter and male})$

Question: How many reporters are there? Answer: 4

Question: How many (reporter and male) are there? Answer: 3

These two questions affirm that there is only one female reporter, Ms. B.

Question: Total salaries of reporters? Answer: Rs 68,000

Question: Total salaries for (reporters and males)? Answer: Rs 48,000

The inference from these answers is that the salary of Ms. B is Rs 20,000.

It might seem that it is not easy to compromise a large proportion of the database because it has required to know identifying characteristics of individuals people in order to build a tracker for that individual. This is not true.

It has been found that trackers are easy to find, perhaps with one or two queries. The tracker attack takes advantage of the fact that the query sets overlap. The query overlap also allows the database to be attacked by the use of simultaneous equations. If a database consists of a set of numbers, for example, x_1 , x_2 , x_3 and x_4 , and it is possible to submit queries about the sum of any subset, then a series of queries and replies can be obtained

Query 1	$(x_1 + x_2 + x_3)$	Answer: A_1
Query 2	$(x_1 + x_2 + x_4)$	Answer: A_2
Query 3	$(x_1 + x_3 + x_4)$	Answer: A_3
Query 4	$(x_2 + x_3 + x_4)$	Answer: A_4

The four equations may be solved to find any of the four unknowns, for example

$$x_3 = (A_1 + A_2 + A_4 - 2 \times A_2)/3$$

Adjustments for Statistical Inference Attacks

Denning and Schlörer have produced good survey techniques for maintaining security in databases. Here the controlling for all statistical attacks are identicals. Similarly, there are two different ways for protecting against inference attacks: the first one, Either controls can be applied to the queries and the second one controls can be applied to single items present in the database. As we observe that, it is very difficult to understand whether a given query will disclose sensitive data. So query controls are effective mechanisms to protect against direct attacks.

Suppression and concealing are two controls applied to data items. With suppression control, where sensitive data values are not be given; so the query is rejected without response. With concealing control, the answer is provided which is close to the actual value of data items but not exactly the actual value of the data items.

Limited Response Suppression

The n-item k-percent rule eliminates certain low-frequency elements from being displayed. It is not sufficient to delete them, however, if their values can also be inferred.

Table 6.11: Students by Hostel name and Students Gender

	Oval	Diamond	Opal	Total
M	1	1	1	3
F	1	2	1	4
Total	2	3	2	7

The data in such a table 6.11 suggest that the cell with counts of 1 ought to be suppressed; because their counts are much revealing. But it does not a good idea to suppress the Male Oval cell when the value 1 it can be determined by subtracting Female Oval (1) from the total (2) to determine 1, as shown in Table 6.12

Table 6.12: Students by Hostel name and Students Gender with Low count Supression

Table .Students by Hostel name and Students Gender , with Low Count Suppression.				
	Oval	Diamond	Opal	Total
M	-	-	-	3
F	1	2	1	4
Total	2	3	2	7

Combined Results

Another control adds rows or columns to guards against sensitive values. For example, the Table shows numerous sensitive results that find individuals items

Table 6.13: Students by Students Gender and students fees

. Students by Student Gender and Student fees.				
	Student fees			
Student Gender	2	1	2	3
M	1	1	1	2
F	1	2	2	0

These counts are combined with other results such as for example sum, it enables us to infer students free with the three males, in addition, to inferring that no feminine was graded 3 for payment of the fees. To curb such sensitive details, we are able to combine the values of the attributes for 0 and 1, and in addition for just two 2 which providing the less very sensitive results revealed in Table 6.14

Table 6.14 Supression by combining and realving values

Table Suppression by Combining Revealing Values.		
Students Gender	Students fees	
	0 or 1	2 or 3
M	2	3
F	4	2

Random Sample

With the help of random test control, we are an ability to derive the results from the entire databases; instead of the result would be produced with the help of computation on the random sample from the database. In this, the sample should be drawn must be large enough for being valid random numbers. Because the drawn sample shouldn't from the entire database, here a query from this sample won't necessarily match the results for the entire database. Thus, due to 5 percentage of the result for a specific query implies that 5 percent of these records have chosen from any sample such as o this query had the required property. You'll expect that around 5 percentage of the queries of the entire database could have the property involved, but the actual percentage could be quite different.

In order that averaging attack from repeated, equal queries should be prevented, the same test set ought to be chosen for comparative queries. In this manner, all equivalent queries will produce exactly the same result, although that final result will be just an approximation for the whole database.

Random Data Perturbation

Statistical databases always use random data perturbation (RDP) methods to protect against confidential numerical attributes that should be disclosed to an unauthorized user. One of the important parts of RDP methods is that they provide the appropriate level of security against snoopers who try to gain confidential information of attributes through statistical inference.

It is sometimes helpful to perturbed the value of the database by the small error values. for example, consider the variable x_i that is the correct value of the data items in the database so we can able to produce a small error value known as ϵ_i and add the value of x_i to its to produce the result. The value of ϵ is considered to positive and negative so some of the reported values are slightly greater than the actual value and other reported values are much lower than the actual value. Statistical measurements such as sum and mean will be close to the actual values but not necessarily to be the same. It much easier to use the data perturbation for generating the random sample selection because of it much easier to aim to order the ϵ values in order to produce the same result from equivalent queries

Query Analysis

A more complex type of security makes use of query analysis. Below, a query and its own implications are examined to find out whether an outcome should be presented. As noted early on, query analysis could be very complicated. One method involves retaining a query history for each individual and examine the query within the framework of what inferences happen to be possible given prior results.

Conclusion within the Inference Problem

You can find no perfect answers to the inference problem. The methods to control the inference problem to follow the three pathways as given below. The initial two methods may be used either to control limited queries that should be accepted or even to limit data that must be provided in response to a query. The final method applies only to data released.

- **Suppress obviously sensitive information.** This step can be obtained fairly easily. The tendency would be to err on the side of suppression, in doing so restricting the effectiveness of the database.
- **Track what the user knows.** Probably this safest method of disclosure, but this method is extremely costly, even though many users are not trying to obtain sensitive data. Information should be managed on all consumers, even though the majority are not attempting to obtain sensitive files. Moreover, this approach considers that what any two different people may know mutually and cannot tackle what an individual user can attain by using several IDs.
- **Disguise the data** Random perturbation and rounding can inhibit statistical attack that is determined by exact values for logical and algebraic manipulation. The users on the database receive just a bit incorrect or perhaps inconsistent results.

It is improbable that the study will reveal a straightforward, easy-to-apply measurement that determines specifically which data could be revealed without compromising sensitive data.

Aggregation

Related to the inference problem is aggregation, this means building sensitive outcomes from less very sensitive inputs. Consider an example, think about how police make use of aggregation usually in solving crimes: They determine who possessed a motive for committing the criminal offenses, when the crime was done, who acquired evidence from the crime site, who had the abilities to this crime, etc. Typically, you may think that police searching operation must start with the entire population and narrow down the search to an individual. If the police officers do the searching job in parallel, one may have a summary of probable suspects, another could have a list of possible motives, and another could have a summary of capable persons. Once the intersection of the lists is a single person, then the police have their prime suspect.

Responding to the aggregation problem is difficult since it requires the database management system able to track the result of each user who has already acquired and concealed any result that would allow the user to derive from the more sensitive result. Aggregation is particularly tough to counter since it can take place outside the system. For example, imagine the security policy is the fact that anyone might have sometimes the latitude or longitude of the gold mine, however, not both. Nothing stops you from receiving one, your friend from obtaining the other, and both of you talking to one another.

Recent interest in data mining raised concerns once again about aggregation. Data mining may be the procedure for sifting through several databases and correlating several data elements to get

useful information. Advertising and marketing companies use data mining extensively to get consumers more likely to buy a product or service.

6.6. Multilevel Databases

So far, we have seen data are divided into two classes: either sensitive or nonsensitive. We have seen that some data elements in the database are getting more sensitive than other data items, but we have allowed whether to access the data in terms of simply yes-or-no gain to the user. In this section, we could have to indicate the level of sensitivity as a procedure of an attribute, the column where the data appears, although nothing we've done depended on this interpretation of the level of sensitivity. Consider a Table 6.16, where two columns are usually identified (by shading) as sensitive data. Actually, though, sensitivity is to find out not only by attribute but additionally in ways that they are investigating.

Table 6.16 Attributes –Level Sensitivity

Attribute-Level Sensitivity. (Sensitive attributes are shaded.)				
Name	Department	Salary	Phone	Performance
Amit	training	43,800	022-2585679	A2
Bhavesh	research	62,900	022-2564897	D4
Saurabh	training	38,200	022-2956745	B1
Pramod	user service	54,600	022-2674523	A4
Chetan	user service	44,500	022-2543768	B1
Suraj	administration	51,400	022-2767834	A3

The Case for Differentiated Security

Consider databases containing data of Indian government expenditures. A number of expenditures happen to be for paper stationery, which is not considered as sensitive data. Some salary expenses fall under the sensitive data. Individual salaries of employees are sensitive, however, the aggregate of the salary isn't sensitive. Some data as expenses of military operations tend to be more sensitive; for instance, the total amount the Indian government spends for ballistic missiles, which could not be made public. There are few people in the government known the budget allocates for purchase of military equipment anything has spent on the military equipment is kept highly confidential.

Table listings employee's information. It could be a fact that some circumstances that Suraj is a temporary employee hired for a special project, and his entire record are different levels of sensitivity from another employee. In the table the mobile phone found for Pramod is kept as a

private line, it is unavailable to the general public. We are able to refine the level of sensitivity of the data by depicting it as displayed in Table 6.17.

Table 6.17 Attributes –Level Sensitivity

Attribute-Level Sensitivity. (Sensitive attributes are shaded.)				
Name	Department	Salary	Phone	Performance
Amit	training	43,800	022-2585679	A2
Bhavesb	research	62,900	022-2564897	D4
Saurabh	training	38,200	022-2956745	B1
Pramod	user service	54,600	022-2674523	A4
Chetan	user service	44,500	022-2543768	B1
Suraj	administration	51,400	022-2767834	A3

From this information, three attributes of database security have evolved.

- The security of an individual element could be not the same as the security of another element of the same records or from additional values of the same attribute. That's, the security of one data item of the database varies from that of other data items with the same row or column. This example implies that security should be implemented for individual data items.
- Two levels of sensitive and nonsensitive are insufficient to stand for some security reason. Several levels or modes of security could be required. These levels may represent certain amounts of understanding knowledge about security, it might overlap the security. Usually, the security grades form a lattice.
- The security of an aggregation of a sum, a count, or a grouping of and values inside a database may change from the security of an individual data element. The security on the aggregate values in the database could be greater or lower than that of the individual elements.

Granularity

Consider an example military document classification types are applied to paper documents and it was adapted to computers. It really is simple enough to classify and monitor an individual sheet of papers or, for example, a paper document, a computer file or one program or process. It is completely dissimilar to classify individual elements.

In some cases, a whole sheet of document is classified at one-degree level, even some particular words, such as and, the, or of, will be offensive in any context, along with other words, such as codewords like Pokhran project, might be delicate in some cases. But determining the sensitivity of every value in the database is comparable to applying a sensitive level to every individual word in the document.

And the problem can become more serious. The term Pokhran alone is not sensitive, in any context. However, the mix of these words generates a very sensitive codeword Pokhran project. A similar circumstance occurs in databases. Therefore, not merely can every part of a database own a distinct sensitivity, every mix of elements in the database may also have a definite sensitivity. In addition, the combination of the elements could be much sensitive than some of its elements.

So what exactly we are required in order to associate a level of sensitivity with each value of data items of a database? First of all, we are required to implement an access control mechanism to dictate which individual may have usage of what data. Usually, to carry out this control access mechanism over each data item is marked showing with limited access. Second, we should always guarantee that the data should not be access by unauthorized users. These two security requirements address both confidentiality and integrity.

Security Issues

We have seen the three principles of security issues such as integrity, confidentiality, and accessibility. In this section, we will further added the initial two of the concepts that play special roles for multilevel directories.

Integrity

We have seen in a single-level database where all elements possess the same amount of sensitivity, integrity could create a difficult task. Regarding the multilevel database, integrity can become a more crucial part and it is harder to achieve. Due to the access control mechanism, a program that reads high-level data is not permitted to write data at a lower level. Such a mechanism should be applied to databases, even so, this principle states a high-level user shouldn't be able to write a lower-level data element.

The problem with the above statement can arise once the DBMS should be able to read all records inside the database and create new records for just about any of the following needs: to acquire backups, to check out the database are able to answer the queries, to reorganize the databases in accordance with a user's needs, or even to update all records of the database.

When the user experiences this issue, they deal with it through the use of trust and commonsense. Individuals who have access to sensitive information from the database are more careful not to tell to unclear individuals or persons. In a processing system, you can find two options: Either the procedure solved at a higher level cannot write to the lower level or the program should be a "trusted program," the personal computer equivalent of an individual with security with a security clearance.

Confidentiality

Users trust a database provides accurate information, and therefore the data will be consistent and correct. It indicates prior, some means of guarding confidentiality may bring about a small change to the data. Even though these disturbances in the data cannot affect statistical analyses, they could produce two answers representing exactly the same underlying data values in reaction

to two differently created queries. Within the multilevel case, two different individuals running at two distinct levels of security could easily get two different answers to the same query. To maintain confidentiality, accuracy can be sacrificed.

Compelling confidentiality can produce results for unknowing redundancy. Let assume that specialist works at one level of access permission. The specialist recognizes that Ramesh is working for a company. However, in the retirement paper list, Ramesh's record does not appear. The specialist assumes that it can be human error and create a record for Ramesh.

The reason why that no report of Ramesh does not appear in the list because Ramesh is a secret agent, and his work with the company could be exposed public domain. A record of Ramesh should be present in the database file but, due to his special position, his record isn't accessible to the personnel specialist. The DBMS cannot dismiss the record from the specialist because doing so would disclose that there was already in this type of record with a sensitivity too much high for the specialist to see. The creation of the new record implies that there are two records of Ramesh is present in the database: one sensitive and one certainly not, as shown in Table 6.18. This example is named polyinstantiation, is known as one records can appear often, with different level of confidentiality each time every time.

Table 6.18 polyinstantiation records

Polyinstantiated Records.			
Name	Sensitivity	Assignment	Location
Ramesh	C	Program Mgr	Bangolre
Ramesh	TS	Secret Agent	Delhi

6.7. Proposals for Multilevel Security

In the previous section we have seen, applying multilevel security levels for databases can be difficult, probably more than applied security levels in the operating system, due to the smaller granularity of the data items or elements in the database has to applied to control mechanism. In this section, we study different methods for multilevel security for databases.

Separation

As we have previously seen, separation is essential for limit access. In this section, we will see how different procedure has implemented the separation in databases. Subsequently, we will see how this procedure can help implement multilevel protection for databases.

Partitioning

One of the important control mechanisms for the multilevel database is partitioning. In partitioning, the database is split into separate databases, each of the databases has own level of security. This mechanism is comparable to maintaining separate files in separate file cabinets.

This control mechanism does not provide a basic benefit of databases: removal of redundancy and better accuracy through possessing only one field to update. In addition, it does not address the problem of the high-level user who needs the usage of some low-level files coupled with high-level data.

Nevertheless, due to the difficulty of creating, maintaining, and employing multilevel databases, numerous users with data of blended sensitivities deal with their data through the use of the separate, isolated database.

Encryption

If the sensitive data will be encrypted, and end-user who accidentally will access it cannot read the data. Thus, each level of sensitive data could be stored in a database's table in encrypted form with a key unique to the amount of level of sensitivity. But encryption provides certain disadvantages.

First, an individual can selected plaintext attacks. Imagine gathering data like TOM or BIRD are stored in encrypted format in each document. A user who accesses this encrypted field can simply decrypt them by creating a new record and comparing the encrypted record with that element in all the records. More seriously, if authentication data happen to be encrypted, the illegitimate user can substitute the encrypted record of his / her own data with any user. This will provides access to the illegitimate user, but it also excludes the legitimate user individual who has authorized to access the authorized data are blocked by user malicious user.

Using a different encryption key for every record in the files may solve these problems. Each field of the record could be encrypted with different types of keys, or all fields of records could be cryptographically linked, much with cipher block chaining.

The disadvantage of this method is every field of the record should be decrypted when users perform standard database operations such as for example "select all records with income > 10,000." Decrypting the income field, also on rejected records, with increases in the time for processing query. Consequently, encryption isn't often employed to implement the separation in databases.

Integrity Lock

For a database, the lock is provides both integrity and constrained access. In U.S. Air Force Summer Study on Data Base Security, The procedure was initially nicknamed "spray paint" where each element is figuratively decorated with a coloring that denotes its sensitivity. The coloring is certainly maintained together with an element, not in a master database table.

A model of the basic integrity lock is shown in Table 6.19. As illustrated, each data item includes three parts: the specific data items itself, a sensitivity label, and a checksum. The sensitivity label defines the sensitivity of the data, along with the checksum is usually computed across both records and sensitivity labels to avoid unauthorized alteration of the data records or its content label. The actual records are stored in plaintext, for proficiency as the DBMS might need to inspect many fields when selecting a record to equivalent to a query.

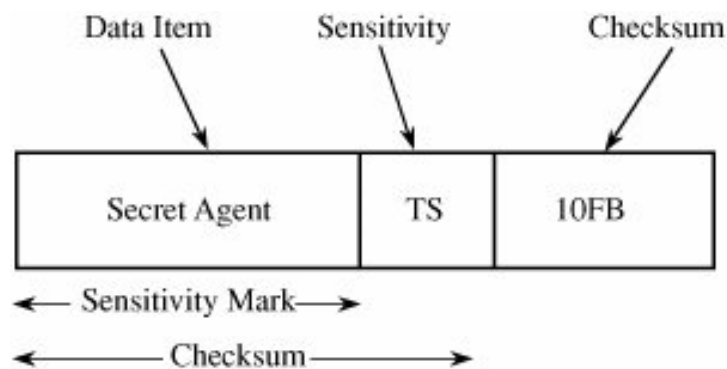


Table 6.19 A model of the basic integrity lock

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The sensitivity label might be

- Unforgeable, in order that a malicious object cannot develop a new sensitivity level for an elements
- Unique, in order that a malicious object cannot replicate a sensitivity level from another element
- concealed, in order that a malicious object area cannot even ascertain the sensitivity level of an arbitrary element

The third part of the integrity lock to get a field is an error-detecting code, known as a cryptographic checksum. To make sure that data values or its level of sensitivity classification are not modified, this checksum ought to be distinct for an element, and must incorporate both element's data values and connect that values to a particular position in the database. As shown in figure a proper cryptographic checksum consists of something distinct to the record, something original to this details field in the record the values of this element, and the amount of sensitivity classification on the elements. This four component against anyone's transforming, copying, or moving the data. The checksum could possibly be computed with a strong encryption algorithm or hashing technique.

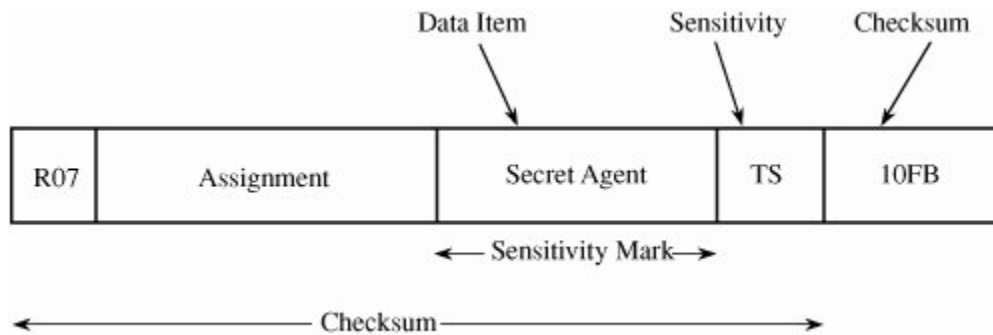


Figure 6.1 Cryptographic Checksum.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Sensitivity Lock

A sensitivity lock is really a combination of a distinctive identifier (as well as the sensitivity level. As the identifier is exclusive, each lock pertains to one particular document. A variety of factors shall have got exactly the same level of sensitivity. Malicious users shouldn't be in a position to identify two elements having equivalent levels of sensitivity or identical data values simply by considering the sensitivity level part of the lock. Due to the encryption, the lock's details, the sensitivity level especially, are hidden from plain text. Hence, the lock is usually connected with one specific document, plus the secrecy can be guarded because of it of the sensitivity level of that document.

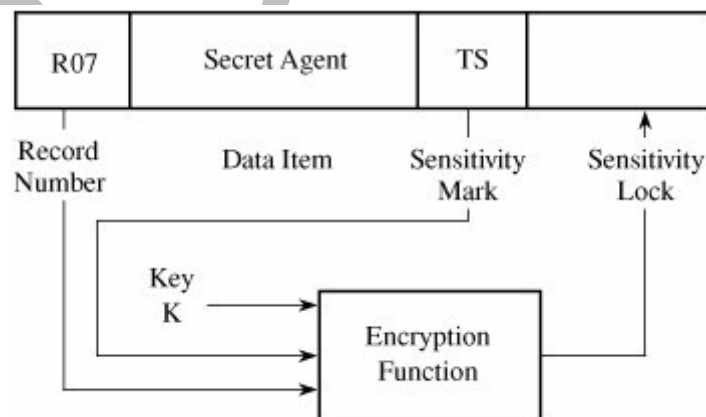


Figure 6.2. Sensitivity Lock.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Styles of Multilevel Secure Databases

This section provides different models for multilevel secure databases. These models show the tradeoffs among efficiency, flexibility, simplicity, and trustworthiness.

Integrity Lock

The integrity lock of DBMS had been created like a short-term solution for the security level for multilevel directories. The intention was initially in order to utilize any (untrusted) database administrator with a reliable method that deals with access management. The sensitive information had been destroyed or hidden with encryption that shielded both a data item and its own sensitivity. In this manner, only the gain access to the procedure would have to be trusted because only it might be able to attain or grant usage of sensitive data. The design of this type of system is proven in Figure 6.3.

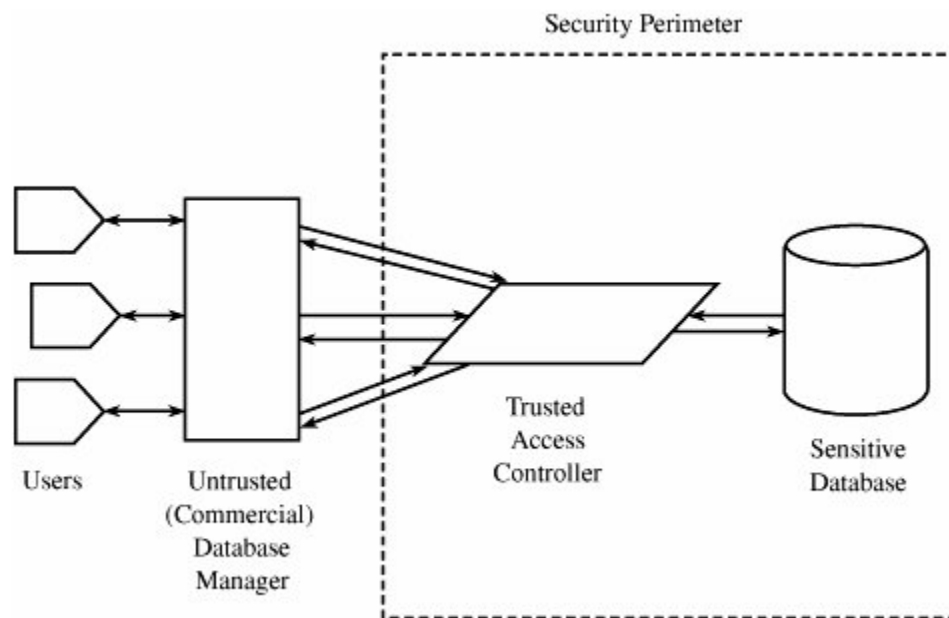


Figure 6.3. Trusted Database Manager.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The effectiveness of the integrity lock is a significant drawback. The memory is required for storing data items that should be extended to support the sensitivity label. Due to which there are many parts inside the label and something label for each and every element, a considerable amount of memory space is required.

There is also a problematic situation for processing time in the proficiency of the integrity lock. A data element is definitely passed to an individual to verify which the user's access data be allowed must be by sensitivity label. Also, whenever a value is usually created or modified, the label should be recomputed. Thus, significant processing time is used. In case the database document could be sufficiently safeguarded, the data items of an element could be in plaintext. That solution benefits select and project inquiries across multiple sensitive fields because data items need not get decrypted merely to determine whether it ought to be selected.

A final difficulty with this particular approach would be that the untrusted database manager sees all authorized data, so data is subject to Trojan horse attacks where data could be leaked through covert programs.

Trusted Front End

The diagram of a trusted front-end process is shown in figure 6.4. A trusted front end can be referred to as a guard and operates similarly to the reference monitor. It identifies that lots of DBMSs have already been constructed and placed into use without an account of multilevel security. Employee is been trained in using these DBMSs already, plus they may actually utilize them frequently. The front-end concept takes benefit of old tools and expertise, improving the security measures of the prevailing systems with minimal change to the system. The interaction between a user's, a trusted front end, including a DBMS involves the next steps.

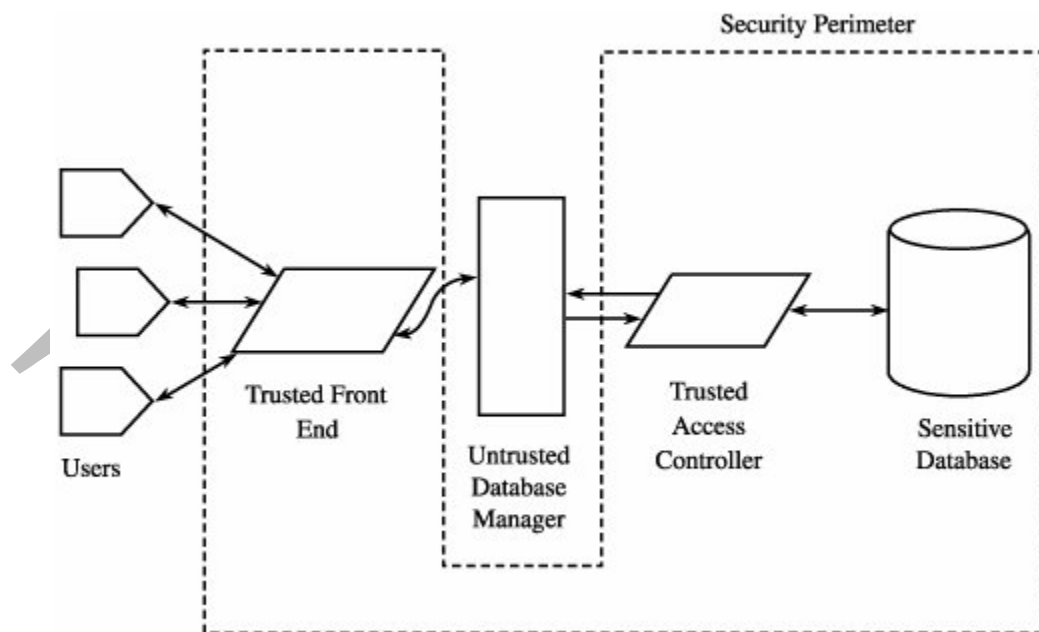


Figure 6.4. Trusted Front End.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

1. A user recognizes himself or herself to the front end; the front end authenticates the user's personal information.
2. The user issuing a query to the front end
3. The front end verifies the user's authorization to data.
4. The front end issues a query to the database manager.
5. The database administrator performs I/O operation, getting together with a low-level access handle to achieve the usage of actual data.
6. The repository supervisor results in the full total consequence of the query for the dependable top ending.
7. The front conclusion analyzes the level of sensitivity levels of the info items in the effect and selects those products in keeping with the user's safety measures level.
8. The front stop transmits selected information towards the untrusted front finish for formatting.
9. The untrusted prominent ending transmits formatted files to an individual.

The trusted front side end functions as a one-way filtration, screening out benefits the user shouldn't be able to access. But the design is certainly inefficient because possibly much data is certainly retrieved and discarded as unacceptable for an individual.

Commutative Filters

The idea of a commutative filtration system was suggested by Denning as being a simplification on the trusted interface towards the DBMS. Basically, the filter monitors the user's need, reformatting it if required, so that sole data of a proper sensitivity level will be returned to an individual.

A commutative filter is a process that creates an interface between the users along with a DBMS. Nevertheless, unlike the trusted front end, the filter attempts to capitalize on the efficiency of all DBMSs. The filtration system reformats the query so the database manager does indeed just as much of the task as possible, screening out many undesirable records. The filtration system then offers a second screening to choose only data that your user has access to.

Filters may be used for security with the record, attribute, or element levels.

- When used on the record levels, the filter demands requested data plus cryptographic checksum info; after that, it verifies the accuracy and availability of data to get passed to an individual.
- At the attribute level, the filtration systems check whether all attributes inside the user's query will be accessible to an individual and, if that's the case, it goes by the query for the database administrator. When return, it deletes all fields' areas to which the user does not have any access rights.

- At the element level, database systems requests desired data plus cryptographic checksum details. When they are returned, database systems will check the classification level of every element of every record retrieved contrary to the user's level.

Suppose several nuclear scientists in DRDO Delhi perform on very sensitive projects, therefore the current user shouldn't be allowed to obtain the nuclear scientist's name in the databases. This restriction offers a problem with this particular query:

retrieve NAME where ((OCCUP= nuclear scientists \wedge (CITY= Delhi))

Suppose, also, that the existing user is prohibited from understanding anything about any individuals in Kolkata. Utilizing a typical DBMS, the query might obtain all records, along with the DBMS would after that pass the outcomes to the user. However, once we have seen, an individual could probably infer reasons for having Kolkata employee or Delhi nuclear scientist focusing on secret jobs without even being able to access those fields directly.

The commutative filtration re-forms the original query in a trustable way in order that sensitive information will be never retrieved from the database. Our trial query would become

retrieve NAME where ((OCCUP= nuclear scientists) \wedge (CITY= Delhi))

from all records R where

(NAME-SECURITY-LEVEL (R) \leq USER-SECURITY-LEVEL) \wedge

(OCCUP-SECURITY-LEVEL (R) \leq USER-SECURITY-LEVEL) \wedge

(CITY-SECURITY-LEVEL (R) \leq USER-SECURITY-LEVEL))

The filter functions by restricting the query for the DBMS and restricting the outcomes before they're returned to an individual. In this situation, the filtration would request Label, NAME-SECURITY-LEVEL, OCCUP, OCCUP-SECURITY-LEVEL, CITY, and CITY-SECURITY-LEVEL worth and would then simply filter and go back to the user just those job areas and items which are of a secrecy degree acceptable for an individual.

Distributed Databases

The distributed or federated database is a 4th design for a secure multilevel database. In this, a known front end user controls usage of two unmodified commercial DBMSs: one for low-sensitivity data and one for high-sensitivity data.

The front ends require to takes a user's query and formulates into single-level queries to the database as an appropriate query. For any user has to remove for high-level sensitivity data, the front end users submit queries to both high- and low-sensitivity database. If the user isn't removed for high-sensitivity data, the front end submits a query to simply the low-sensitivity database. If the results come from only the back-end database, then the front end passes the outcomes back to an individual. If the outcomes originate from both databases, the front end must combine the outcomes appropriately. For instance, when the query is really a join query

getting some high-sensitivity data and some very low data, the front end has to do the same as a database join itself.

Distributed database design is not popular because the trusted front end is complex and may contain most of the functions of the entire database itself. Additionally, the design does not scale well to various levels of sensitivity. Each level of data sensitivity must be managed in a separate database.

Window/View

One of the superiority of using a DBMS for many users with different interests with the different levels of sensitivity is the ability to create different views for each individual user. That is, each individual user is limited to access data that only represent what the user needs to be view. For example, the registrar can only view class assignments and grades from each student at the university without having to display extracurricular activities or medical records. On the other hand, the university health center needs medical records of the students but not the results of standard academic records of the students.

The window or a viewer's perception can also be foundationally structured for a multilevel database to get access. A window or view is a subpart of a table in the database, containing exactly the data that a user is able to access. Denning surveys the builds of views for multilevel database security.

Views can be the database of a user subset, so each user's queries are included in the simplest database. This subset ensures that users no longer access to values that exceed permitted values because invalid values don't even exist in the user's database. The views are exactly specified as a set in a database relationship, so the data in the view is subset change when the data are changed in the database

One mechanism that can be used to implement security is the SQL view. Views allow you to limit the data available to users, the types of operations that users can perform through views or both.

Consider the following DDL-SQL statement, which can be accepted in all three major RDBMS implementations:

```
CREATE VIEW v_customer_status (  
    name, status ) AS SELECT cust_name_s, cust_status_s FROM  
    customer
```

In this view, only two fields are selected from the CUSTOMER table, which contains a total of seven fields. This is called vertical restriction because access to a subset of columns (fields) is restricted. Other fields may contain confidential information that should only be available for management. If you grant SELECT privileges for the role view (for example, Employee "ROLE"), anyone in that role can see the customer 's name and status while the rest of the information in the table remains inaccessible.

If the SELECT statement from view V_CUSTOMER_STATUS is executed by an authorized person, this leads to the following result (the same statement issued by a person who has no privileges, however, causes an error):

```
SELECT * FROM v_customer_status
```

```

name status ----- HOME SEAL
CORP. Y DATAMATICS INDIA INC. Y RADIOMETRIC DEVICES INC. Y . .
.....
HLGO SWITCHBOARD INC. N

```

You can also restrict access horizontally by specifying a subset of rows. For example, you might want to provide access to historical data entered in a table a year or earlier and prevent access to data added after that date. or, using the example in the ACME database, say that you have salespeople who are divided into two groups based on their responsibilities and experience (one group has dealers dealing with customers with orders over 15,000, and another group handles customers who generate less volume). In the last example, the SQL syntax for three databases is as follows:

```
CREATE VIEW
```

```
v_customer_totals_over_15000 AS SELECT * FROM v_customer_totals WHERE
total_price > 15000
```

If you choose from the following views, only records for customers with a total of more than 15,000 are displayed:

```

SELECT * FROM
v_customer_totals_over_15000 customer_name order_number total_price
----- HOME ELECTRONICS INC.
523735 15613.60 WILE BESS COMPANY 523741 15464.10 .....
WILE
SEAL CORP. 523775 15613.60 HOME SEAL CORP. 523781 15464.10

```

The V_CUSTOMER_TOTALS_OVER_15000 view is based on a different view, V_CUSTOMER_TOTALS, and different permissions can be set for each of these objects. With this method, a sophisticated and refined security hierarchy can be created.

Of course, the horizontal and vertical selection options can be combined in one view.

Views can also limit the data displayed to summary information, e.g. in V_CUSTOMER_TOTALS, where sales information is summarized and grouped by order number.

Additional restrictions that can be applied to views include WHERE clauses and JOIN conditions. They are useful when there is more than one table in view. For example, you can limit your display to only showing customers who have placed an order and hiding everything else as follows (syntax applies to all three databases):

```
SELECT DISTINCT cust_name_s FROM
    Customer cu JOIN order_header oh ON cu.cust_id_n = oh.ordhdr_custid_fn
    CUST_NAME_S ----- HOME BESS
    COMPANY WILE DATAMATIC INC. WILE ELECTROMUSICAL INC. . . . .
WILE
    ELECTRONICS INC. HOME SEAL CORP.
```

The view is not only used for SELECT, but also for UPDATE, INSERT, and DELETE statements. Some of these operations are controlled by the inherent property of the display object, others can be specified when creating the display object. For example, you cannot update or insert a view that was created with an aggregate function. If you try this, an error will be generated. This is an inherent behavior. On the other hand, you can create limits for updated views that can accept or reject data changes based on certain criteria

6.8. Data Mining

Databases are a huge collection of achieving and current data. More data are being gathered and stored in storage disk of size may be in petabytes. Three decades ago, the sharing of databases by individuals through the networks and the Internet was unimaginable. However, discovering useful data in that tremendous amount of data requires smart intelligent technique and querying of data. Without a doubt, finding important data is a specialized technique, called data mining, has developed. In a generally computerized manner, data mining applications are sorted and search throughout the data.

.Data mining utilizes statistics, machine learning, statistical models, pattern recognition, along with other techniques to find out patterns and relationships on large datasets. Data mining tools are used for association in which one event follows with another event), sequences in which one event leads to other events), clustering in which some data items have similar attributes and forecasting in which past events forecast future events. The comparison between a database and a data mining application is getting obscured; you can presumably perceive how you could execute these queries in a conventional database. For the most part, database queries are manual, though data mining is more programmable. You could build up a database queries to perceive what different items are purchased by individuals who purchase cameras and you may see a dominance of MP3 players in the outcome, yet you would need to see that relationship yourself. Data mining tools would display the noteworthy connections, among cameras and MP3 players,

yet in addition among bagels, airline tickets, and sports shoes. People need to break down these correlations and figure out what is more significant.

Computer security gains from data mining. Data mining is extensively used to analyze system data, for example, audit logs, to discover patterns associated with attacks. Finding the systems or programs to an attack can assist enhance good prevention tools and techniques, and seeing the actions related to an attack can assist pinpoint vulnerabilities to manipulate and harm that might also have occurred.

In this section, however, we choose to have a look at the security problem involving data mining. In confidentiality, integrity, and availability provides us hint to what these security issues can occur. Confidentiality mainly starts with privacy however additionally include proprietary and commercially confidential data and protection of intellectual property: How do we manage what is disclosed or derived? For integrity the necessary problem is correctness data can be useful or useless and cause potentially damaging, however, we want to investigate how to measure and make certain correctness. The availability consideration relates to both performance and structure: Combining databases no longer firstly designed to be blended effect whether consequences can be acquired in a well-timed manner or even at all.

Privacy and Sensitivity

Aim of data mining is summary results, not individual records items, you would now not expect a hassle with a sensitivity of character statistics items. Unfortunately, that is not true.

Individual privacy can go through from the equal sorts of inference and aggregation issues we studied for databases. Because privacy, in particular defending what an individual considers non-public information, is a vital subject matter that relates to many areas of computer security,

There have been little studies on the sensitivity of data received from data mining. Clifton has investigated the problem and proposed technique that might produce near but not specific combination results that would forestall revealing sensitive data.

Data Correctness and Integrity

"Connect the dots" refers to the child's game wherein discrete data points (i.e., numbered dots) are converted into a single image through drawing links between the dots in the right order, accordingly creating information (the image) from the data (the dots). Connecting the dots has to turn out to be a metaphor for coming across the "big image" from reputedly unrelated facts.

But earlier than we are able to connect dots, we want to do two other critical things: collect and accurate them. Data storage and current technology are making it viable to acquire more dots than ever earlier than. But if your name or address has ever seemed incorrectly on a mailing list, you realize that no longer all accumulated dots are accurate.

Correcting Flaws in Data

Let's take the mailing listing for instance. Your company vendor at ray road brought a catalog for kitchenware to you at 519 Pali Street with your name however address 510 as opposed to 519; surely a person made a mistake coming into your address. You contact the kitchen supply vendors, and they are willing to change your address with on their records because it's their interest to send catalogs to people that are inquisitive about them. But they bought the kitchenware to your name and address in conjunction with others from a mailing list, and they haven't any incentive to contact the list owner to correct your address record. So additional catalogs hold to show up together with your neighbor name and address. You can see in which this story leads to mistaken addresses in no way change.

Data mining makes a problem in this situation. Databases need distinct keys to greatly help with structure and searches. But different databases may not have got shared keys so that they use some data field as though it were a key. In our instance, this shared data field may be the address, hence right now your neighbor's deal with is connected with cooking.

Nonetheless consider terrorists. The intelligence service of a government gathers data about suspicious activities. But the names of suspects are foreign, written in a different alphabet. When translated into the alphabet of the government the transition is irregular: one agent writes "Moe," another "Mo," and another "Mho." At best, it is difficult to try to use these names as common keys. Phonetic is one solution. You cluster a similar sound. However, in that case, "So," "Sho," "Hoe," and "Lsiao," too, could be brought into the terrorist search involving innocent men. There are always two problems if a human analyst could properly separate all of these and wanted to correct Moe / Me / Moh databases. Second, the researcher may not have access to other agencies' original databases. Even if the analyst could access the originals, it was likely the analyst would never know where the original databases had been copied else.

One important objective in databases is to have a record in one location so that a correction can be used for all times. The result is an aggregate of several databases with data mining. From the resulting to the amalgamated databases, no easy way to work backward to classify and fix errors is found.

Using Comparable Data

Data semanticization is another important consideration in data mining. Take into account two geo-databases with family income info. The other database has revenue in thousands of Rupees, except one database has an income by Rupees. While field names are similar it would lead to poorly skewed statistics by combining raw data.

In one database, the rated attribute has a numerical scale from 1 to 5, and another database has an attribute that has high/medium/low rated. Is it necessary to treat high/medium/low as 1/3/5? Even if analysts use the transformation, the consistency of the results is decreased by using computing with some 3 points and 5 points accuracy. Or how can you merge one database with another that does not have a special attribute?

Eliminating False Matches

Coincidence, as we mentioned before, is not causation or cause. When two things happen together, it does not mean any cause. Data mining attempts to highlight unknown data connections, but often fuzzy logic is used to identify such connections for data mining applications. Both methods should produce false (false matches) positive and missing (false negatives) relations. We must be alert to the inherent inaccuracy of Data Mining approaches and we must protect ourselves from putting too much faith in the performance of a data mining application simply because "the machine has said that."

Availability of Data

The third security issue for data mining is the interoperability of different databases. As we just mentioned, the structure and semantic consistency of databases must be possible to exploit the data. Missing or incomparable data will incorrectly result in data mining, so a better alternative is perhaps not to produce a result. But there is no correlation finding which is not the same. Data mining systems have to fix several sensitivities, like with single databases. In trying to merge attribute databases with more sensitive values, there can be no data and therefore no match.

6.9 Reivew Question

1. How can indefinite postponements occur in an environment in which several users share access to a single database? Describe a scenario that could allow two users to postpone each other indefinitely. Describe a scenario where a single user may postpone all users indefinitely
2. Using the commit two-step provided at the start of this chapter, illustrate, as in the example of the railway researavtion, how to avoid seating one seat for two people.
3. UNDO is a database retrieval method. It is a command that receives information from a transaction log and resets the data base elements to its values in advance of a particular transaction. Describe a situation that would be beneficial with a UNDO request.
4. Suppose that a database manager allowed one transaction to be nestled inside another. It ensures that the DBMS would require you to choose another record after you have updated one record and then to restore it to the first record. The first record is updated. What would be the effects of nesting on database integrity? Propose a mechanism that allows nesting.
5. Can a database have two identical records without having a negative impact on database integrity? Why not? Why or why not?
6. Explain the inconvenience of partitioning in order to implement multilevel database security
7. In a multilevel, secure database management system how does encryption work?

6.10 References

1. Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger - RAND Corporation Publisher: Prentice Hall
2. Cryptography and Network Security - Principles and Practice fifth edition Stallings William Publisher: Pearson
3. Cryptography And Network Security 3rd Edition behrouz a forouzan and debdeepmukhopadhyay 3/E Publisher: McGraw Hill Education
4. Cryptography and Network Security, 3e AtulKahate Publisher: McGraw Hill

Chapter 7. Security in Networks

7.0 Introduction

7.1. Network Concepts

7.2. Threats

7.3. Network Security Controls in Networks

7.4. Firewalls

7.5. Intrusion Detection Systems

7.6. Secure E-Mail

7.7 Review Questions

7.8 References

7.0 Introduction

Networks their design, expansion, and usage are significant to our design of computing. We connect to networks daily, whenever we perform banking transaction, make calls, or drive trains and planes. The electricity companies use systems to track electric power or water utilization and bill for this. When we purchase groceries or gas, networks permit our credit card or debit card deals and billing. Living without networks will be considerably less practical, and many things to do would be difficult. Not surprisingly, after that, computing networks will be attackers' targets of preference. For their actual and possible impact, network problems attract the eye of journalists, supervisors, auditors, and everyone. For example, once you read the everyday newspapers, it's likely you'll find a storyline in regards to a network-based attack at the very least on a monthly basis. The insurance policy coverage itself evokes a feeling of evil, employing terms such as for example hijacking, spread denial of service, and our common friends Trojans, worms, and Trojan horses. Because any large-scale episode will probably put a large number of computing systems at an increased risk, with potential deficits well in to the huge amount of money, network attacks create good copy.

In this section we describe why a network is much like and various from a credit card application system or an operating-system that you've studied in previous chapters. In looking into networks, become familiar with how the principles of confidentiality, integrity, and accessibility apply in networked settings. At exactly the same time, so as to the essential notions of id and authentication, accessibility control, accountability, and guarantee are the schedule for network security.

Networking keeps growing and changing maybe even faster than various other computing disciplines. Subsequently, this chapter is definitely unlikely to provide you with current technology, the most recent attack, or the most recent defense mechanism; it is possible to find out about those in everyday newspapers with web sites. However the novelty and alter build on which we know right now: the essential concepts, dangers, and settings for systems. By developing a knowledge of the fundamentals, you can soak up the most existing news efficiently. Moreover, your understanding can help you in building, safeguarding, and applying networks

7.1. Network Concepts

Networks are flexible as well as strong. Think about the electricity, cable TV, telephone, or water network that serves your home, to see that most. If the power line to your home is broken by a falling tree branch, you are without electricity until that line is repaired; you are prone to what is called a single point of failure, since electrical infrastructure for your entire home is disrupted by one cut to the network. Similarly, there could be one telephone trunk line or water main that serves your home and others nearby; without service, a failure will leave your house, street, or neighborhood. But we've got ways of stopping the whole network from failing. If we trace the origins of what flows through it back through the network from your house, we are likely to find that a whole city or university is served by many key distribution lines. That is, there is more than one way to get to your neighborhood from the source, allowing engineers to direct the flow along alternate routes. For an entire city, redundancy makes it rare to lose infrastructure from a single failure. We claim that such a network has durability or fault tolerance for this purpose.

The flow not only around failures but also around overloaded parts is redirected by complex routing algorithms. Usually, the routing is performed automatically; human supervision or involvement also strengthens the control software. By design, not by mistake, many forms of networks have very high level of reliability. But since there is always less redundancy at the endpoints of a network than elsewhere, we conclude that in the Centre, the network has great strength and perimeter fragility.

A network is often built from the user's viewpoint, so that it appears like two endpoints with a network connection in the middle. For example, there can seem to be nothing more to the municipal water supply than a reservoir (the source), the pipes (the means of transmission or communication) and your water faucet (the destination). Although this simplistic view is functionally accurate, the complex architecture, implementation, and management of the 'pipes' are neglected. In a similar manner, in this chapter, we explain computer networks in terms that concentrate on security principles but introduce the networks themselves in a simplistic manner, to highlight the importance of security and prevent our attention from being diverted by the complexity of the networks. Please note that our representations of the network are always abstractions of a more complicated fact.

The Network

Figure 7.1 indicates a system in its simplest type, as two products linked across some moderate by components and software program that allow the communication. In some instances, one device is really a computer (in some cases known as a "server") and another is really a simpler machine (sometimes known as a "client") empowered just with some method of input (like a keyboard) plus some means of output (like a screen). For instance, a powerful laptop could be a server, but a handheld individual digital helper (PDA) or perhaps a cell phone may be a network customer. Actually, because more buyer devices have become network-enabled, network security and safety issues will continue steadily to grow.



Figure 7.1. Simple View of Network.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

While a basic network is described by this model, the actual situation is often considerably more complicated.

- A PC or workstation is often the simpler client device, used for user-to-computer communication, so the client has significant storage and processing capability.

- You can configure a network as just a single client that is connected to a single server. But more generally, with multiple servers, many clients communicate.
- Many computers also provide the services of the network. As the communication of an individual user passes back and forth from client to server, it may merely move through certain computers but pause for essential interactions with others.
- The end user is normally unaware of many of the communications and computations on the user's behalf that take place in the network.

In a network, a single computing system is sometimes referred to as a node and its processor (computer) is called a host. A connection is defined as a communication between two hosts. Network computing consists of users, communications media, visible hosts, and end-user devices that are not normally visible. Systems 1 through 4, in Figure 7-2, are nodes. The users are on the lettered client machines in our figure, maybe communicating with Server F.

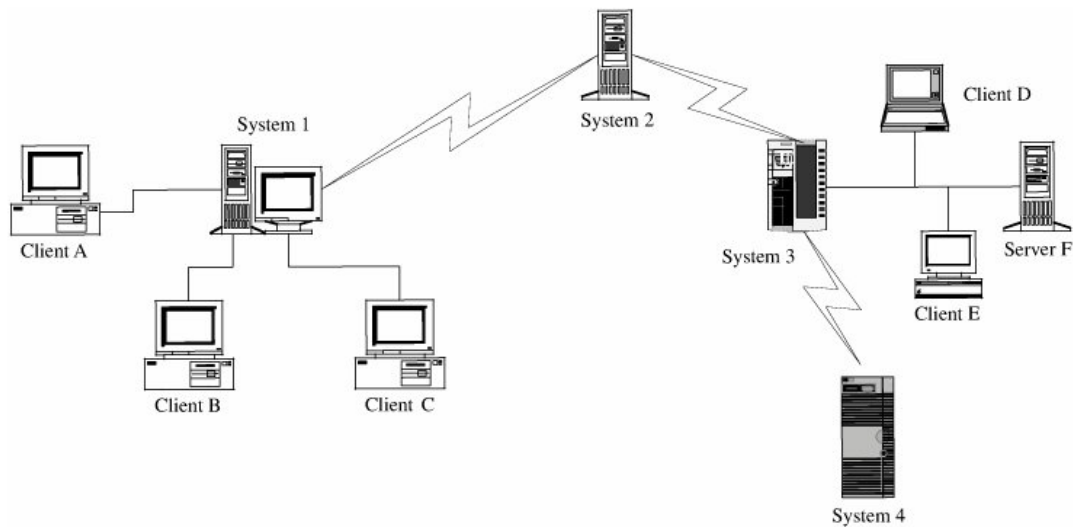


Figure 7-2. More Complex but More Typical View of Networks.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Networks could be described by more than a few typical attributes:

- **Anonymity.** You would have seen the cartoon picture that shows a dog typing at a workstation and telling another dog, "On the Internet, nobody knows you're a dog." A network eliminates most of the clues that we identify acquaintances by, such as presence, speech, or background.
- **Automation.** In certain networks, machines with only limited human control may be one or both endpoints, as well as all intermediate points, involved in a given communication.

- **Distance.** Several networks connect devices that are physically far apart. However not all network connections require distance, communication speed is high enough that people are usually unable to tell whether a remote site is close or far away.
- **Opaqueness.** Since the distance's dimension is secret, users can not say whether a remote host is in the next door room or in a different region. Users do not differentiate, in the same way, whether they are connected to a node in an office, school, home, or warehouses, or whether the computer system of the node is large or small, modest or strong. Users can not necessarily say if the new contact involves the same host they interacted with the last time.
- **Routing diversity.** To keep up or improve dependability and overall performance, routings between two endpoints are often dynamic. That's, the same connection may carry out one path from the network the very first time and an extremely different path the next time. Actually, a query might take a different journey from the reaction that follows a couple of seconds later.

Size and shape

Just how a network is usually configured, with regards to nodes and connections, is named the network topology. It is possible to think about the topology because the form of the network

Both of these extremes highlight three proportions of networks which have particular bearing over a network's security.

- **Boundary.** The cap distinguishes those from an outside part of the community. We can easily list all the components for a simplistic network and get an invisible straight line around it to split up what is inside the network from what is outside. But it's typically practically impossible to list all the hosts connected to the Internet. For example, a line bordering the Internet will need to surround the complete globe right now, and online connections also go via satellites in orbit round the earth.
- **Ownership.** It's always hard to see who owns a network host. The network administration organization, including cable and network appliances, will own the network infrastructure. However, for convenience, such hosts may be connected to the network, not actually owned.
- **Control.** In the last example, control must also be regulated if ownership is unknown. Choose an arbitrary host to see how. Does it belong to network A? If yes, is the network administrator managed by A? Does this administrator set network access control policies, or decide when and to which version of its software? Is the administrator also aware of which software version the host is running?

Mode of Communication

The communication between two end points is achieved in a computer network. Data are transmitted by either the digital (with discrete binary values for data items) or the analogue formats (where the data objects are presented with the medium of sound and electrical voltage as points within a continuous spectrum.) Computers usually store and process digital information, but telephone and related cables are analog form. Since phones were initially designed for voice transmission. If analogue data is to be transmitted from the transmitting medium, the digital signals must be converted to analogue for transmission and then to digital for computation at the receiving end.

Media

Various media methods of communication are permitted. We are able to choose from different forms like copper wires or fibre optics, or air-like mobile phones. Let's take a look at any form.

Cable

Because a lot of our computer interaction has historically long been done over mobile phone lines, the most frequent network communication method today is cable. Inside our homes and office buildings, we use with a couple of insulated copper wiring, known as a twisted couple or unshielded twisted match (UTP). Copper has got good transmission attributes at a comparatively low priced. The bandwidth of UTP is bound to under 10 megabits per 2nd (Mbps), thus engineers cannot transfer a lot of network simultaneously about the same line. Additionally, the signal power degrades since it travels throughout the copper wire, also it cannot travel extended distances without a boost.

Another preference for network communication is definitely coaxial (coax) cable connection, the kind utilized for cable TV. Coax cable will be constructed with an individual wire encircled by an insulation coat. The jacket can be itself surrounded by way of a braided or spiral-wound line. The inner cable carries the transmission, and the exterior braid functions as a soil. The most trusted computer conversation coax cable is usually Ethernet, carrying around 100 Mbps over ranges as high as 1500 feet.

Coax cable in addition is suffering from degradation of sign quality over range. Repeaters (for electronic indicators) or amplifiers (for analog indicators) could be spaced periodically across the cable to get the indication, amplify it, take away spurious signals known as "sound," and retransmit it.

Optical Fiber

A newer type of cable is constructed of very slim strands of glass. Instead of holding electricity, these fibers bring pulses of lighting. The bandwidth of optical fiber content is around 1000 Mbps, plus the signal degrades much less over fibers than over line or coax; the fibre is good for a run of around 2.5 miles. Optical fiber entails less interference, much less crossover between adjacent multimedia, less expensive, and less weight than copper. Therefore, optical fiber is normally a far greater transmission channel than copper. Therefore, as copper ages range, it is staying changed by optical fibers in most connection systems. Specifically, most long-distance communication lines are actually fiber.

Wireless

Communications can also be transmitted by radio signals. Wireless radio could also be used in network, in accordance with the protocol for short range telecommunications, designated as the Standard 802.11, including pagers, wireless microphones, garage door openers, and handheld telephones. This wireless medium is particularly useful for networks where nodes are physically

connected, such as an office or home building. Many 802.11 applications for home and office wireless networks are now available.

Microwave

Microwave is an outdoor communications form of radio transmission that is especially appropriate. The channel capability of Microwave is comparable to that of the coax cable, i.e. it carries similar data. The biggest advantage is the strong signal from transmitting point to receiving point. There is, however, no need to regenerate microwave signals with repeaters, like signals on the cable. After all, there is a straight line in the microwave signal, which is troublesome since the Earth curves. The microwave signals pass by line of sight. There are no sporadic obstacles, like mountains, that must be in a straight line between the transmitter and receiver. As seen in Figure 7-3, because of the Earth's curvature a straight microwave signal sent from the height of a tower can only travel about 30 miles. Thus, microwave signals from recipient to recipient are "bounced" to span longer distances, spacing less than 30 miles apart.

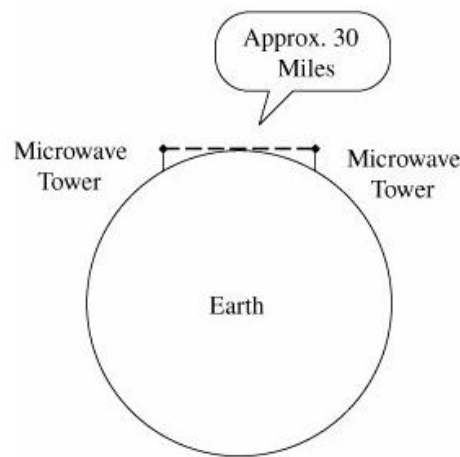


Figure 7.3. Microwave Transmission.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Infrared

Infrared communication carries signals (up to 9 miles) for short distances, and also needs a clear sight line. Since cabling is not needed, portable objects, including laptop computers and peripheral connect to peripherals. A point to point signal makes it impossible to intercept an infra - red signal. The interceptor works, like a repeaters, collects a signal, removes any desired data and transmits the original signal or updated version to the original target. It is, however, vulnerable to "in the middle" attacks. Infrared is usually used in a secure environment such as an office in which

In-the-middle attacks are impossible to hide because of the line-of-sight criteria and limitations on distance.

Satellite

Many communications would cross the globe, including International telephone calls. Early in telecommunications technologies, telephone companies maintained vast cables through the ocean floor allowing calls to travel from one continent to another. We now have other choices. Satellites are positioned in orbits by communication companies aligned with earth's rotation (called geosynchronous orbits), meaning the satellite seems to be hovering 22,300 miles above the earth in a fixed location. Satellites behave like naive transponders: they retransmit whatever they receive. Therefore the satellites are highly advanced receivers that transmit and reproduce signals as their sole feature. In the user's experience, the signal "bounces" off the satellite and back to earth. For instance a North American's signal travels 22,300 miles to the sky and back to one point in Europe on the same distance. Figure 7-4 illustrates the mechanism of bouncing off a signal from a satellite

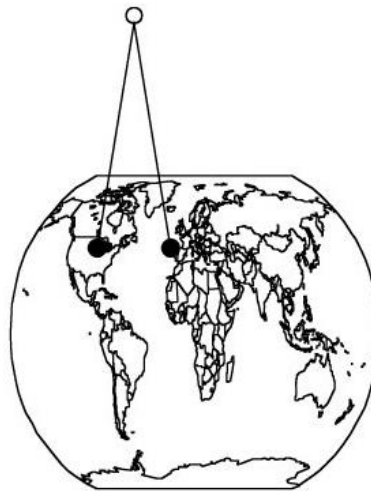


Figure 7.4. Satellite Communication.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Protocols

Typically the media are opaque to all of us when we use a network. In other words, most of us don't know if our communication is carried by copper cable, fibre optics, satellite or microwave. Actually, a communication medium will change from transmission to transmission. In reality, this complexity is a positive characteristic of a network: its independence. In other terms, the communication is split from the existing communication channel. Independence is feasible since we have specified protocols to allow a user to view the network on a high, abstract level of communication (user and data); the details of communications conducted are concealed from both software and hardware. The software and hardware help us to build up a network for a layered communication architecture according to a protocol stack. Each layer in the stack is somewhat similar to a language that communicates relevant information.

ISO OSI Reference Model

The International Standards Organization (ISO) Open Systems Interconnection model consists of layers by which a network communication occurs. The OSI reference model contains the seven layers listed in Table 7-1.

Table 7-1. OSI Protocol Layer Levels.		
Layer	Name	Activity
7	Application	User-level data
6	Presentation	Standardized data appearance, blocking, text compression
5	Session	Sessions or logical connections between parts of an application; message sequencing, recovery
4	Transport	Flow control, end-to-end error detection and correction, priority service
3	Network	Routing, message blocking into uniformly sized packets
2	Data Link	Reliable data delivery over physical medium; transmission error recovery, separating packets into uniformly sized frames
1	Physical	Actual communication across physical medium; individual bit transmission

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Figure 7-5 shows how communication operates through the various layers. We may think of the layers as constructing an assembly line through which the communication is supplemented by each layer. The layers together address the different tasks to be carried out to deliver the message effectively. Each layer has a purpose separately; for senders and receivers the equivalent layers carry out identical functions. For eg, the four layer of the sender applies a message header, which indicates the sender, the receiver and the sequence information. Layer four reads the headers at the

receiving end to check if the message is for the recipient and then deletes the header

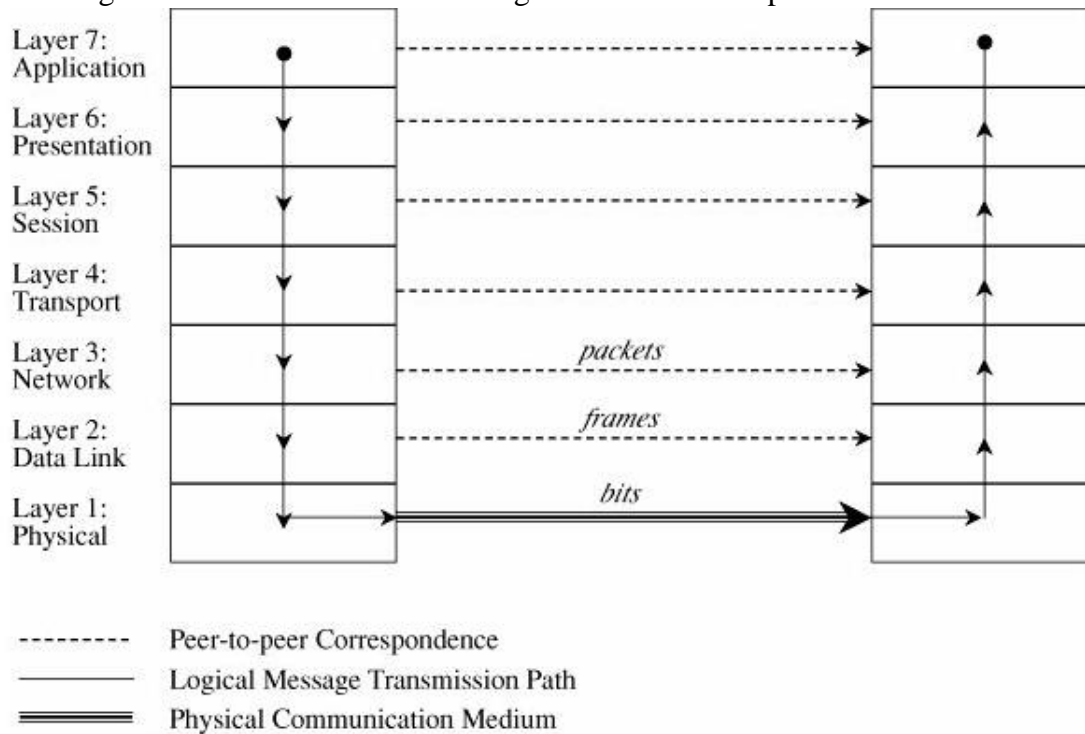


Figure 7.5. ISO OSI Network Model.

Above Image taken from the book Security in Computing, Fourth Edition by Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Each layer transfers data into 3 ways: above a layer that communicates in another host more intuitively, concurrently or over to the similar layer, and below with less abstract (i.e. more conceptual) data objects handling layers. The above and the below communications are direct interactions, while the parallel is a virtual path. "peers." Were called parallel layers.

Consider an example of a simple transmission of protocols. Suppose you are run a programme like Eudora, Outlook or Unix mail to deliver an email to your friend. Using an application editor you enter a message, as well as the application format the message in 2 components: a header which indicates the person to whom the message is meant and a body containing the message's content. The application reformats your letter in a standard manner, so that you can still share emails even though you and your friend are using multiple mail applications. Figure 7-6 illustrates this

transformation.

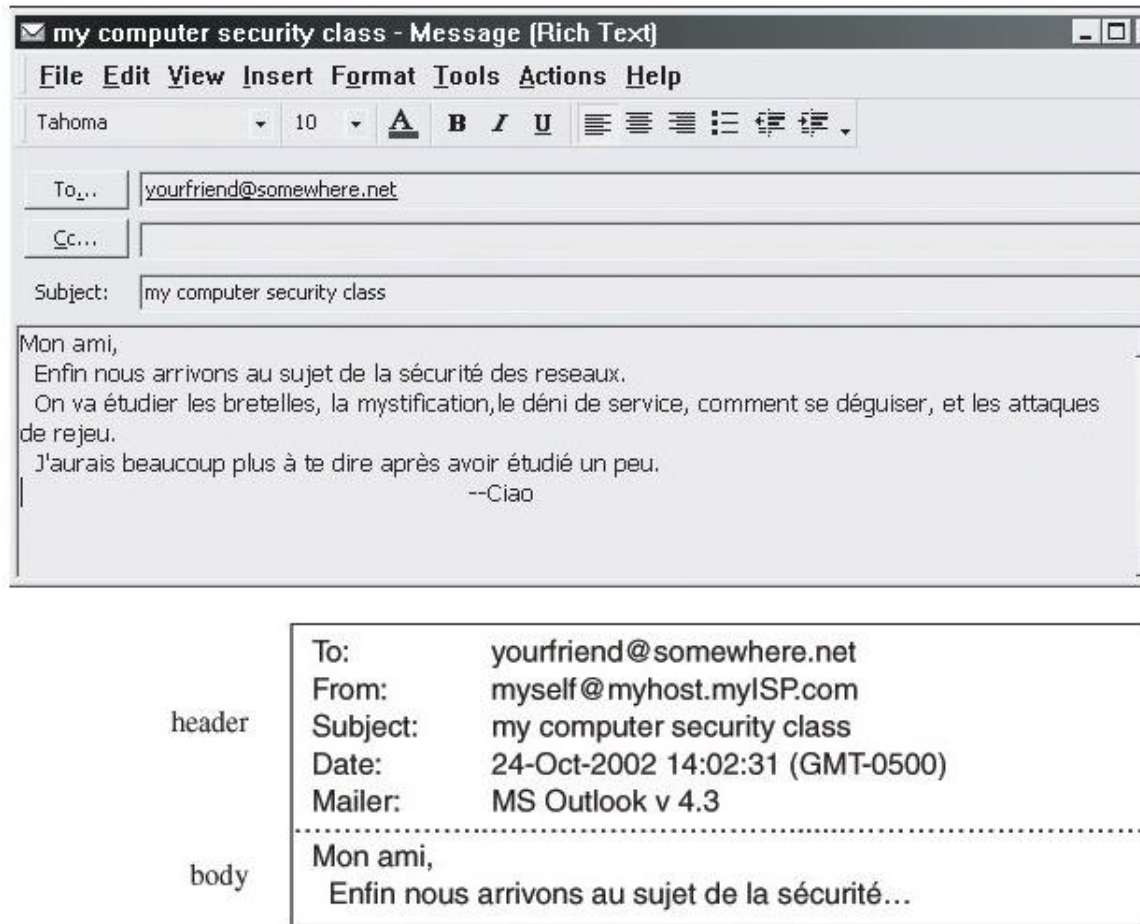


Figure 7.6. Transformation.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The message would not be sent, however, as raw text as you typed it. Raw text represents a somewhat inefficient code, since for an 8-bit byte, an alphabet utilizes comparatively few of the 255 possible characters. The presentation layer would presumably modify the raw text in another one. It will make compression, translation of characters and even some encryption. An e-mail message is a one-way exchange, so a session is not started in which data flow from one endpoint to another. Although there is little clear relevance to the concept of a communication session, the session layer is neglected for the time being. Even unintended signals intervene in a communication channel, as if a telephone line is interrupted or a radio or a television transmission is interfered by noise. To resolve this the transport layer includes error detection and correction coding to eliminate these incorrect signals.

TCP/IP

The OSI model is abstract; it demonstrates the various activities necessary to deliver a communication. The total implementation of a seven-layer transmission, however, would be too expensive for megabits per second; the OSI protocol slows down processes to unreasonable speeds. For this purpose, the protocol stack for most widespread network communications is the TCP/IP (transmission control protocol / Internet Protocol). The concept of TCP/IP is based on protocols and not on layers, but on four layers: application, end-to-end (host-to-host) transport, Internet, physical. Application software only works with conceptual data objects that are relevant to the applicant. Although TCP/IP is sometimes used as one acronym, there are actually two separate protocols: a connected communication session is being introduced by TCP above and beyond a more basic IP transport protocol. A third protocol is also an important feature of the suite, UDP (user datagram protocol). The transport layer accepts messages of varying lengths from the application layer, which are split up into convenient size in packets.

Table 7-2. Internet Communication Layers.	
Layer	Responsibilities
Application	user interactions User interaction, addressing
Transport	Sequencing, reliability (integrity), error correction
Internet	Flow control, routing
Physical	Data communication

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

A TCP protocol must guarantee that packets are appropriately sequenced and that the data is accurately transmitted in packets. The protocol will place packets out of sequence in the correct order, retransmit a missed packet and get a new copy of a corrupted packet. In order to trigger the invoking application, TCP sends the appropriate data stream. Sequence numbers are recorded and checked; integrity controls are verified, defective or missing packets have to be checked and waited for re-transmission time and overhead. Many applications expect perfect bits but a less precise data stream can be accepted in certain applications where speed or reliability is key. A TCP packet consists of a sequence number, an acknowledgment number to connect packets of a communication session, flags and source and destination port numbers. A port is a number that defines a certain programme that runs on a computer. If Jose and Walter start communication, for example, they generate a specific channel number by which their computers will route their respective packets to each. A port is called the channel number. Every service is linked through a well-known port, such as port 80 for HTTP, port 23 for Telnet, port 25 SMTP port or port 161 for SNMP connection.

(network management) More specifically, one of these services is awaiting for the given port number to be monitored and the operation on all data passed on to the port will be done.

7.2. Threats in Networks

What Makes a Network Vulnerable?

For a lot of threats, an isolated home user or a standalone workplace with a few staff is impossible. However, add a network and the risk increases significantly. Consider the distinction between a network and a standalone environment:

- **Anonymity** An intruder can attack thousands of miles and can never personally reach the system, its administrators or users. Therefore, underneath the electronic protection the possible intruder is secure. In order to conceal the origin of the attack, the attack could be transmitted through several other hosts. Computer-to-computer authentication is not the identical for machines as is for humans
- **Many factors of attack both targets and origins..** A basic computer system is an autonomous machine. Access controls on a computer maintain the details on the machine confidentiality. When a file is stored on a remote host, the data or the file itself will be transferred to the user via several hosts. An administrator of a host may impose strict safety policies, but the administrator has no power over other hosts on the network. The user must then rely on each of these systems' access control mechanisms. An attack can be performed from every host, because there are several vulnerabilities in a large network. -
- **Sharing** Since networks allow the sharing of resources and workloads, it is possible for more users than individual machines to access networked systems. Worse still, access to more networks is given so that network access controls for specific systems may be ineffective.
- **Complexity of System.** We see a complex piece of software as an operating system. Reliable security on a complex operating system is difficult, particularly one that is not explicitly configured for security. Two or more probably different operating systems are integrated in a network. Therefore, with a single computing system, a network operating system is likely more complex than an operating system. In the last two decades, more than many of the offices, the ordinary desktop computer today has higher processing capacity. By forcing the victim's machine to perform part of the attack computation, the attacker can use this power to benefit. And because of the capacity of an ordinary computer, most consumers don't know at any point what their machines actually do: Which processes operate in the background while you play Mars Invaders? This uncertainty decreases network security trust.

Unknown perimeter. The expansion of a network often means confusion about the network's limit. One host can be a node on two separate networks, such that one network's resources are still usable for users of the other network. While widespread usability is advantageous, it is a security drawback to this undisclosed or unregulated group of potentially malicious users. There is a similar issue of introducing additional hosts to the network. The potential existence of new, untrusted hosts should be faced by any networking node. The issues with network borders as seen in Figure 7-11. Note, for instance, that on the host in network D, users of networks A and B might not be

aware of possible connections. And in fact, A, B, C and E belong to the host in the middle of the A and B networks. If these networks have separate security rules, what are the rules in which the host is involved? Figure 7-11 points out the problems in defining the boundaries of a network. Notice, for example, that a user on a host in network D may be unaware of the potential connections from users of networks A and B. And the host in the middle of networks A and B in fact belongs to A, B, C, and E. If there are different security rules for these networks, to what rules is that host subject? Who Attacks Sites?

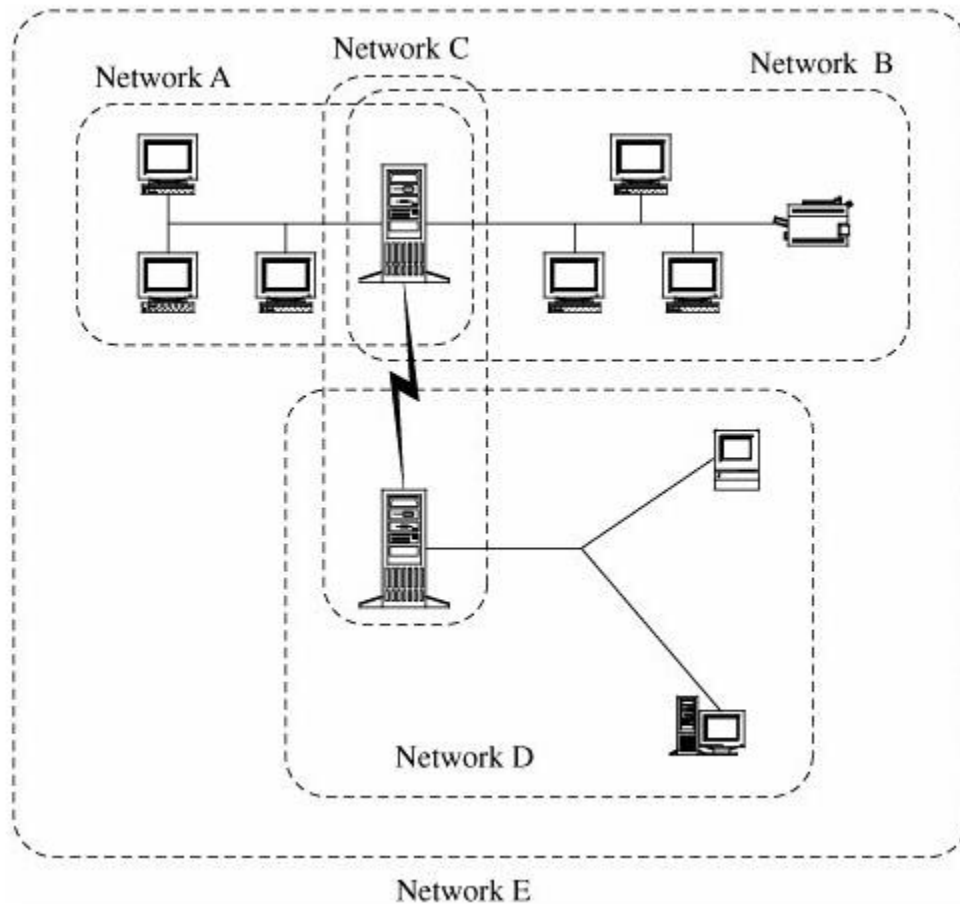


Figure 7-11. Unclear Network Boundaries.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Unknown path. Figure 7-12 shows that many host-to-host paths may exist. Suppose a user on host A1 wishes to deliver a message on host B3 to a user. This message could be redirected via hosts C or D before host B3. Host C can provide a security that is acceptable but not D. The routing of their messages is rarely done by network users.

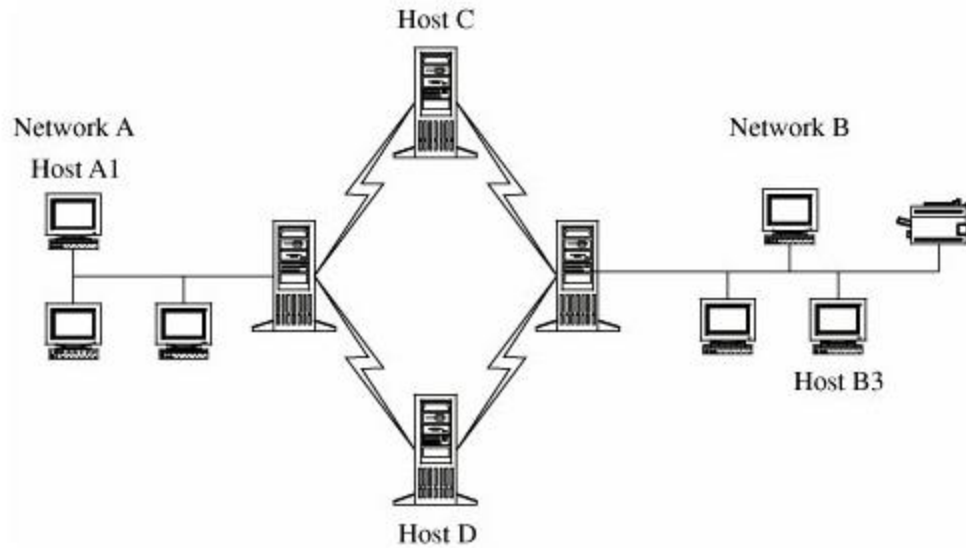


Figure 7-12. Uncertain Message Routing in a Network.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Who Attacks Networks?

Who are the attackers? We can't list the names, as we don't know who the criminals are in our town, our country or our world. Even if we know who they are, we don't know if we can stop them.

Challenge

Intellectual challenge is the only strong motivation for a network attacker. He or she is fascinated by the answers I can disrupt this network? What if I attempted this approach or technique?

Some attackers are motivated to undermine the allegedly undefeatable. In 1988, for example Robert Morris, who created the Internet worm, reportedly attacked as an attempt to see whether he might exploit a specific vulnerability. Additional attackers like the Cult of the Dead Cow are trying to expose security protection weaknesses such that people are paying heed to security strengthening. Some perpetrators are also nameless, anonymous people who work insistently to see how far unwanted they can go.

However, only few attackers will eventually discover unexplained defects. The overwhelming majority of attackers replay well-known and well-documented attacks just to see how they succeed on multiple hosts. If such a person actually [runs] to trigger an attack detected, planned and executed by someone else, intellectual stimulation is definitely not the driving force.

Fame

For certain attackers, the challenge of accomplishment is quite enough. Yet other attackers are finding recognition. Part of the challenge here is to perform the act; another part is to pay credit.

In certain situations we do not know who the attackers are, however they leave a name or moniker behind a "calling card": for instance, Mafiaboy, Kevin Mitnick, Fuzzy Rabbit, and members of the Chaos Computer Club. The actors are mostly anonymous, but they still reach fame by using pseudonyms. You may not be too open to praise, but you are pleased to see your attacks written down in the news media.

Funds and Espionage

Financial rewards also inspire attackers like in other environments. Some attackers conduct industrial espionage in pursuit of information on products, consumers or long-range plans of a company. We are aware of the role of industrial espionage as we read about laptops and confidential papers that have been stolen from hotel rooms when certain other important objects were left behind. Certain countries are renowned for their spying to support their state-owned enterprises.

Reconnaissance

We have already listed several reasons for attack and switch to the ways in which attackers carry out their attacks. Attackers don't normally sit at an attack terminal. Before acting, a skilled attacker investigates and prepares. Just like you can spend time learning about a gem shop before you enter to steal from it, a network intruder knows a lot about a potential target before the attack begins. The precursors for an attack are observed so as to allow one to block the attack until it is initiated if we know the signature behaviors.

Due to the Internet connection in most vulnerable networks, the attacker starts preparation by discovering the target to the extent practicable.

Port Scan

A port scan, a program for a specific IP address, is a simple way of gathering network information, which reveals ports responding to messages with several known vulnerabilities.

A port scan resembles a doctor's regular physical test, particularly the first questions for determining a medical history. Questions and answers may not be important in themselves, but they point to areas which indicate more study.

Port scanning shows an intruder 3 things: what standard ports or services are operating and performing on the target computer; which operating system installs on the target system, and which applications versions are available. For seeking information from an interconnected network system, this information can be easily accessed quietly, secretly, without identification or authentication, and little or no attention can be given to the scan.

Port scanning tools are widely accessible. Fyodor's nmap scanner is a helpful tool to download from www.insecure.org/nmap. In the case of an address, nmap will list all the available ports, the support server and the user ID owner of the service daemon. (The owner is relevant because it means that anyone who compromise the service will have the privileges.)

The netcat, published by Hobbit, on www.l0pht.com/users/l0pht is another publicly available scanner. Commercial products are a little more pricey but not prohibitive. Nessus (Nessus Corp), Network Associates and CyberCop Scanners, and Internet Scanners are well-known industrial scanners (Internet Security Systems).

Social Engineering

Social engineering is to drive someone to disclose or take action, usually by means of technology. The theory behind social engineering is to manipulate the inherent patterns and emotional responses of a possible victim.

The typical hacker will look for a software flaw to access a computer network. However, as an engineering support person, a social engineer might trick an employee to reveal their login credentials. The fraudster hopes to turn to a colleague to support him, maybe to act first and to consider later.

This kind of social engineering depends on a victim who takes a bait, unlike a fish who reacts to a worm on a hook. The person who hangs the bait needs to make the target act.

Example

A cybercriminal could leave a malicious software USB stick where the target is going to look. The criminal could also impressively mark the device — 'Confidential' or 'Bonuses.' A victim who takes a bait picks up and plugs it into a computer for details. The malware then injects itself into the computer automatically. And, maybe, first act and think, then think.

Intelligence

The attacker knows what's available from a port scan. The attacker learns some internal information from social engineering. However, a more comprehensive strategy would be good. The general term intelligence for information gathering. Security also refers to the collection or assembly of discreet bits of information from multiple sources.

One common intelligence strategy is referred to as "dumpster diving." where products have been dumped in waste bin or recycled boxes. It's unbelievable what we're throwing away without thought. Network diagrams, security settings written on devices, system inventions and source code, telephone and employee lists and more could be combined with the remainder of the lunch. It may also be helpful for obsolete printing. Rarely will the security system interface shift altogether. Very commonly, only one law is introduced, removed or updated, so an intruder is extremely likely to be attacked effectively on the basis of old information.

Intelligence gathering can require eavesdropping as well. Trained spies will accompany staff to lunch and listen to co-workers from nearby tables as they chat about security problems. Or spies may be used by key staff for the transmission of valuable information, to coopt, coerce or trick.

Operating System and Application Fingerprinting

The port scan contains very precise details for the intruder. For example a port scan may be used by an attacker to evaluate that port 80 is available and supports HTTP, the web pages transfer

protocol. However, the intruder will have some associated questions, such as which commercial server application operates, which version and what the operating system and version are. Armed with this particular information, the attacker may consult a list of known vulnerabilities in individual applications in order to decide which flaws are used.

How will the attacker answer such questions? Standard and vendor independent network protocols. However, the code of each vendor is individually implemented, and there could be minor interpretations and behavioral differences. The improvements do not make the program incompliant, but they are sufficiently distinct to differentiate each version. For e.g., different sequence numbers, TCP flags, and new options can occur in each version. To see that it should be taken into account that sender and receiver should manage the connection to a TCP session with a sequence number. Certain implementation solutions react with a particular sequence number, others react with a larger number and others react with an arbitrary number. Similarly, in one variant, such flags are unknown or incompatible. The framework and version will also show how a system responds to the request (by acknowledging or retransmission it, for example). Finally, modern features give a good indication: the new release adds a new feature but an existing version refuses the request. These features may be defined by the manufacturer and the update, often called the operating system or device fingerprint.

For instance, a scanner like nmap responds on the target operating system in addition to performing its port scan. See www.insecure.org/nmap/nmap-fingerprinting-article.html for more detail on how this is achieved.

The application may sometimes be recognised. Usually a client-server connection will be treated in full according to the Protocol Rules inside the application: "Please send me this page; OK but run this support code; thanks, I just did." For eg, a Telnet program might be used by the attacker to transmit irrelevant messages to another application. Ports such as for example 80 (HTTP), 25 (SMTP), 110 (POP), and 21 (FTP) may react with something similar to

Server: Netscape-Commerce/1.12

Your browser directed a non-HTTP compliant subject matter.

or

Microsoft ESMTP Email Service, Variant: 5.0.2195.3779

This reply shows the attacker which use and version happen to be running.

Bulletin Boards and Chats

The Internet is perhaps the strongest tool since the invention of the media to spread information. Perhaps the most dangerous method for knowledge sharing.

The sharing of information is facilitated by numerous underground bulletin boards and chat rooms. Attackers will publish new vulnerabilities and techniques, read what others have done, and look for more information about systems, software or websites. Note that anyone can publish something

like anything on the Internet, and so there is no guarantee that the content is credible or precise. And you never know who's on the Internet blogging.

Availability of Documentation

Even the vendors themselves distribute information valuable to an attacker. For example, in order to build compatible software Microsoft will develop application vendors can examine a Microsoft product. This toolkit also provides attackers tools that can then be the target of an attack while analyzing a product.

Threats in Transit: Eavesdropping and Wiretapping

Paying attention in is the best place to attack. The content of a communication passing by may be chosen by an attacker. The word "eavesdrop" means "overhearing" with no additional effort. For instance, we might claim that an intruder (or system administrator) eavesdrops all traffic passing through a node to monitor. The administrator may have a legitimate aim, such as seeking for an unsuitable resource usage (for example to access non-work-related websites from an enterprise network) or communication with the wrong entities (for instance, passing files to an enemy from a military computer)

Wiretap is a more hostile term, meaning communication is intercepted by some action. Passive wiretapping just sounds like "listening," much like eavesdropping. However, active wiretapping involves something being inserted into communication. Marvin, for instance, could substitute Manny's communications for his own or establish allegedly Manny-based communication. In its original source, the word wiretapping is a physical act that usually conveys a device which extracts information while flowing over a wire. But no real interaction is currently needed. A wiretape can be made secret so that no one knows that the material is intercepted either by the sender or the recipient of a message.

Wiretapping works in different ways with regards to the communication medium utilized. Let us seem more diligently at each probable choice.

Cable

Both the signals in an Ethernet or other LAN may be intercepted on the cable at the local level. Every LAN connector (e.g. computer board) has a unique address and each board as well as its drivers is designed to mark all packets with their unique address ("return address") from their host and only to take those packets that are addressed to their host from the internet.

However, the exclusion of only the packets addressed by a single host. A program could hardly avoid analyzing each packet as it occurs. Both packets can be retrieved on the LAN by the device called a packet sniffer. Alternatively, the so-called unique address of another current card can be reprogrammed for one of the device cards in the LAN so that all packets can be fetched for one single address. LANs are typically only used in settings which are reasonably friendly, so that these types of attacks do not happen often. (The rogue card is necessary, to prevent detection, to place on the net copies of packets it intercepted.)

Clever attackers can use the properties of a wire to read packets without being physically controlled. Standard wire (and many other elements of electronics) emit radiation. An attacker will tap a wire and decode the radiated signals without having physical contact with the cable by means of a technique called induction. A cable signals only pass short distances, and other conduction materials may interrupt them. Inductance threats are a deep concern for cable networks, and are cheap and quick to procure the hardware needed to capture these signals.

If the attacker is not sufficiently close to use an inductance, more drastic steps may be needed. Direct cutting is the fastest way to intercept a cable. When a cable is disconnected, it ceases all service. An intruder will split into a secondary cable, which then receives copies of all signals through the main cable, as part of the reparation process. It is a little less clear but there are ways to reach the same goal. For e.g., some of the external drivers could be carefully revealed by the attacker, connected to it, then exposed and connected to those internal drivers carefully. These two operations adjust the resistance of the cable, known as the impedance.

Network signals are multiplexed which ensures that more than one signal at a certain time is transmitted. For e.g., it is possible to merge two analog (sound) signals, such as two tones in a music chord, with two digital signals interleaving, such as play-cards being mixed. A LAN holds various packets, but a WAN can be highly multiplexed by leaving its host. So a wiretapper on a WAN must be able to capture and retrieve not just the requested communication from others, for which it is compounded. Although this can be achieved, it is sparingly used by the initiative involved.

Microwave

Microwave signals are not transmitted along a wire; they are transmitted through the air to make them usable to external users. Usually, a signal of the transmitter is centered on its recipient. As in Figure 7-13, the signal path is very wide, so you can reach the recipient. The broad swath is an invitation to misfortune from a security point of view. Not only can an individual interfere with the view line between sender and receiver to intercept a microwaving signal, but even a person can pick up a whole transmission on an antenna near but just beyond the direct focus point.

In general an interception-prevention signal is not safe or isolated from a microwave signal. Therefore, the microwave is a very unpredictable medium. It is doubtful, though, that anyone will isolate an individual transmission of all the other connected ones, because of the vast amount of traffic conducted by microwave connections. There is not such a well-protected amount for a

private microwave connection that carries only contact for one business

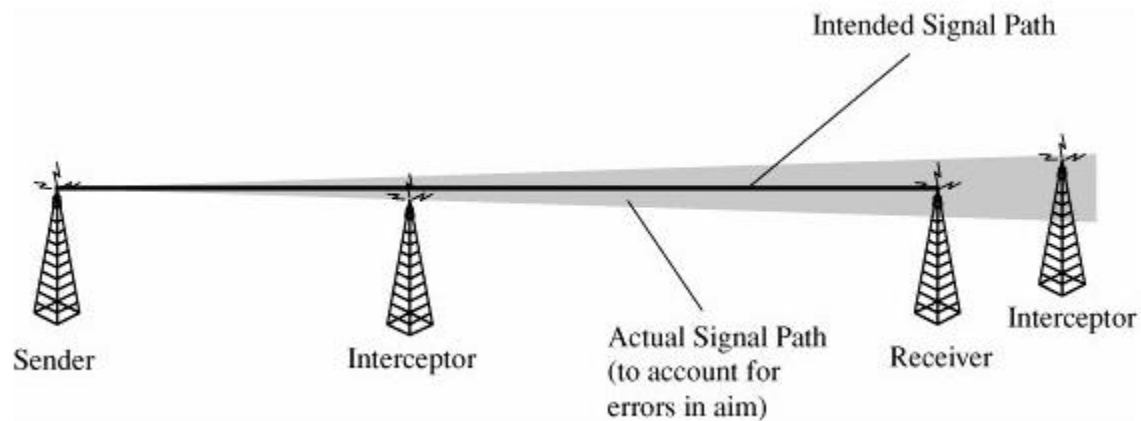


Figure 7.13. Path of Microwave Signals.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Satellite Communication

Satellite communication has a related issue of being spread across an area larger than that of the expected point of receipt. Various satellites have different features, but certain signals can be intercepted in an area many hundred miles wide and a thousand miles long. Consequently, the intercept capacity for microwave signals is much higher. However, since the amount of satellite communication is usually high, the probability of detection is limited.

Optical Fiber

Optical fiber provides two main security benefits relative to other communication medium. Firstly, any time a new link is created the whole optical network must be carefully calibrated. Therefore, without detection, no one can tap an optical system. The only way to break the network stability is to break one fiber in a bundle.

Second, fiber optic carries light, not electricity. As light does not emanate from a magnetic field as electricity does. An inducing tap on an optical fiber cable is also not necessary.

However, the mere use of fiber does not guarantee security, nor does encryption.

Wireless

Wireless networks are electronic networks not connected to any kind of cable. Using a wireless network allows businesses to ensure that wires are not reached in buildings or connected to multiple equipment locations at cost. Wireless networks are based on radio waves, which are applied physically at network structure level.

The challenges with wireless communication occur in the capacity of interception by intruders.

But it is not interference, the main threat; it is interception. For around 100 to 200 feet, a wireless signal is strong. To understand these numbers, a 10-story ordinary office building is seen, with ten offices "wide" in five deep offices, equivalent to several buildings found in office parks or on campuses. Assume that in the top floor corner you have a cellular base station (receiver). Signals from opposite corners of the ground floor could be obtained from that station. The signal could also be received in this building if a similar building is adjacent.

It's easy to get a strong signal. As well as a wireless signal can be picked up within a few miles with a low-cost tuning antenna. In other words, anyone who would like to steal your signal from many streets could do so. The interceptor could hold your communications in a truck or van for a while without arousing suspicion.

Interception

Wireless interception is often a threat by passive or active wiretapping. You should respond to this threat by believing that it is addressed by encryption. Unfortunately, encryption isn't necessarily used for wireless communication and encryption is not as effective as the protection of a dedicated attacker.

Theft of Service

The Wireless device often acknowledges a secondary problem: a rogue network connection. Several hosts run the DHCP, which helps a single client to exchange an IP address and a connection with a host. In office or on campus settings, where all users (clients) are not always involved, this protocol is useful. Users can share a limited number of IP addresses. The addresses can essentially be found in a pool. A new client demands a connection and an IP via DHCP, which is allocated by the server from the pool.

This scheme recognizes a major authentication challenge. If the host does not authenticate users prior to a connection, an IP address and network access are given to each requested client. The situation is so bad that a map is visible in some urban areas, which shows a vast number of networks accepting wireless connections. The task typically happens before the user on the client workstation finally recognizes and authenticates to one server, so there might be no authenticated identities which the DHCP server may request.

Overview of Wiretapping

There are various points where an interceptor can access network traffic. Figure 7-14 demonstrates how communication to its target is revealed from its origin.

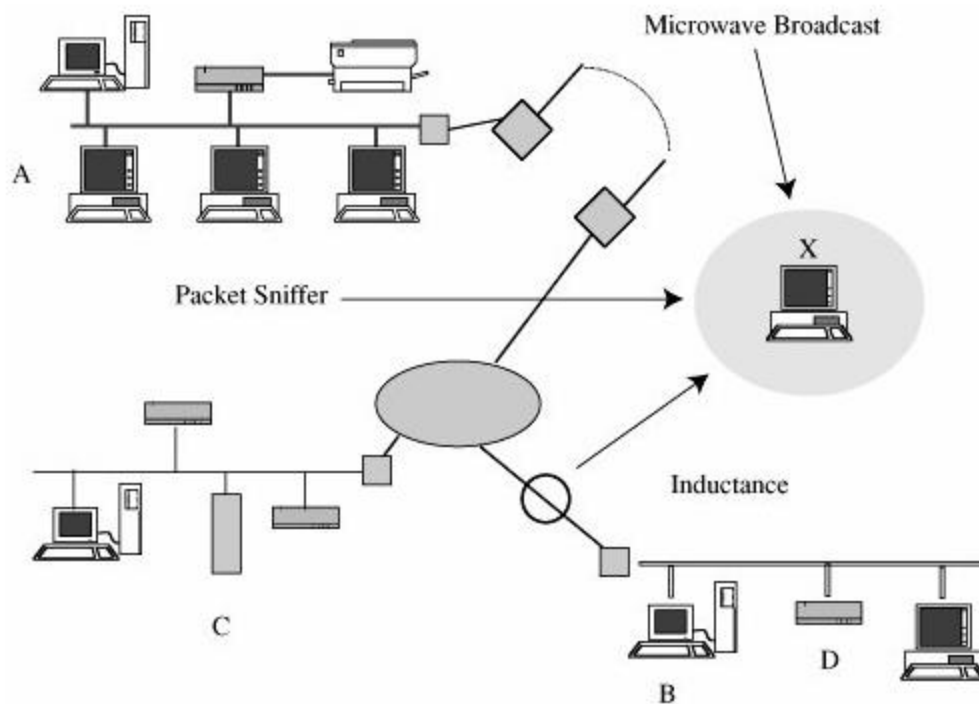


Figure 7.14. Wiretap Vulnerabilities.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

It can presume, through a security point of view, that all connections between network nodes can be disrupted. For this purpose, corporate network users use encryption to secure their communications confidentiality. Local network communications can be secured, even though local connections with a strong physical and administrative security may be more protected for performance reasons.

Protocol Flaws

The entire Internet group publishes Internet protocols for public review. Each authorized protocol is recognized by its Request for Comment (RFC) number. Many protocol problems were identified and resolved by sharp reviewers before the Protocol was standardized.

But the meanings of protocol are defined and examined by humans. Protocols are often carried out by those who are imperfect. For e.g., sequence numbers are used to establish TCP connections. The client (initiator) sends a connection opening sequence number, the server responds to that number and its own sequence number, and the client responds by the sequence number of the server.

Impersonation

In certain situations, it's better to get information on a network than through wiretapping: Do not personalize another user or process. Why risk taping a line or why interrupt one communication from many, if you can directly obtain the same data?

In a larger network than in a local one, impersonation is a greater threat. Local people also have easier ways than other users of access; they may only sit down at an unattended workstation, for example. Also in local area networks, however, impersonation attacks cannot be overlooked, since local area networks may often be linked to larger area networks without anybody worrying about security.

Within an impersonation, an attacker offers many choices:

- Guess the identification and authentication information on the target.
- Pick the identification and authentication details of the target from a previous or communication from wiretapping.
- Bypass or disable the target computer authentication mechanism.
- Make use of a target that won't be authenticated.
- Target a goal whose authentication information are known.
- Let us take a look at each choice.

Authentication Foiled by Guessing

Multiple experiments indicate that many individuals pick passwords that are easy to guess. We've seen the 1988 Internet worm capitalizing on this vulnerability precisely. In order to try a few changes to the username, a list of around 250 popular passwords and, finally, the term in a dictionary, Morris's worm attempted to impersonate each individual on a target computer. Unfortunately, these simple attacks do open several user accounts.

Default passwords are a second source of password guessing. Many systems are installed initially as login IDs with default accounts with the following names: "guest" or "null" or "password" accompanying these IDs. The administrator may setup the system with this type of IDs. Administrators often ignore or at least forget to delete or deactivate these accounts.

A password can merely be the indication that the user doesn't want other users using a workstation or account in a secure environment, such as the Office LAN. The password-protected workstation often holds confidential data such as employee salaries or new product information. Users might assume that their password is appropriate for preventing a curious colleague from being targeted. But if this reliable environment is connected with an insecure larger network, it will be easier for those users using passwords become easy targets.

Authentication Thwarted by Eavesdropping or Wiretapping

As distributed and client-server computing increases, certain users have privileges to connectivity on various connected systems. Authentication between hosts is necessary to defend against unauthorized outsiders using such accesses. This access may be made directly to the user or by means of a host-to-host authentication protocol on behalf of the user. In any event, the subject's account and authentication information are sent to the host of destination. If this information is

passed on to the network, it is revealed to anyone who observes the network communication. An impersonator will replay this same authentication data before it is updated.

Due to the major weakness of transmitting the password in clear, protocols were developed that would prohibit the password from being passed by the user's workstation. But the details are important, as we have seen in many other locations.

Authentication Foiled by Avoidance

Authentication is precisely only successful if it works. A weak or unreliable authentication permits the entry of any system or entity who can override the authentication.

In a standard operating system malfunction, the buffer was of a set size for type characters, measuring all the typed characters, including correction. If a user has inserted more characters than the buffer contains, the overflow caused the operating system that allowed a password comparison to be overridden and acts as though a proper authentication has been provided. Anyone seeking access will take advantages.

Many network hosting systems, in general those connected to wide networks, operate Unix System V or BSD Unix variants. Many users do not know which networked operating system is in use in a local environment; even less will be aware of, able of, or involved in exploiting flaws. Any hackers however search wide ranging networks scan on a daily basis for hosts with weak or defective operating systems. Connection to a wide network, particularly the Internet, thus reveals these flaws to an enormous audience intent on exploiting them.

Nonexistent Authentication

Whether the same user uses two computer systems to store data and run the processes and if each user authenticates first-time access, it can be concluded that authentication of computer-to-computer or local user-to-remote method is unnecessary. This two machines and their users are a secure environment where repeated authentication seems to have an unnecessary complexity.

But that is not a true statement. Consider the operating system of UNIX and see why. In UNIX the file `.rhosts` lists trustworthy hosts and `.rlogin` lists trusted users that have access without authentication. The files are designed to allow computer-to-computer communications from users already authenticated in their primary hosts. Those "trusted hosts" may also be used by external persons who access one system via an authentication flaw (e.g. the guessed password) and then move to another system which acknowledges the authenticity of the user from a trusting system.

An attacker might also know that a system does not need an authentication of certain identities. Some systems provide "guest" or "anonymous" accounts to allow externals to access things they want to share. For example, a bank may publish an existing list of foreign currency rates, or a library with an online database may provide the catalogue for anyone for searching or any of its reports can be viewed by a company. A user can access publicly-available products as a "guest". Normally, there is no need for password, or when the user asks a password, the user is seen to type GUEST (or the name, indicating a string like a name). These accounts allow unauthenticated users to access any one of them.

Well-Known Authentication

It should be unique and impossible to guess data for authentication. Unfortunately, though, it is sometimes the convenience of a widely known method of authentication that uses protection. For instance, the same password was designed for one computer manufacturer to grant its remote maintenance personnel access to one of their customer's computers worldwide. Fortunately, before this idea was implemented, security analysts figured out the possible threat.

The SNMP is a commonly used system network management protocol to remotely control network devices like routers and switches that do not support normal users. SNMP uses a 'community string,' which is mainly a password for the device's community that interacts with each other's. However, network devices are particularly built for quick and minimal set-up, and often network administrators do not modify the default community string mounted on a router or switch. This looseness opens up these devices for multiple SNMP attacks in the network perimeter.

Any vendors also dispatch computers with a default password installed with a server management account. Or even the systems have such a demonstration or test account as well as no password is required. Some administrators can not alter the passwords or delete these types of accounts.

Trusted Authentication

Authentication, eventually, would become a concern if identity is passed to other sources of trust for e.g., a file can show who a specific host can be trusted or a device will "confirm for" for a user through the authentication process. We previously noted how the Unix .rhosts, rlogin and /etc/hosts/equiv files show hosts or trusted users on other hosts. These functions are important, but must be used very cautiously, for users who have accounts on many computers or for the administration, management and service of a network. Any of them is a possible hole from which a remote user can reach a remote attacker.

Spoofing

Spoofing can be extended to several communication techniques, and varying degrees of technological competence can be utilized. Phishing attacks which are scams to get confidential information from individuals or organizations may be used for spoofing. The numerous examples of spoofing techniques are masquerading, session hijacking, and man-in-the-middle attacks.

Masquerade

Attack masquerade involves creating fake identities and impersonating legitimate sources. Attackers use masquerade attacks to identify themselves as someone else and access victims' computers or networks by unauthorized. The masquerade attack includes the impersonating of legitimate sources and the creation of fake identities to lead victims to send their personal and financial details. Attackers often use masquerade threats to claim that they are someone else to obtain unauthorized access to victims' networks or systems.

During Masquerade attacks, attackers send phishing e-mails to targets that are genuine and ask victims to request their personal details. Attackers often steal the login credentials of users and acquire unauthorized access by masquerading the user to confidential files from a computer.

Examples of Masquerade attacks

Example 1 - Tax-themed phishing campaign

Researchers witnessed tax phishing campaigns impersonated legitimate tax agencies, such as the US Internal Revenue Service, the Canada Revenue Agency, and the Department of Home Affairs in New Zealand.

Such phishing emails included HTML attachments or URLs that opened clicking on online form or sent victims to a broken spoofed IRS login page

The online form includes financial information of victims, while the login page gathers login credentials for victims.

Victims will then be redirected to official websites of the tax authority.

Session Hijacking

Session hijacking is an attack where an attacker takes over a user session. When you log in to a service, such as your banking application, a session will commence and end when you log out. The attack depends on the intruder knowledge of your session cookie, so cookie hijacking or cookie side-jacking is often called. Though every computer session can be hijacked, it is commonly browser sessions and Web apps which can be hijacked.

In most situations when you login to a web application, you will have the server set to log in and authenticate a temporary session cookie within your browser. For your server the most common way for you to define your browser or current session is HTTP, a stateless protocol, and session cookies added to all HTTP headers. To perform session hijacking, an attacker needs to know the victim's session ID (session key). This can be obtained by stealing the session cookie or persuading the user to click a malicious link containing a prepared session ID.

Man-in-the-Middle Attack

In cryptography and computer protection, a person-in-the-middle attack (MITM) is an attack wherein the attacker secretly relays and likely alters the communications among events who accept as true with they may be directly speaking with every other. One example of a MITM assault is lively eavesdropping, wherein the attacker makes independent connections with the sufferers and relays messages between them to make them consider they may be speak immediately to each other over a private connection, while in truth the complete communications is managed with the aid of the attacker. The attacker need to be capable of intercept all applicable messages passing between the 2 parties and inject new ones. This is straightforward in many instances; as an example, an attacker within reception range of an unencrypted wireless access point (Wi-Fi) may want to insert themselves as a man-in-the-middle.

As it targets to avoid mutual authentication, a MITM attack can be triumphant best whilst the attacker impersonates each endpoint sufficiently nicely to fulfill their expectations. Most cryptographic protocols include some form of endpoint authentication in particular to save you

MITM attack. For instance, TLS can authenticate one or each events the use of at the same time depended on certificate authority.

Suppose Alice desires to talk with Bob. Meanwhile, Mallory desires to intercept the communicate to eavesdrop and optionally to supply a fake message to Bob.

First, Alice asks Bob for his public key. If Bob sends his public key to Alice, however Mallory is able to intercept it, an MITM attack can begin. Mallory sends Alice a solid message that looks to originate from Bob, however rather includes Mallory's public key.

Alice, believing this public key to be Bob's, encrypts her message with Mallory's key and sends the enciphered message back to Bob. Mallory again intercepts, deciphers the message the use of her private key, probably alters it if she desires, and re-enciphers it the use of the general public key she intercepted from Bob whilst he in the beginning attempted to send it to Alice. When Bob gets the newly enciphered message, he believes it got here from Alice.

1. Alice sends a message to Bob, that's intercepted by way of Mallory:

Alice "Hi Bob, it is Alice. Give me your key." → Mallory Bob

2. Mallory relays this message to Bob; Bob cannot inform it isn't always definitely from Alice:

Alice Mallory "Hi Bob, its Alice. Give me your key." → Bob

3. Bob responds along with his encryption key:

Alice Mallory ← [Bob's key] Bob

4. Mallory replaces Bob's key together with her very own, and relays this to Alice, claiming that it's far Bob's key:

Alice ← [Mallory's key] Mallory Bob

5. Alice encrypts a message with what she believes to be Bob's key, wondering that best Bob can examine it:

Alice "Meet me at the bus forestall!" [Encrypted with Mallory's key] → Mallory Bob

6. However, as it become surely encrypted with Mallory's key, Mallory can decrypt it, examine it, adjust it (if favored), re-encrypt with Bob's key, and ahead it to Bob:

Alice Mallory "Meet me on the van down by way of the river!" [Encrypted with Bob's key] → Bob

7. Bob thinks that this message is a at ease conversation from Alice.

This instance suggests the need for Alice and Bob to have a few way to ensure that they're really every using each other's public keys, rather than the public key of an attacker. Otherwise, such assaults are usually possible, in principle, in opposition to any message sent the use of public-key era. A variety of strategies can assist shield in opposition to MITM attacks.

Message Confidentiality Threats

The public nature of networks can be easy for an attacker to violate the confidentiality of message (and maybe integrity). Eavesdropping and impersonation attacks can result to confidentiality or a failure of integrity. Several other vulnerabilities that may affect confidentiality are considered here.

Misdelivery

Messages are sometimes misdelivered because of a network hardware or software defect. Messages, which is an integrity or an availability problem, are most often lost completely. However, sometimes a destination address or some malfunctions defects are changed to cause a message to be sent to someone other than the intended recipient. All these "random" occurrences are very unusual.

Extra common than network flaws are human mistakes. It's far a long way too easy to mistype an cope with inclusive of 100064,30652 as 10064,30652 or 100065,30642, or to type "idw" or "iw" in preference to "diw" for David Ian Walker, who is referred to as Ian via his friends. There may be truly no justification for a computer network administrator to discover people with the aid of meaningless lengthy numbers or cryptic initials when "iwalker" would be a ways much less prone to human error.

Website Vulnerabilities

The idea that a website is almost entirely open to the user is extremely vulnerable. You don't normally get to see the coding of the program if you use an application program. The attacker will import the code of the site for offline studies over time using a website. You may not have the ability to control in any way what aspect of the software you use, but a web attacker can control which sites to enter, maybe even page 5 without first running pages 1-4. It is also possible that the attacker selects which data to include and plays with various data values to see the response of the site. In brief, the intruder has some benefits and can be uncontrollable.

Web Site Defacement

The website defacement attack is one of the most known threats. The great number of defaced pages and the coverage of the outcome also indicate the attacks in the popular press.

A defacement is not only common due to its visibility but also because it is easy to do so. Websites are designed to download their code so that an attacker can get the complete hypertext document and any programs that are directed to the client during the loading process. An attacker may also access feedback left by programmers as they created or kept the code. Essentially, the installation process provides the attacker the website blueprints.

Buffer Overflows

Buffers are memory storage regions which hold data temporarily while transferring it from one location to another. When the amount of data exceeds the memory buffer storage space, a buffer

overflow (or buffer overrun) takes place. This overwrites the adjacent memory locations while the program is trying to write the data into the buffer.

For eg, the login credential buffer may anticipate user name and password input of 8 bytes, which means that the program may enter the excess data after the buffer limit if a transaction needs an input of 10 bytes (that is, 2 bytes more than expected).

Buffer overflows can impact all software types. They are usually attributed to malformed inputs or failures to assign sufficient buffer space. If the transaction overwrites executable code, it may lead to unpredictable compliance with the program and cause incorrect results, memory access errors or crashes.

Server-Side Include

A server-side inclusion is considered a potentially more severe concern. The problem is that web pages can be organized to automatically trigger a specific function. In the "contact us" part of the displayed page, several pages use web commands for email sending. Commands are placed in the field interpreted in HTML, such as email. One of the vital Server-Side Include commands is exec, to execute an arbitrary file on the server. For illustration, the Server-Side Include command

Exec is one of the server-side commands for executing an undefined server file. For example the server-side include command

```
<!--#exec cmd="/usr/bin/telnet &"-->
```

Needs to open a Telnet server session running in the server name (i.e. the server privilege). It may be interesting for an attacker to execute commands such as chmod (change object access rights), sh (create a shell command), or cat (copy to a file).

7.3. Network Security Controls

A long list of security attacks and reports of serious security attacks are frequently reported by news media. You may be prepared to assume from this that security in the network is desperate. Several strategies have already been presented in previous chapters to address security problems, such as confidentiality and integrity encryption, access control reference monitors and deep defense overlapping. These strategies are also useful for network protection.

Security Threat Analysis

In other situations, remember the three steps of a security threat analysis. First, we examine all components of a system in order to find out how each component interacts and what each part is doing. Next, confidentiality, integrity and accessibility may damage. Finally, we assume the types of attacks that might cause such damage. With a network we can do the same things. We start by examining the different components of a network:

- Local nodes attached via
- Local communications back links to a

- Local area system, which also offers
- Local data safe-keeping,
- Local procedures, and
- Local devices.

The local system is also linked to a

- Network gateway gives access via
- Network communications links to
- Network control information,
- System routers, and
- Network resources, such as for example databases.

Design and style and Implementation

Architecture

Definitely planning can be the most controlled, as with so many of our areas studied. In particular, we can consider its global architecture and plans to build security as one of the key constructions if we build or modify computerized systems. The architecture or design of a network can also have a substantial impact on its safety.

Segmentation

As segmentation is a strong operating systems security regulation, this will minimize the likelihood of harm in a network in two major ways: Segmentation eliminates risks and reduces the risk to a single vulnerability.

Assume that your network carries on electronic business for internet users. Your network can be of fundamental importance

- Web server for handling HTTP sessions of users
- application code, to submit your purchase goods and services
- a goods database and maybe an accompanying inventory to inventory and supplier requests
- a database of orders taken

If all this task would be done on a single machine, your network would find it difficult: any compromise or failure of the same machine would destroy all of your trade capacity.

Several segments with a more stable nature as shown in Figure 7-19. Suppose that a piece of hardware is a web server box that is open to public access. This box does not include other more sensitive features, for example, user naming or access to an insecure data archive, to reduce the possibility of attacks from outside of the system. Separate segments and servers which are in accordance with the less privileged and encapsulated concepts mitigate the possible damage should

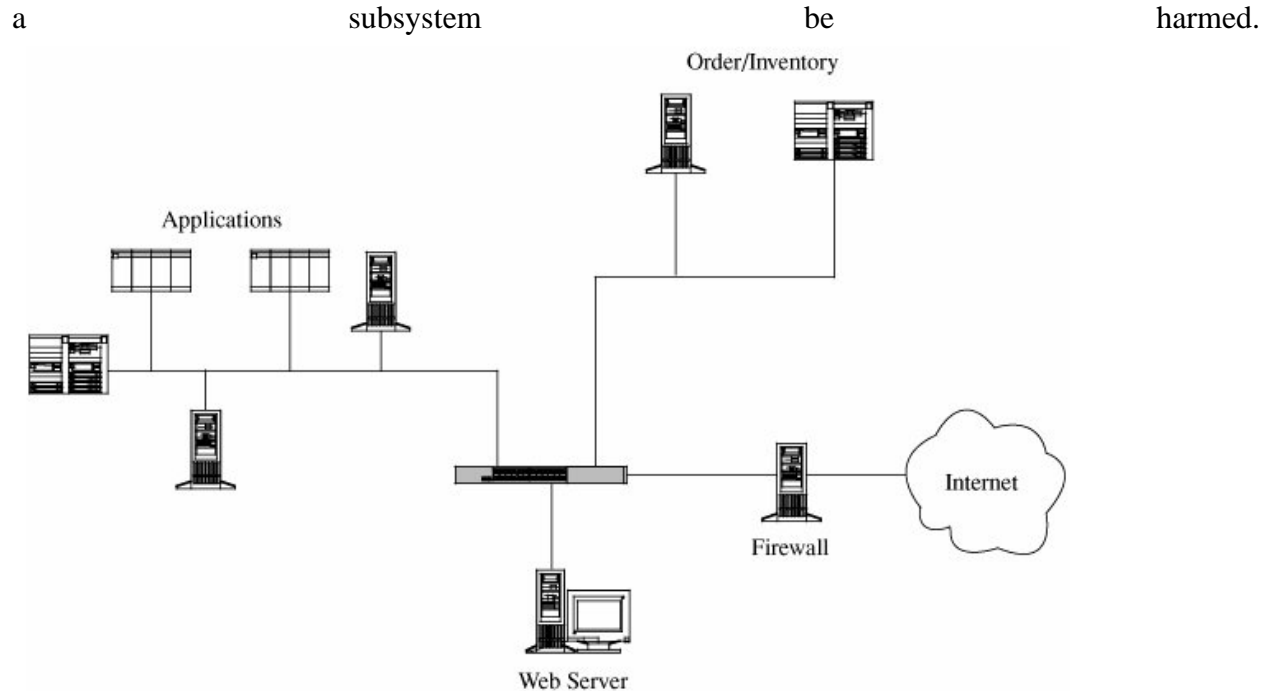


Figure 7-19. Segmented Architecture.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Different access is another path to network segmentation. For example, assume a network is used for three purposes: to test the next production version and build subsequent networks with the "live" production system. If the network is well segmented, outside users can only access the live system, test testers should only access the test system, and developers should only have access to the production system. Segmentation enables all three populations to coexist without risking a shift in the development mechanism by a developer, for example.

Redundancy

An additional key architectural control is the ability to do redundancy: allow a feature to be executed on more than one node so as to prevent "putting all the eggs in one basket." Two servers, which is called failover mode, will have a better configuration. In failover mode, servers regularly communicate with each other to decide if the other is still active. If one loses, the other can do all processing. Although when a malfunction happens, output is decreased by about half, at least some processing is carried out.

Single Points of Failure

The architecture must ideally prevent the network from failing. Indeed, the design should ensure that the system acceptably tolerates a failure (such as slowing down but not stopping processing, or recovering and restarting incomplete transactions). One approach to test the failure tolerance of

the network architecture is by looking for single failure points. In other words, we could ask if there is a single point on the network which could reject any or a substantial portion of the network if the network were to fail. For instance, a single database is vulnerable to all the failures in one location. Good architecture of the network removes individual failure points. Distributing copies of the database to separate networking segments will possibly mitigate at some stage the risk of significant loss by failure and in different physical locations. The execution of such a design frequently requires tremendous overpower, for example synchronizing the independent databases. However, we can typically deal with fault tolerant features more quickly than with the damage caused by a single broken connection.

Encryption

The most critical and flexible tool for a network security specialist probably is encryption. In previous chapters, we saw that encryption offers data privacy, authenticity, integrity and limited access. Since networks also entail much greater threats, encryption often secures records, likely in conjunction with other controls.

Let us remember these points before we start studying the use of encryption to counter network security threats. Be aware that first encryption is not a magic bullet or panacea. A broken system design is indeed a defective system design with encryption. Second, notice that encryption only covers encrypted material. Before transmitting data, it is exposed between a user's hands and the encryption process, and exposed after remote decryption. The best encoding cannot guard against a malicious Trojan horse who intercepts data before the encryption stage. In the end, cryptography is not better than the main administration. If a weak encryption key may be guessed or assumed by an attacker, the game is over. Often people who struggle to understand encryption misinterpret fairy dust with a magic protection system to sprinkle. If there was such a fairy dust this book would not be needed.

Encryption may be achieved between two hosts (called link encryption) or between two applications in network applications (called end-to-end encryption). We look at each of them below. Key distribution is still an issue for all forms of encryption. The sender and the recipient can receive encryption keys securely. We also examine techniques for secure network key distribution. Eventually, we study a network computing environment cryptography facility.

Link Encryption

In link encryption, data is encrypted immediately prior to being placed on the physical communication link by the system. In this case, the layer 1 or 2 of the OSI model is encrypted. The same happens as communications arrive at a receiver's device and reaches the receiving

computer, decryption is also happening. Figure 7-20 demonstrates a model for link encryption.

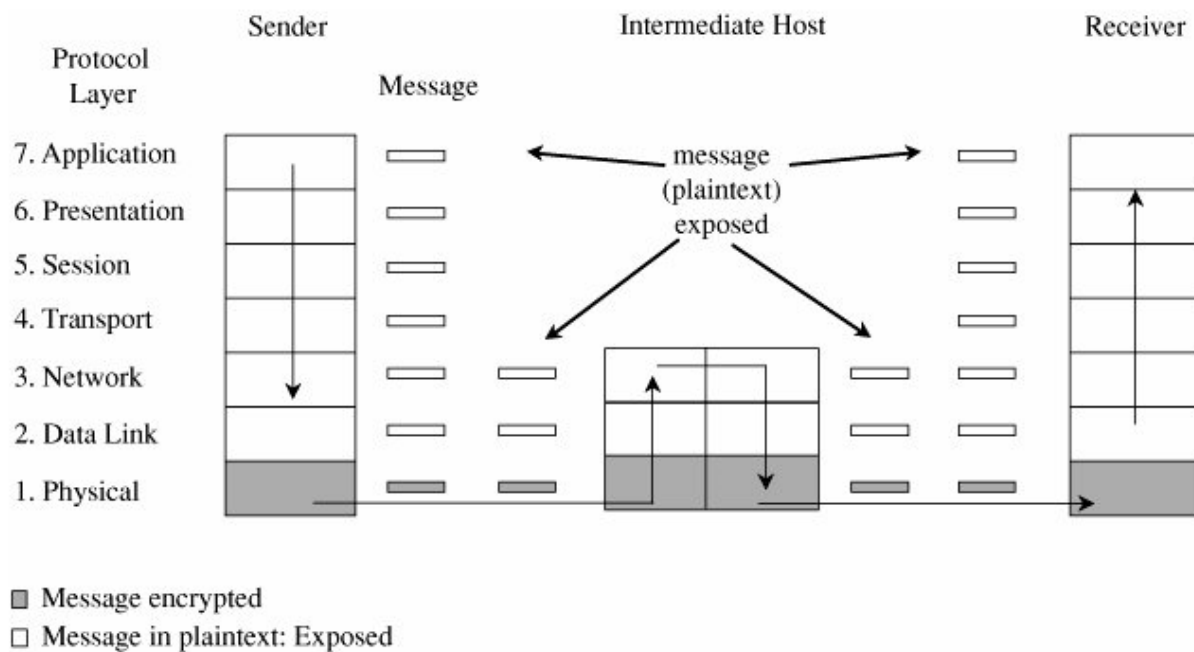


Figure 7-20. Link Encryption.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Encryption preserves the transit message from two computers, but it is stored inside the hosts in plain text. Note that because the encryption is placed on the lower layer of the protocol, the message is revealed in all layers of the sender and the recipient. We might not be concerned about this exposure if we have proper physical safety; an exposure is made on the sender or recipient host or workstation, protected by alarms or locked doors. For instance. However, you should remember that there are two levels of the message from all intermediary hosts where the message can be passed on. This is because it is not in the lower layer where routing and addressing is read, but rather in higher layers. In the intermediate hosts, the message is evident and one of these hosts might not be especially confident.

The user cannot see encryption of links. The encryption is transmitted by the low-level layer of network protocols, such as message routing or communication error detection. The traditional encrypted link message with shaded fields is seen in figure 7-21. Although some data link header and trailer are being used before encrypting the block, some of these blocks are shaded. As the message M is treated on each layer, information about the header and control is inserted on the send ant side and deleted from the receiving side. The hardware encryption devices operate easily and trustworthily; in this situation, both the operating system and the operator cannot see link encryption.

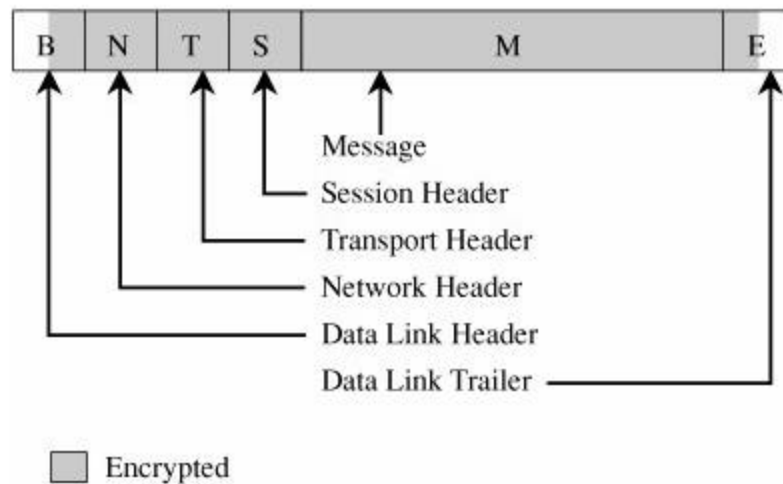


Figure 7.21. Message under Link Encryption.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Link encryption is particularly suitable where the most vulnerable point is the transmission line. If all network hosts are relatively protected, but the media is shared or is not secure with other users, linking encryption is an easy control to use.

End-to-End Encryption

The end-to-end encryption from one end of the transmission to the other provides protection, as the name implies. The encryption can be implemented between the user and the host through a hardware device. The encryption can also be performed by the host computer software. The encryption of the OSI model is carried out at the highest levels of each case (layer 7, application or probably layer 6, presentation). Figure 7-22 displays a model of end-to-end encryption.

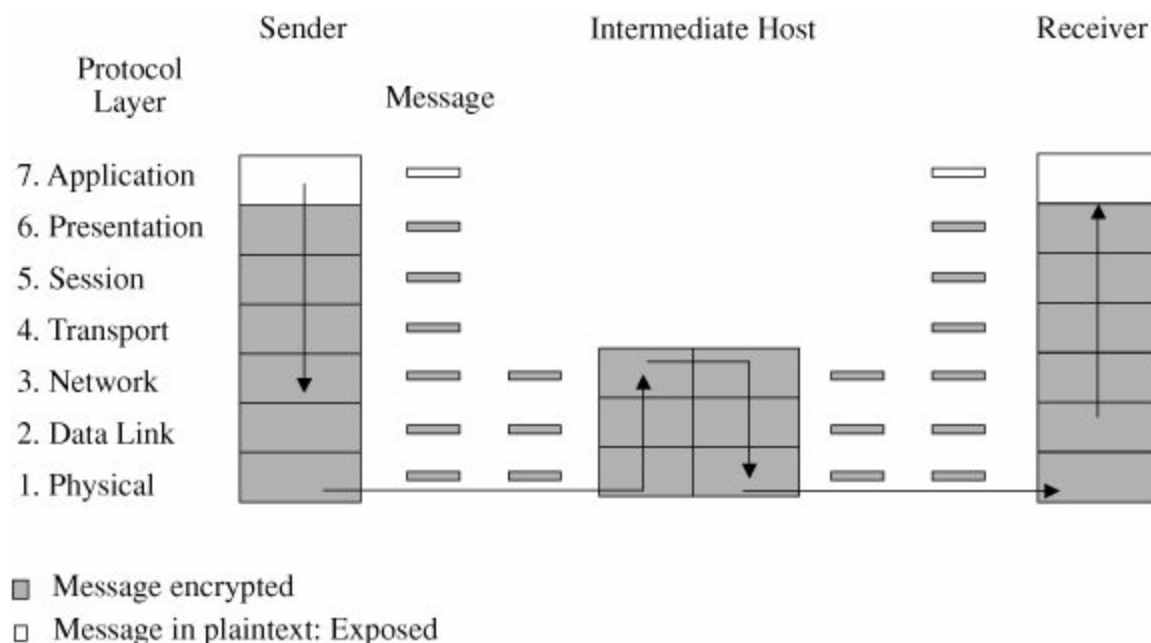


Figure 7.22. End-to-End Encryption.

Above Image taken from the book **Security in Computing, Fourth Edition** By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

As the encryption takes priority over the whole routing and transmission processing of the layer, the message is sent to the thought the entire network in encrypted format. Encryption addresses possible defects in the lower layers of the model of transition. Should the lower layer fail to preserve and disclose received data, the confidentiality of the data is not compromised. The standard End-to-end encryption message, again with the encrypted field shaded, is shown in Figure 7-23.

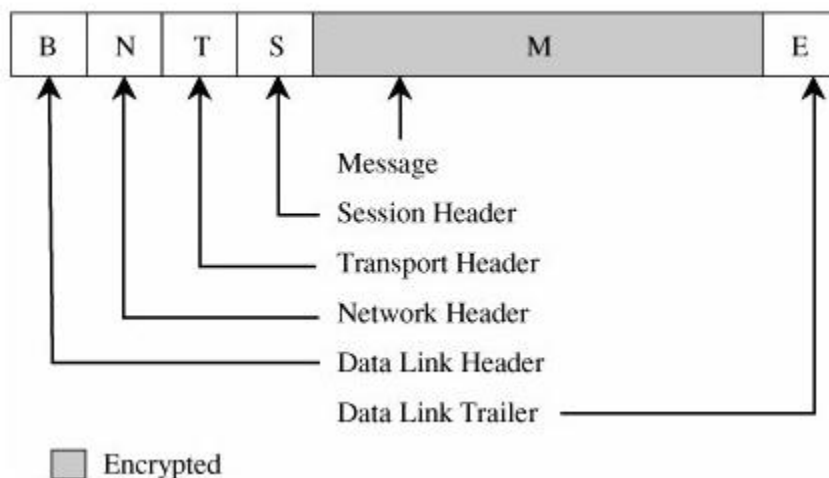


Figure 7.23. End-to-End Encrypted Message.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The messages sent via multiple hosts are protected when End-to-End encryption is used. As shown in Figure 7-24, the data contents of the message are still encrypted when transiting (protected from disclosure). So, while a message must pass through possibly unknown nodes on the path between A and B (such as C through G), the message is shielded from being exposed in transit.

Comparison of Encryption Methods

It's not completely clear simply that a message is encrypted that during or after transmission it will not be exposed. However, in many cases the encryption strength is sufficient protection, taking into account the possibility of the broken encryption of the interceptor and the reliability of the message. As with many security aspects, the strength of protection must be balanced with the possibility of attack.

With link encryption, encryption is implemented with all communications along a particular link. Usually, a defined host has only single link into a network, indicating that all network traffic started on that host would be coded by such a host. But this encryption scheme guarantees that every other host accepting these communications should be able to decrypt messages as well. In addition, both hosts need keys to share. On the way to its final destination, a message can pass through one or more intermediate hosts. When some of the network links but not others encrypt the message, some of the encryption advantages are lost. Link encryption is then generally carried out on all network links if the encryption is completed.

In comparison to the "logical links," channels between two processes at a level well above physical path is end-to-end encryption is implemented. Since intermediary hosts will not have to encrypt or decrypt message along a transmission path, they will not need cryptography facilities. Therefore, encryption should only be used for the messages and applications it needs. In addition, software can be used for encryption, so that we can selectively apply it to an application or even to a single message within a specific application.

The limited benefit of end-to-end encryption is also a key weakness. There is a virtual encryption network between each pair of users under End-to-End Encryption. Each pair of users should share a distinct encryption key to ensure proper security. Thus, the number of keys required is equal to the number of users pairs, which for n users is $n * (n-1)/2$. This number grows exponentially with a growing number of users. This number, however, assumes the use of single key encryption. Only a few keys are required per recipient with a public key system.

As shown in Table 7-5, the user can rapidly and effortlessly encrypt the link, and then use less keys. End-to-end encryption are more versatile, selectable, user-level and integrated with the application. For all cases, neither type is right for all situations.

Table 7-5. Comparison of Link and End-to-End Encryption.	
Link Encryption	End-to-End Encryption
Security within hosts	
Data exposed in sending host	Data encrypted in sending host
Data exposed in intermediate nodes	Data encrypted in intermediate nodes
Role of user	
Applied by sending host	Applied by sending process
Invisible to user	User applies encryption
Host maintains encryption	User must find algorithm
One facility for all users	User selects encryption
Typically done in hardware	Either software or hardware implementation
All or no data encrypted	User chooses to encrypt or not, for each data item
Implementation concerns	
Requires one key per host pair	Requires one key per user pair
Provides node authentication	Provides user authentication

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

In certain cases, it is possible to use all forms of encryption. An end-to-end encryption is also possible if a user does not trust the quality of the link encryption supplied by the device. A system administrator concerned about the security of an application-applied end-to-end encryption scheme may also install a link encryption device. This duplication of the security effect has little negative effect if both encryptions are relatively fast.

Virtual Private Networks

Link encryption could be used to give users of a network the impression that they will be part of a private network even if it belongs to a public network. For this purpose, the approach is called a virtual private network. (or VPN)

Physical security and administrative security are usually sufficiently strong to secure communications within the network perimeter. Therefore, the user's greatest access is between the workstation or user and the host network or server perimeter.

A firewall is a device for access control between two networks or two segments of a network. It filters every traffic between both the protected or "within" network and the network or segment that is less trustworthy or "external"

A VPN can be implemented through a lot of firewalls. The user can request a VPN session from the firewall if they first communicate with the firewall. The user's client and firewall negotiate a session key, which is then used by the firewall and the client to encrypt all traffic among them. Thus, only those with privileged access by the VPN are constrained by the larger network. In other words, even if it is not, the user feels that the network is private. The VPN tells us that an encrypted tunnel or tunnel is the communication. Figure 7-25 shows the creation of a VPN.

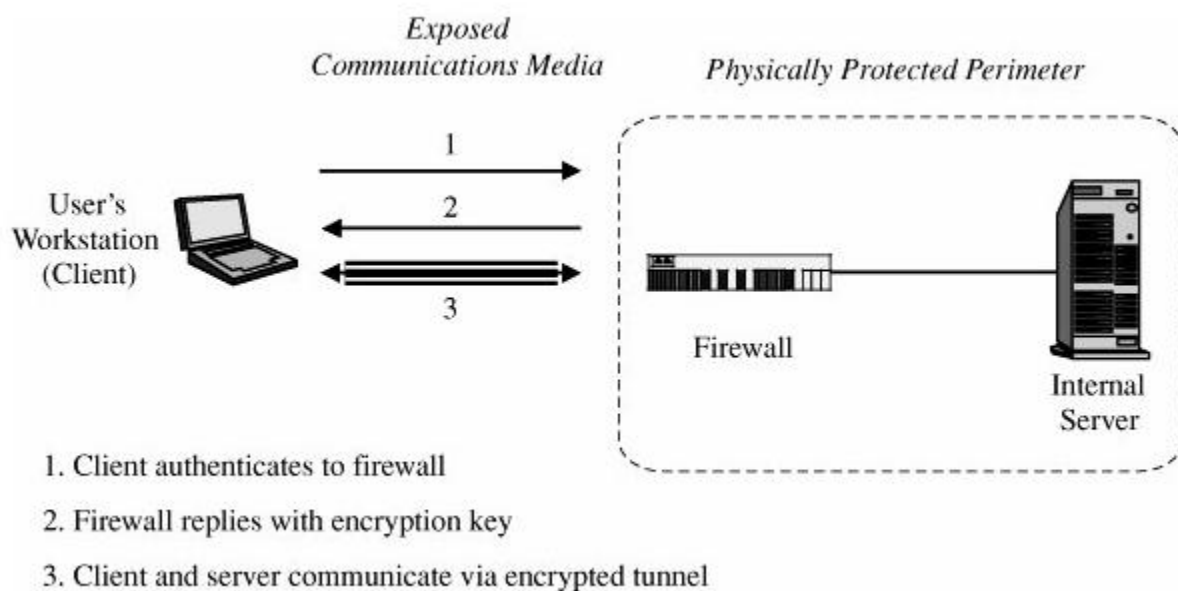


Figure 7.25. Establishing a Virtual Private Network.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

When the firewall communicates with an authentication service within the perimeter, virtual private networks are established. The firewall can pass the user authentication data to the authentication server and the firewall provides users with the necessary security privileges upon approval of the authenticated identity. A known, trusted person may be permitted access to the resources which are not available to the general users for example, such as an employee or system administrator. This access control is carried out using the VPN from the firewall. Figure 7-26 displays a VPN with privileged access. The firewall communicates the (privileged) user identity to the internal server in this figure 2.

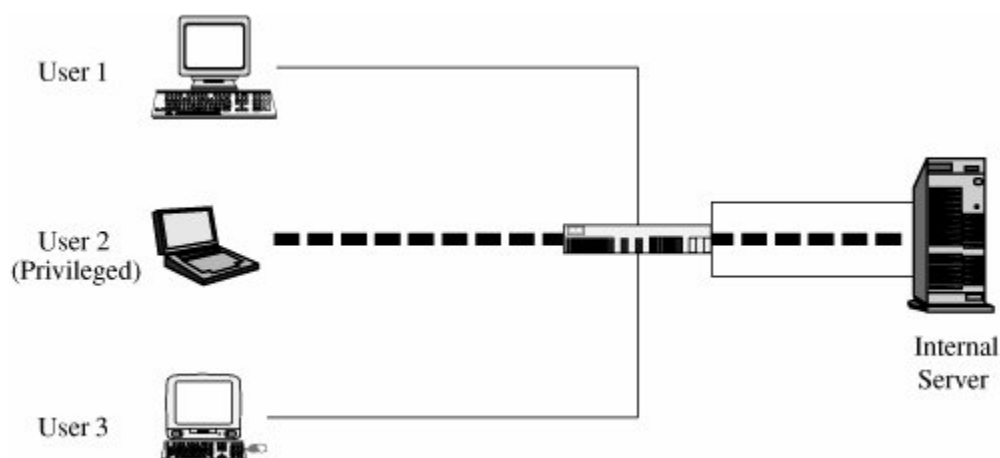


Figure 7.26. VPN to Allow Privileged Access

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

PKI and Certificates

A public-key infrastructure or PKI is a method that users can use, generally in the broad (and often distributed) setting, to implement public-key cryptography. Each user is offered a set of services ,identifying and access control services related to PKI as follows:

- Create certificates with a (public) cryptographic key associating the identification of a user
- Provide your database with certificates
- Sign certificates generating their credibility to the certificate's authenticity
- Confirm or refuse the validity of a certificate
- Deactivate users certificates whose private key has been revealed or no longer allowed tp access

PKI is sometimes classified as a standard, but in simple terms it is a collection of policies, products and processes that leave room for doubt. The policies lay down rules for the operation of cryptographically systems. In particular, policies indicate how the keys and important information are managed and how the control level is compared to the risk level. Procedures specify how to create, handle and use the keys. In conclusion, the products implement the policies and generate, store and manage keys.

PKI establishes entities that enforce PKI certificate policy, called certificate authorities. The basic understanding would be that a certificate authority is trusted, such that users could even assign building, issuing, accepting and revoking of certificates to the authority, the certificate authority's particular actions include:

- Management throughout their whole lifespan of public certificates
- issued certificates via a public key with digital signature, adhering a user or system identity arranging
- Scheduling for certificate expiration dates
- Making sure that certificates are revoked by publishing revocation lists when needed

A certificate authority's functions can be performed in-house or via a commercial service or a trusted third parties.

PKI also includes a registration authority acting as an interface between a user and a certificate authority. The registrant authority collects and authenticates a user's identity and then delivers a certificate request to the concerned certificate authority. The registration authority is similar to the Indian Postal Service; the postal service is acts as an agent in the India. Government's Passport department to allow Indian citizens to obtain passports in order to provide the relevant forms, verify the identity and request an actual passport from the corresponding passport issuing office the certificate authority. The performance of the registration authorities, just like passports, determines the trust levels that can be placed in the issued certificates. PKI most generally fits into a central, hierarchically organized organization such as a governing body.

Most PKI processes are using key identity-binding certificates. However, research work has been carried out to expand the notion of certificate to credentials. For example, a credit card company would be more interested in inspecting your financial status than identity; a PKI system could entail a certificate based on binding financial status to a key. This approach is being follows by the Single Distributed Security Infrastructure (SDSI), including identity certificates, group membership certificates and name-binding certificates: Two related standards are drafted: the Simple Public Key Infrastructure (SPKI) and the ANSI Standard X9.45; the latter only has a set of requirements and a certificate format

PKI is near but not but a mature procedure. Many issues should be resolved, specifically since PKI offers yet to get integrated commercially on a big scale. Table 7-6 lists more than a few issues for being addressed once we find out about PKI.

Issue	Questions
Flexibility	How do we implement interoperability and stay consistent with other PKI implementations?
	Open, standard interfaces?
	Compatible security policies?
	How do we register certificates?
	Face-to-face, e-mail, web, network?
	Single or batch (e.g., national identity cards, bank cards)?
Ease of use	How do we train people to implement, use, maintain PKI?
	How do we configure and integrate PKI?
	How do we incorporate new users?
	How do we do backup and disaster recovery?
Support for security policy	How does PKI implement an organization's security policy?
	Who has which responsibilities?
Scalability	How do we add more users?
	Add more applications?
	Add more certificate authorities?
	Add more registration authorities?
	How do we expand certificate types?
	How do we expand registration mechanisms?

Table 7-6. Issues Relating to PKI.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The protection of the certificates includes administrative procedures. In order to authorize certification requests several operators should be required. To identify hackers and to deter them from issuing false certificate requests, controls must be put in place. Strong encryption and digital signatures may be included. In the event that the system fails or a hacking attack effectively corrupts authentication, a safe audit trail is also needed to reconstruct certificate information.

SSH Encryption

SSH (Secure Shell) is a couple of Unix-defined protocols (versions 1 and 2) that also exist under Windows 2000 to provide an authenticated and encrypted path for the shell or OS command interpretation. Both SSH versions substituted for remote access UNIX utilities such as Telnet, rlogin, and rsh. SSH protects against spoofing attacks and changes in communication data.

The SSH Protocol involves negotiation of the encryption algorithm (for instance DES, IDEA, AES) between local and remote sites and authentication (including public key and Kerberos).**SSL**

Encryption

Netscape originally developed the SSL (Secure Sockets Layer) protocol to secure communication between a web browser and a server. It is now also called TLS for transport layer security. SSL interfaces (e.g. browsers) with TCP/IP protocols to authenticate servers, optionally authenticate clients, and an encrypted communication channel between client and server. Client and server negotiate mutually supported session encryption and hashing, including Triple DES and SHA1, and MD5 RC4 with a 128 bit key.

The client requests an SSL session in order to use SSL. In order to establish the authenticity of the server, the server replies with its public key certificate. The client returns portion of a symmetric session key encrypted underneath the public key of the server. The server and client calculate the session key, and then switch the Shared Session key to the encrypted communication.

The protocol is straightforward but powerful and the encrypted communication protocol most commonly used on the Internet. SSL only defends from the browser to the server (often only to the firewall of the server or, slightly stronger, to the computer running the web application). Data is revealed to the browser from the user's keyboard and from the company of the recipient. Blue Gem Security has produced the product LocalSSL, which encrypts data after typing, and then transfers it to the client's browser via the operating system, thereby preventing any key logging Trojan horse inserted in the user's computer from disclosing any types of users.

IPSec

As noted earlier, the Internet address space is exhausted. The initial 32-bit address structure of the Internet fills up as domain names and devices proliferate. The problem of addressing is addressed by a new structure, called an IPv6 (IP Protocol Suite version 6). The Internet Engineering Task Force (IETF) has had an outstanding chance to fulfill serious safety standards during this restructuring.

The IETF implemented either IPSec or the IP Security Protocol Suite as part of the IPv6 suite. The IPSec protocol describes a standard way to handle encrypted data to resolve basic vulnerabilities such as spoofing, eavesdropping and session hijacking. IPSec is implemented in the IP layer, affecting all of its above layers, particularly TCP and UDP. IPSec also would not need to modify the vast number of existing protocols of TCP and UDP.

IPSec is a bit like SSL because it supports authentication and confidentiality such that there are no major modifications above or below it (in the TCP protocols). Like the SSL, it is designed to be

independent of such encryption protocols and to allow two communicating parties to agree on a collection of protocols which are mutually supported.

The foundation of IPsec is the security association, which is basically a set of security parameters protected for communication channel. It is approximately identical to SSL session. A security group includes

- Encryption algorithm and mode (for instance, DES in block-chaining mode)
- Encryption key
- Encryption parameters, like the initialization vector
- Authentication standard protocol and key
- association lifetime, allowing long sessions to choose a new cryptographic key as much as required
- Secure message sensitivity level (usable for classified data)

A host can have multiple security associations in place for simultaneous communications with different remote hosts, for example, a network server or a firewall. An SPI (security parameter index), the data element which basically points into a table of security associations, is selected for the security association.

The AH (authentication header) and ESP (encapsulated security payload) are the primary data structures in IPsec. The ESP substitutes, as shown in figure 7-27, the standard TCP header and data portion of a packet. The physical header and trailer dependent on the communication medium of the data link and physical layer, including the Ethernet.

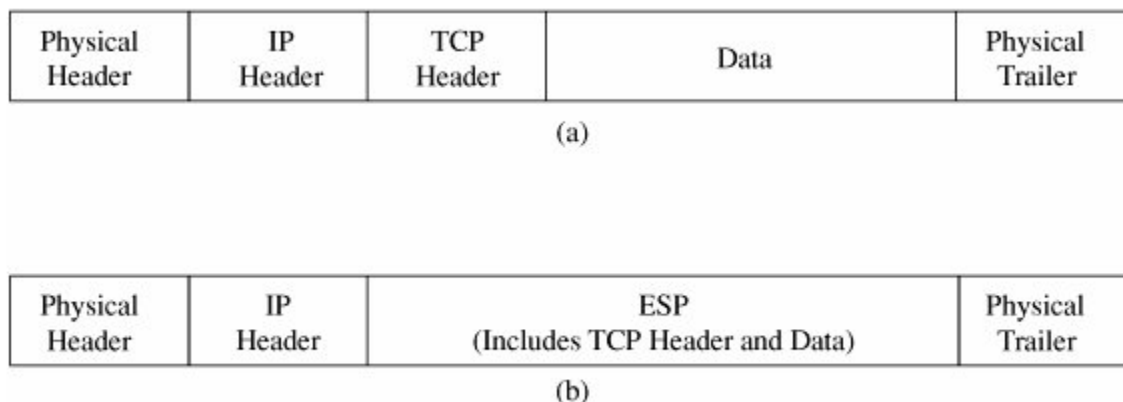


Figure 7.27. Packets: (a) Conventional Packet; (b) IPSec Packet.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

As seen in Figure 7-28, the ESP includes both an authenticated portion and an encrypted portion. For each packet sent to the specific address using the same SPI for sequence number is increased

by one so that packet playback attacks are avoided. Data for payload are the actual packet data. Since blocks of some sizes are required for certain encryption or other security mechanisms, the padding factor and padding length fields provide padding and the sum of padding to guarantee the payload data meets an acceptable length. The following header displays the type of payload data. The field of authentication is used to authenticate the whole object.

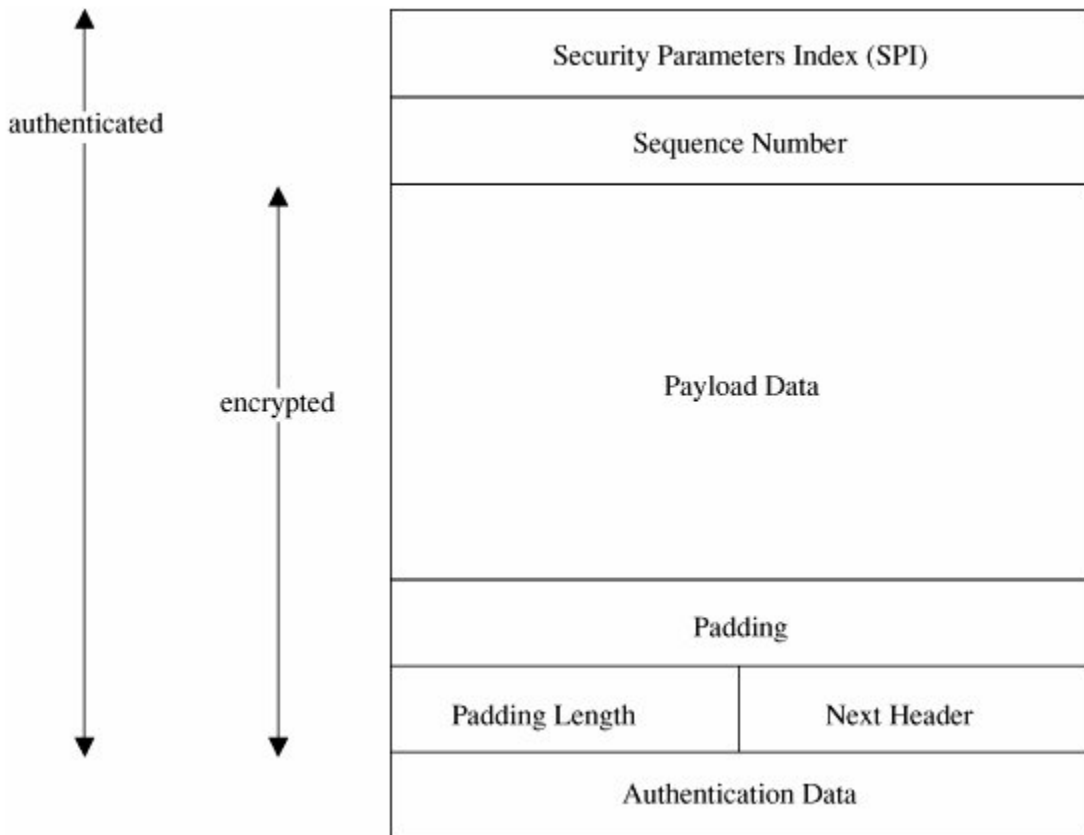


Figure 7.28. Encapsulated Security Packet.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The critical component is key management, identical to most cryptographic applications. IPsec addresses this need with ISAKMP or the Internet Security Association Key Management Protocol. Like SSL, ISAKMP needs that with every security association a distinct key is generated. ISAKMP is a simple, flexible, and scalable protocol. ISAKMP is implemented in IPsec via an exchange of IKE or ISAKMP key. IKE presents a means for protocols, algorithms and keys to be agree upon and managed. IKE uses the DiffieHellman scheme for the key exchange between unrelated parties.

Consider an example of DiffieHellman algorithm

If Alice and Bob wish to communicate with each other, they first agree between them a large prime number p , and a generator (or base) g (where $0 < g < p$).

Alice chooses a secret integer a (her private key) and then calculates $g^a \bmod p$ (which is her public key). Bob chooses his private key b , and calculates his public key in the same way.

Bob knows b and g^a , so he can calculate $(g^a)^b \bmod p = g^{ab} \bmod p$. Therefore both Alice and Bob know a shared secret $g^{ab} \bmod p$. An eavesdropper Eve who was listening in on the communication knows p , g , Alice's public key ($g^a \bmod p$) and Bob's public key ($g^b \bmod p$). She is unable to calculate the shared secret from these values. The two parties now exchange identities and certificates for their shared secrecy in order to authenticate them. Finally, a shared cryptographic key is derived and a security alliance joins.

The key exchange is incredibly efficient: two messages can be exchanged, with an optional additional two authentication messages. As this is a public key approach, for each pair of communicating parties, only two keys are necessary. In an existing security association, IKE has submodes for authentication and the creation of new keys.

IPsec can establish multipurpose cryptographic sessions, such as VPNs, applications and network management at lower levels (such as routing). The IPsec protocols have been written and scrutinized thoroughly.

Signed Code

As we saw, somebody can put malware malicious code on a website that suspicious users download it. If you run with the privilege of someone downloading it, such active code can cause severe harm, from files to emails to Trojan-specific Trojans to subtle, hard-to-recognize Trojan malicious actions. Today, apps and updates can be downloaded from central websites, but there is an increasing risk possibility of downloading anything that malicious.

To reduce this risk, use of the signed code is a partially not complete approach. A trustworthy third party applies a digital signature to a piece of code which is allegedly more confidential. A signature structure in a PKI allows the signature to be validated.

Who will be the trustworthy party? As a code signer will be a well-known manufacturer. And what about the small, virtually unknown system driver manufacturer or a code add-in? If the code vendor is uncertain, it does not benefit the vendor to sign a code for themselves; errors will still publish a code that has been signed by them.

In March 2001, Verisign announced that it had erroneously given two certificates of code signing to someone claiming to be but who was not an employee of Microsoft. These certificates circulated for nearly two months prior to detection of the error. All will know that certificates had not been withdrawn even after Verisign noticed a mistake and canceled the certificates by searching the Verisign list. Most don't ask for Microsoft-signed download code.

Encrypted E-mail

An email is identical to the back of a postcard. The mail carrier and everyone in the postal system whose hands the postal card transfer can read not just the address, but anything in the letter field. We may use encryption to protect message's confidentiality and even its completeness to ensure that the message's privacy or routing information is protected.

As we found in many other applications, it's simpler to encrypt; the more complicated challenge is key management. A hierarchical certificate-based PKI solution for key exchanges and the usage of a standard, flat-specific exchange process are the two major strategies to key management. The hierarchical technique is named S/MIME and many commercial mail processing systems such as Microsoft Exchange or Eudora use this method. A single method is known as PGP and is a commercial extension.

Content Integrity

The integrity of content comes as a cryptography bonus. Without cracking the encryption, nobody can effectively change encrypted data. However, this does not state the encrypted data cannot be altered. Moving a bit of an encrypted data stream also changes the result after decryption, such that the final text is modified significantly. Three potential threats must be considered:

- malicious modifications that effectively change the content
- Malicious or nonmalicious changes which do not necessarily have a meaningful effect on content
- nonmalicious modification that changes content in a manner which is not detectable

Encryption deals very effectively with the first of these threats. We may use other controls to address the others.

Error Correcting Codes

In order to protect against changes in the transmission, error detection and error correction codes may be used. The codes operate similarly to their names: the error detection codes can identify that an error has happened, and the error correction code can correct errors without the initial message having been retransmitted. The error code and the original data are transmitted such that you can recompute the error code and check that the response obtained corresponds to the predicted value.

Parity check is the simplest error detection code. Based on their sum or exclusive OR, one additional bit is added to an existing group of data bits. The two parity are known as even and odd. The even parity added bit is 0 when the sum of the data bits is also even 1 and 1 when the sum of the data bits is odd; In other words, the parity bit is set in order to evenly apply the sum of all data bits and parity bit. Odd parity is the same but the sum is odd. For example, the data stream 01101101 would have an even parity bit of 1 (and an odd parity bit of 0) because $0+1+1+0+1+1+0+1 = 5 + 1 = 6$ (or $5 + 0 = 5$ for odd parity).

A parity bit can reveal that a single bit is modified. Parity does not detect double bit errors in which two bits are modified within a group. This assumes that having a parity bit depends on the fact that

single-bit errors are uncommonly, so two bits would unlikely be modified. Parity suggests only that a bit is altered; it doesn't identify which bit is altered.

Additional error detection codes are available, such as hash codes and Huffman code. Any of the more complicated codes can identify multi-bit errors and can distinguish which bits have been changed (two or more bits changed for a data group).

In cases in which error-transmission equipment is unreliable, noise in communications and interference or other sources of erroneous data changes are detected, parity and simple errors are used to identify and correct error codes.

Cryptographic Checksum

Malicious alteration should be managed in such a way, so both the error detection mechanism and the data bits itself cannot be changed by the attacker. One way is to employ the technique that, according to the value of data bit is decreases and transitions of data.

To see how this approach works, an error detection code is used as a transition in several respects. This ensures that every error detection code decreases a data block to a smaller digest, which relies on the bit value in the block. The proportion of decrease (i.e., the proportion of original block to transformed block size) is related to the efficacy of code in error detection. If an 8-bit data block is reduced to a 1-bit result by coding, half of the 2^8 input values are mapped to 0 and a half to 1, given the output is uniformly distributed. This means that there are $2^8/2 = 2^7 = 128$ various bit patterns, each generating the same 1-bit output. The less input the output maps the less the attacker could modify the value of the input without impacting on its results. For many uses, however, a 1-bit result is too weak or many applications. Each result is $2^8/2^3$ or $2^5 = 32$ inputs if the output is three bits rather than one. For preventing malicious change, a smaller amount of inputs to a particular output is essential.

A cryptographic checksum is a cryptographic mechanism that produces a checksum. It is also called a message digest. The cryptography restricts a data block (the plaintext) being changed from the attacker and changes the checksum value (cypher text) to match. The code tamper protection and the message integrity protection in transit are two main uses of cryptographic checksums. A System Administrator calculates the checksum of each program file on a system for code protection, then calculates and compares the values of new checksum. Since executable code generally doesn't really alter, the administrator could identify unexpected modifications from, for instance, malicious code attacks. Likewise, a checksum on data in communication detects data which have been altered in transmission, maliciously or accidentally.

Strong Authentication

Database systems and operating systems implement a security policy that lays out how the entities, groups, subjects, resources and objects could access. Authentication is essential to this policy: the accuracy of identities are known and confirmed.

Authentication is often required in networked environments. In a network situation, though, the probability of snooping and wiretapping less frequent to non-networked environments could render authentication more difficult to achieve. Before transmitting your password over your network, all ends of a communication can need to be authenticated:

One-Time Password

The possibility of a wiretapping means that you will intercept a password from a user entering password over an unsecured network. A one-time password will protect a remote host from wiretapping and spoofing.

As the name suggests, a one-time password is only appropriate for use for one time. Find the easiest way to see how it works, wherein the user and host have access to similar password lists, such as a one-time cryptographically pad. Users will enter the first login password, the second login password and so on. As long as there are secret password lists is remained and no one else could guess a password. Wiretapping becomes useless by acquiring a password. However, human beings have difficulty for maintaining these password lists, as with one-time cryptographic pads.

To solve this problem, we could use a password token, which creates an unpredictable password that can be checked at the receiving end. The Securid Device from RSA Security, Inc., provides the simplest type of password token. A Random Number is displays by this device, each minute a new number is generated. A separate device is provided for each user (that generates a different random number sequence). The user reads the device display number and types it as a one-time password. The computer runs the algorithm at the receiving end for generating the password for the current minute; the user is genuinely authenticated if the user's password matches the remotely computed one. Since devices may fall out of alignment if one clock is marginally faster than the other, these devices apply reasonably normal drift rules.

What are the advantages and drawbacks of this approach? It's easy to use first of all. The choice of a wiretapped reusing a password is largely counters. It is resistant to spoofing with a powerful password-generating algorithm. The system fails however if the user loses the device or, worse yet, if it falls into the hands of an attacker. Since you build a new password only once a minute, you can find a tiny vulnerability window, which enables an eavesdropper to reuse an intercepted password.

ChallengeResponse Systems

A challenge and response device looks like a basic pocket calculator. The user authenticates the device first, normally using a PIN. A random number is transmitted by the device called the 'challenge,' which the user enters into the device. The device responds this number with a different number that the user transfers to the system.

The system brings a new challenge to the user with any application. This device removes the small vulnerability window through which the user can reuse an authenticator that is time sensitive. Without the PIN, a generator falling into the wrong hands is useless. But the user still has to login

with the response generator and a damaged device refuses the service to the user . Finally, the threat of a rogue remote host is not addressed by these devices.

Digital Distributed Authentication

The 1980s had seen the Digital Equipment Corporation as a concern where non-human entities in a computer system had to be authenticated. A method may, for example, retrieve a user query that reformats it, maybe restricts it and submits it to a database manager. The database manager and query processor both want to guarantee that a specific communication channel between the two is authentic. Neither one of those servers is operated by a human being directly (although each process was, of course, somehow initiated by a human). Human forms of access control are thus inappropriate.

For this requirement, Digital created a basic architecture to tackle the following threats:

- Server impersonation by a rogue process for one of the two authentication servers
- interception or modification of data exchanged between servers
- Replication of the previous authentication

The architecture implies that each server has a private key and that any other process which may need to establish an authenticated channel will access or retain the corresponding public key. A sends a message to B, encrypted with B's public key, to initiate authenticated communications between server A and server B. B decrypts the request and responds with an encrypted message under the public key of A. A and B should be added to the message that is to be encrypted to avoid repeated them.

A and B could create a secure channel by selecting an encryption key and sending it in the authenticating message to the other. Upon completion of authentication, all communication under this secret key can be taken to be as secure as the original dual public key exchange. Gasser suggests a separate cryptographic processor, for example the smart card, to protect the secrecy of the channel, so that private keys are never revealed outside the processor.

Kerberos

In distributed networks, Kerberos is a platform that enables authentication. Initially developed to operate with secret key encryption, Kerberos uses public key technology in its latest version to enable key exchange. In Massachusetts Institute of Technology, the Kerberos system has been developed.

Kerberos is used between intelligent processes such as clients-to-servers and other hosts for authentication. Kerberos is based on the principle of using authenticated tokens known as tickets, to request applications from a central server. A ticket is an authenticated, unforgeable, nonreplayable object. In other words, it is an encrypted data structure which names a user and a service which is allowed by the user. It also has a time value and controls information.

As seen in Figure 7-29, the first step to using Kerberos would be to sessionb in to the Kerberos Server. When a user signs in, the user's ID is sent to the Kerberos server. The Kerberos server checks whether the user is authorized. Two messages are sent by the Kerberos server:

1. A session S_G key for communication with the ticket-granting server (G) and a T_G ticket for the ticket-granting server are communication to the user workstation; S_G is encrypted under the user's password: $E(S_G \& T_G, pw)$
2. A copy of S_G session key and user identities to the ticket-granting server (encrypted under a key shared between the Kerberos server and the ticket-granting server)

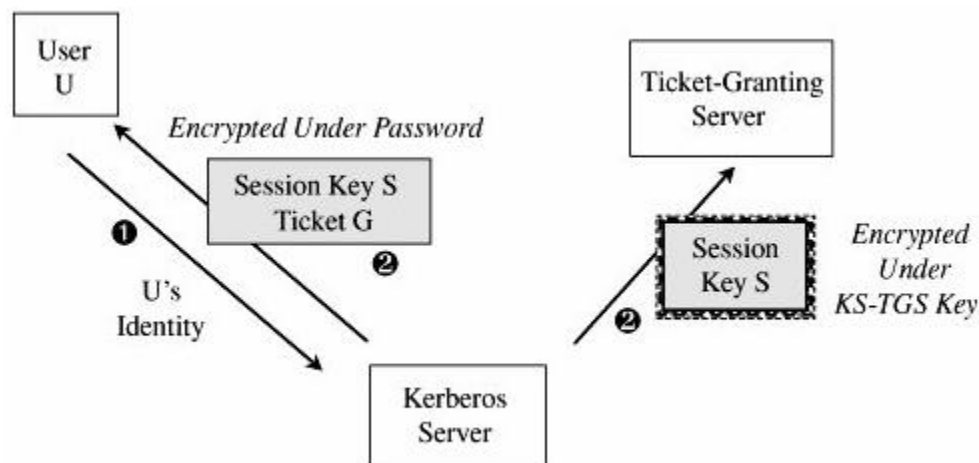


Figure 7.29. Initiating a Kerberos Session.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The user has successfully authenticated the workstation if the workstation will decrypt $E(S_G + T_G, pw)$ with pw , the password of the user.

note that passwords are saved on the Kerberos server, not on the workstation, and that even encryption form , the user password does not need to be transmitted through the network. It is a security advantage to provide keys centrally, but not through the network.

The user would then like to perform other distributed system's services such as file access. The user U requested a ticket to access file F from the ticket-granting server using the S_G key given by the Kerberos server. As Figure 7-30 shows, it returns a ticket and a session key after the ticket-server has validated U's permission to access. This ticket includes an authenticated identity U (in the Kerberos servers ticket U is obtained), an identification of F (file to access), access privileges (for example to read), a file server session key S_F to use for communicating the file to U, and an expiration date for the ticket. The ticket is encrypted by a key exchanged between the ticket-granting server and the file server exclusively. The user U cannot read this ticket, modify it or forge it (or anyone else). Therefore, the ticket-granting server must also give U the copy of S_F , the file server session key. Access requests for other services and servers are similarly handled

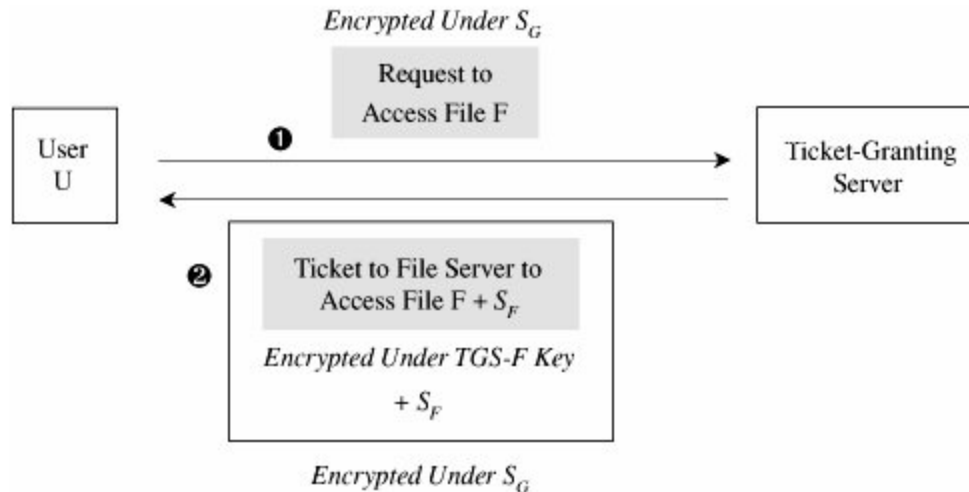


Figure 7.30. Obtaining a Ticket to Access a File.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

In order to resist attack in distributed environments, Kerberos was carefully designed:

- No passwords communicated on the network. A user password is saved only on the Kerberos server as described above. When the user initiates a session, the user password is not forwarded by the user's workstation. (The original user password obviously should be sent, for example in a letter, outside of the network.)
- Cryptographic protection against spoofing. The ticket-granting server, which knows the requester's identity, facilitates each access request by the original authentication of the Kerberos server and by the ability of the user to send an encrypted request under a key, encrypted under the user's password.
- The term of validity is limited. For a limited duration, each ticket is issued; a time stamp is given for each ticket to determine its validity on a receiving server. Thus, certain long-standing attacks, such as cryptanalysis of the brute force, are generally eliminated since the attacker has little time to complete the attack.
- Timestamps to avoid threats from being replayed. A universal clock is a requirement of Kerberos. The time of the request is stamped on the request of each user to a server. A request server compares this time to the current time and only comply with the request when the time is reasonably close to the current time. Since the attacker's presentation of the ticket will be delayed too long.
- Mutual authentication. The customer of a service will ensure the authenticity of the server by requiring an authentication server response. The customer sends a ticket to a service server and then sends the server an encrypted request for that service, given by the ticket-granting server under the session key. The ticket and session key is provided. Only when you have the unique

key it shares with the ticket-granting server does the server decrypt your tickets. The session key is within the ticket, which is the only way to decrypt the user request on the server.

Access Controls

Authentication handles the who of security and safety policy enforcement; entry adjustments enforce the what and exactly how.

ACLs on Routers

Routers carry out the major job of directing community site visitors either to subnetworks they handle or to some other routers for succeeding delivery to additional subnetworks. Routers switch outside IP addresses into interior Macintosh addresses of hosts on an area subnetwork.

Suppose a bunch has been spammed (flooded) with packets from the malicious rogue variety. Routers could be configured with entry control listings to deny usage of certain hosts from specific hosts. Therefore, a router could erase all packets having a source address from the rogue host along with a destination target of the prospective host.

This approach possesses three problems, even so. First of all, routers in large sites perform a large amount of work: They need to take care of every packet getting into and moving away from the network. Incorporating ACLs towards the router demands the router to assess every packet contrary to the ACLs. One ACL provides job, degrading the router's efficiency; as additional ACLs are added in, the router's efficiency may become undesirable. The second difficulty can be an efficiency matter: Due to the volume of do the job they do, routers are made to perform only important providers. Logging of action is usually not really done over a router due to the volume of site visitors and the efficiency charges logging would entail. With ACLs, it might be useful to understand how many packets have been being deleted, to learn if a specific ACL could possibly be removed (in that way improving overall performance). But without logging it really is impossible to learn whether an ACL has been used. Both of these problems together imply ACLs on routers will be most reliable against specific acknowledged threats but they shouldn't be used indiscriminately.

The final restriction on adding ACLs on routers considerations the nature with the risk. A router inspects simply source and location addresses. An attacker typically does not expose an actual supply address. To show you the real origin address will be equal to a lender robber's making his home target and a information of where he projects to retail outlet the stolen funds.

Because someone can simply forge any origin address over a UDP datagram, various attacks make use of UDP practices with false origin addresses so the attack can't be blocked easily by way of a router having an ACL. Router ACLs are of help only when the attacker delivers several datagrams with exactly the same forged source tackle.

In rule, a router is a superb point of admittance control since it grips every packet getting into and moving away from a subnetwork. In particular situations, largely for inner subnetworks, ACLs may be used effectively to limit certain traffic moves, for example, to make sure that only specific

hosts (addresses) get access to an internal community management subnetwork. But also for large-scale, general visitors screening, routers will be less beneficial than firewalls.

Firewalls

A firewall is a security device for the network, which monitors input and output traffic and allows or blocks data packets based on a set of rules of security. Its aim is to create a barrier, between the internal network and incoming traffic from external sources (such as the Internet) for interlocking malicious traffic such as viruses and hackers.

Wireless Security

Benefiting from the fact that wireless computing are so vulnerable, they require measures to ensure that communications between a client and a base station or access point are protected. Recalling that all of these communications are on predetermined radio frequencies, you can anticipate an attacker to intercept and to impersonate. Protective parts find the point of access, authentically select the remote computer to the access point, and vice versa.

SSID

A service set identifier (SSID) is a sequence of characters named uniquely by a wireless local area network (WLAN). An SSID is sometimes called a "network name." which can be used to connect stations to the desired network if multiple independent networks operate in the same physical area.

Wireless devices such as phones and laptops search for networks broadcasting their SSIDs from the surrounding area and have a list of names. By choosing a name from a list a user will start a new network connection.

A Wi-Fi scanner also decides if each network has wireless security options enabled, in addition to obtaining a network name. Typically, a protected network is identified with a lock symbol next to the SSID.

WEP

A security protocol Wired Equivalent Privacy (WEP), as defined in IEEE Wireless Fidelity Standard, 802.11b) to provide a security and privacy level equivalent to the ordinarily anticipated wired LAN for a local wireless area network (WLAN). A wired local area (LAN) network is normally secured by physical security mechanisms (for example, managed access to a building) that operate in a controlled physical environment but may not be successful for WLANs because radio waves are not constrained by the walls of the network.

WEP aims to ensure that data encrypting over the WLAN provide the same protection as the physical security measures of the wired network. Data encryption protect insecure wireless client and access point links; after this is accomplished, other standard LAN security protocols may be placed in place to guarantee privacy, including password protection, end to end encryption, private virtual networks (VPNs) and authentication.

WEP uses a method of Data Encryption based on a combination of key values created by users and systems. The initial WEP implementations supported 40 bits encryption keys with 24 additional data bits system-generated, resulting in a total length of 64 bits keys. These encryption methods were later extended to support longer key such as 104-bit modifications (128 bits overall), 128-bit variants (total 152 bits), and 232-bit variations (total 256 bits).

WEP encrypts the data stream with these keys when implemented with a Wi-Fi connection in order that they are no longer human-readable, nor can be used to receive devices. The keys are not distributed through the network but are saved in wireless network adapter or Windows registry.

WPA and WPA2

Wireless Protected Access (WPA) is a security standard for users of wireless internet-based devices. The Wi-Fi Alliance developed WPA to provide sophisticated data encryption and better user authentication than the original WI-FI security standard, Wired equivalent privacy (WEP). The IEEE-ratified new standard, 802.11i, was designed to be backdated to the WEP to encourage fast, easy adoption. Network security professionals could use a simple firmware update to support WPA on many WEP-based devices.

For enterprise customers and for personal use WPA has distinct modes. The enterprise mode, the WPA-EAP, uses stricter 802.1x with Extensible Authentication Protocol authentication (EAP). The personal mode, WPA-PSK, uses preshared keys to promote customer and office implementation. Enterprise mode involves the use of a server for authentication. The Temporary Key Integrity Protocol (TKIP) for WPA encryption. TKIP consists of a per-packet mixing function, a message integrity check, an extended initialization vector and a re-keying mechanism. WPA supports powerful 802.1x and the Extensible Authentication Protocol (EAP) based user authentication.

In 2004, WPA2 was replaced by WPA. WPA2 utilizes the Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP). It bases on the mandatory Advanced Encryption Standard algorithm which allows authenticity and completeness of messages and is much more reliable and stronger than the original TKIP WPA protocol.

WPA2 has vulnerabilities; specifically unauthorized access to the wireless enterprise network where some Wi-Fi Protected Setup (WPS) connection points invade an attack vector. The invader will work for the state-of-the-art computer technology over several hours .It is recommended the WPS be disabled for each attack vector access point in WPA2 to discourage such threats.

Alarms and Alerts

The logical view for network protection is like in Figure 7-32, which provides protection layers for the internal network, both a router and a firewall. Let us now add another security layer

An intruder detection system is a device inside a network that monitors what is occurring in the network. If an attacker passes through the router and passes the firewall, it is possible to determine

the attack initially, in progress or once an intrusion detection system has occurred. An alarm is activated by intrusion detection systems which could take defensive action.

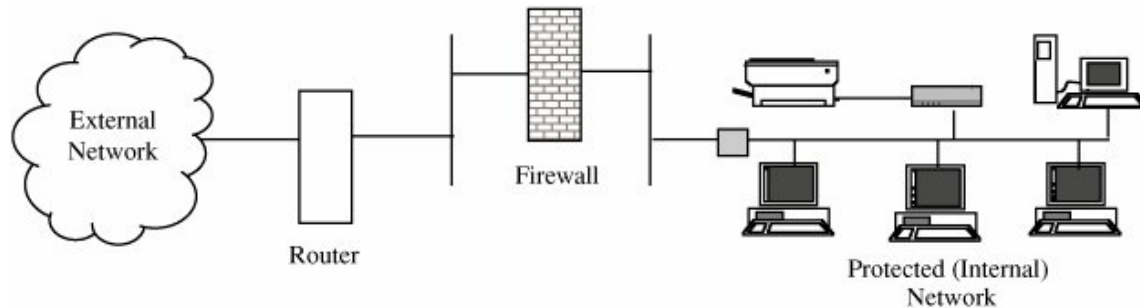


Figure 7-32. Layered Network Protection.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Honeypots

The word “honeypot” refers to a security structure or mechanism built to deflect attackers in computer terminology. In other words, there is a honeypot to deflect the intruders from the organization's valuable assets.

Honeypot could be described as a system that is attached to the network. It has been designed to become a decoy. It attracts hackers and waste their time trying to gain unauthorized access to the organization’s network or systems.

Honeypots are mainly used by large cyber security companies and organizations. It helps cyber security researchers know about the various kinds of attacks by attackers. Even cyber criminals are suspected of using these honeypots to trick researchers and disclose false information.

The cost of a honeypot is relatively high, because it requires specialist skills and resources so that a honeypot provides the resources of an organization to prevent attacks and access to any production system.

Traffic Flow Security

We've looked so far at controls which cover the most common network threats: eavesdropping encryption, authentication methods, intrusion detection systems for ongoing attack, structural fault architecture. We have listed threats, including a traffic flow threat. If the attacker detects an exceptional traffic volume between two locations, the attacker can detect the presence of an event.

The countermeasure against threats to traffic flow is to cover up traffic flow. One means, although expensive and perhaps harsh, of masking traffic flows is to ensure a steady volume of traffic between two points.

A and B may agree to pass identifiable but meaningless encrypted traffic when traffic between A and B is encrypted in a way that the attacker can only find the numbers of packets which are flowing. There will be few pointless packets when A has much to communicate to B. When the communication is light, A will pack the traffic stream with several false packets.

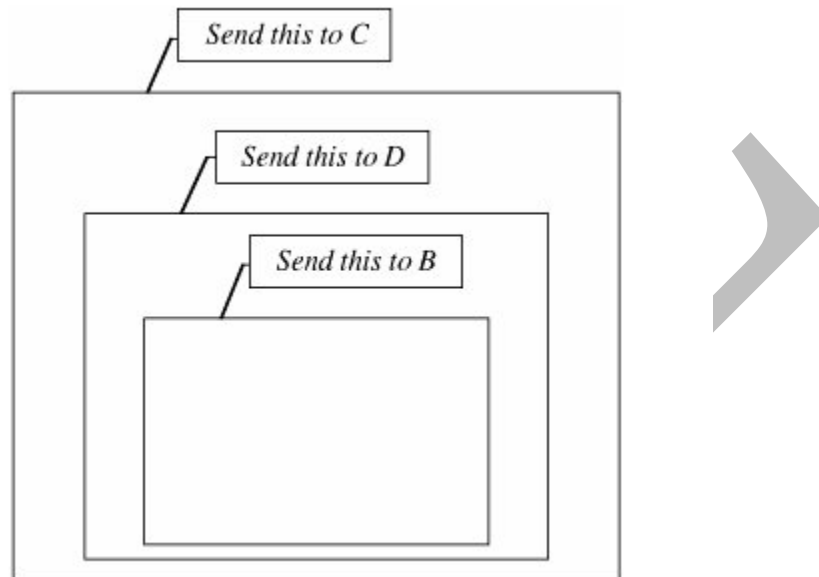


Figure 7-33. Onion Routing.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

C knows that the package was received from A, though C doesn't know whether A was the originator or the point of intermediation. C will then unwrap the outer layer and send it to D. C cannot at this point know whether or not D is the final recipient. C sends a message to the next layer unwrapping D. D knows neither the original source of the package nor the final destination of the package. D transfers the package to its final destination, B.

This scheme does not allow any intermediate recipient to understand where or where the package originated other than the original sender and the ultimate recipient. In this scheme, content, source, destination and routing are confidential.

7. 4. Firewalls

What Is a Firewall?

A firewall is a system to prevent unauthorized access from or to a private network. A firewall can be implemented in hardware or in software form or both combined. Firewalls avoid unauthorized internet users having access to private networks connected to the internet, in particular intranets. Every message you enter or exit from the intranet (the local network to which you are logged) must go through the firewall to examine each message and to block those that do not meet the security criteria specified.

Design of Firewalls

A reference monitor must be

- always invoked
- tamperproof
- small and simple enough with regard to rigorous analysis

A reference monitor is a specific form of firewall. By placing a firewall carefully within a network, all of the network accesses we want to control have to be passed through. This limitation complies with the condition "always invoked" Typically, a firewall is well isolated, which makes it very much immune to modifications. A firewall is usually implemented on a separate computer, with direct links only to external and internal networks. The "tamperproof" requirement is expected to meet this isolation. And firewall designers strongly advise to keep the firewall functionality simple..

Types of Firewalls

Firewalls include a wide range involving capabilities. Forms of firewalls incorporate

- packet filtering gateways or even screening routers
- stateful inspection firewalls
- application proxies
- guards
- personal firewalls

Packet Filtering Gateway

The easiest and most effective type of firewall in some situations is a packet filtering gateway or screening router. A gateway packet filtering controls packet access by the packet address (source or destination) or specific type of transport protocol (such as HTTP web traffic). The ACLs on routers can seriously impede their performance. But behind (on the local side) a separate firewall can screen the traffic on the router before it reaches the protected network. A packet filter that blocks access from (or to) addresses in a single network is displayed under figure 7-34; a filter allows HTTP traffic and blocks Telnet protocol traffic.

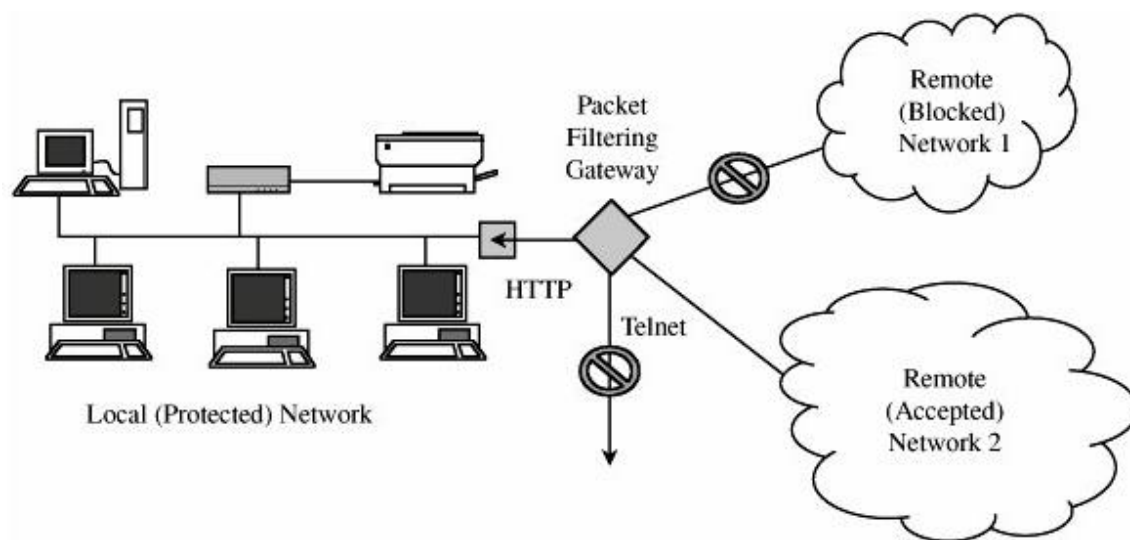


Figure 7.34. Packet Filter Blocking Addresses and Protocols.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Suppose, for example, that an international company has three LANs at three sites globally, as shown in Figure 7-35. The router has two sides in this example: indoors and outdoors. We say that the local LAN is within the router and the two remote LANs are connected to the outside by large area networks. The company may only want communication between the company's three LANs. You could use the LAN screening router at 100.24.4.0 so that you can only use 100.24.4.0 for communications to host, and you can only use 144.27.5.3 or 192.19.33.0 for communication

addresses.

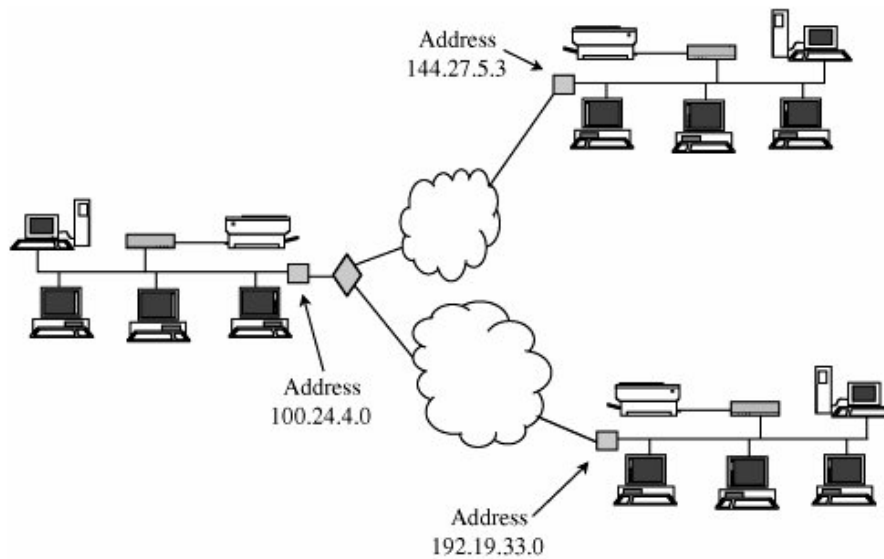


Figure 7-35. Three Connected LANs.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Packet filters don't look at packets inside; only on IP addresses and ports do they block or accept packets. Any information on the packet field is therefore beyond the capacity of a packet filter (e.g. allowing certain Telnet commands while blocking other services).

Packet filters can provide the very important service for inside addresses to be valid. For the reasons described as LAN characteristics, indoor hosts generally trust other indoor hosts. But only by the address shown on the source field of a message can an inner host distinguish another host. Source addresses can be forged in packets to allow an internal application to think that it was communicating with another host on the inside rather than with an outside forger.

A packet filter is located between the internal network and the external network so that you know whether a packet forges an internal address from outside, as Figure 7-36 indicates. A filter screening packet may be configured to block any external packet that claimed its source address was an internal address. The packet filter blocks all packets that claim to come from any address in the 100.50.25.x in this example (but, of course, it permits in any packets with destination 100.50.25.x).

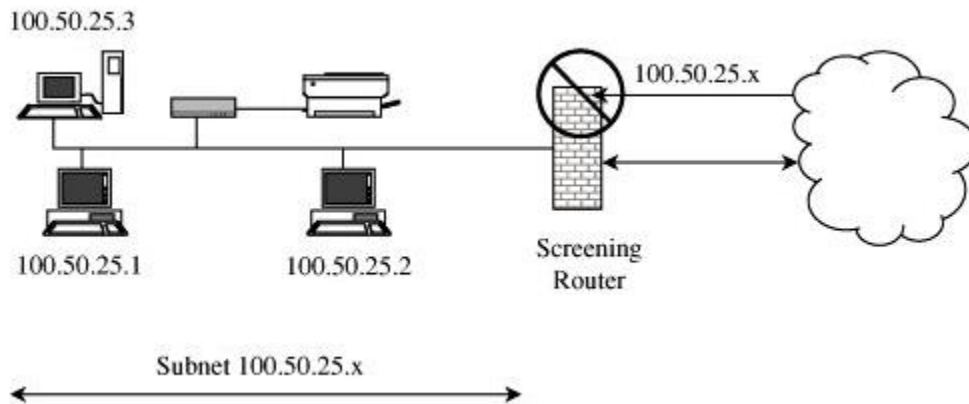


Figure 7-36. Filter Screening Outside Addresses.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

A combination of simplicity and complexity is the main drawback of routers filtering packets. The inspection of the router is simple; the filtering rules described have to be very thorough to perform sophisticated filtering. A comprehensive collection of rules would be challenging and therefore error-prone. For instance, it is easy and straightforward to block all port 23 traffic (Telnet). However, if any Telnet traffic is allowed, the rules should show each IP address from which they are allowed, so the rule set can be very long.

Stateful Inspection Firewall

Filtering firewalls operates on packets one by one, approves or rejects every packet and continues to the next one. No "state" or "context" definition is found between packets. A stateful inspection firewall keeps state information in the input stream from one packet to another.

An attacker's conventional strategy is to break down an attack into many packets by requiring some packages to have very short lengths such that a firewall cannot detect a signature of an attack divided into two or more packets. A stateful firewall inspection will monitor the sequence of packets and conditions in order to avoid a firewall attack.

Application Proxy

Packet filters only look at packet headers, not the packet data. Therefore, if its screening regulations allow incoming connections to port 25, the packet filter will pass everything to port 25. However, programs are complicated and involve errors often. Furthermore, the programs (like the e-mail delivery agent) also operate on behalf of all users (for example, to store incoming mail messages so that inside users can read them). A faulty app with the privileges of all users can cause a lot of harm.

An application proxy gateway, also referred to as a bastion host, is a firewall which imitates the applications' effects so that the application receives only requests to act correctly. A two-headed device is the proxy gateway: it looks inside as though it is an outside connection, but outside behaves just like the insider.

An application proxy executes pseudo-applications. For example, when e-mail is sent to a place, a sending process at one place and a receiving process at its destination communicates via a protocol that sets out the validity of a mail transfer and transfers the message. The sender-destination protocol is carefully specified. A proxy gateway effectively enters the middle of a protocol exchange, pretending to be a communication destination with the sender outside the firewall and to be the sender in communication with the actual inside destination.

Consider the FTP (file transfer) protocol as an example of application proxy. Based on file commands fetch (get) files remotely, store (put) files to a remote host, list files (ls) on the remote host directory and position the process (cd) at any point in your remote host directory tree. Some administrators might want to allow blocking and listing only those files or not allowing changes from a certain directory (so that an outsider could retrieve only files from a prespecified directory). The proxy will simulate the exchange between the two sides. For example, the proxy might accept get commands, reject put commands, and filter the local response to a request to list files.

To understand the true reason for a proxy gateway, why don't we consider several cases.

- An organization wishes to create a price list online so that externals can display the products and prices available. It is certain (a) that no external party can alter the prices or the product list and (b) that external parties can only have access to the price list rather than any more sensitive files inside.
- A school requires its students to access information on the Internet from World Wide Web resources. In order to provide effective service, the school would like to know which websites were accessed and which files had been caught from those websites.
- In a data base system, a government agency wishes to answer queries. The agency wants to limit queries to return the mean of a range of less than five values, however, because of inference attacks against databases.
- A multi-office company would like to encrypt the data part of all mail to other offices. (The encryption is deleted by a corresponding remote proxy.)
- An enterprise will allow its employees to dial in without exposing the company's resources to remote non-employees' login attacks.

A proxy may be applied to each of these requirements. During the first case, the proxy is going to monitor the data on the file transfer protocol so that it can only access the price list file and only read and not alter this file. A logging process as part of the web browser might satisfy the school's requirement. The Agency's requirements can be met with a special purpose proxy, which interacts with the database management system, performs queries, but obtains the number of values from which the response has been calculated and adds small sample size random errors. A specially written proxy which needed strong user authentication (e.g. challenger response system) could

handle the requirement for restricted login that many operating systems do not need. Figure 7-37 displays these functions.

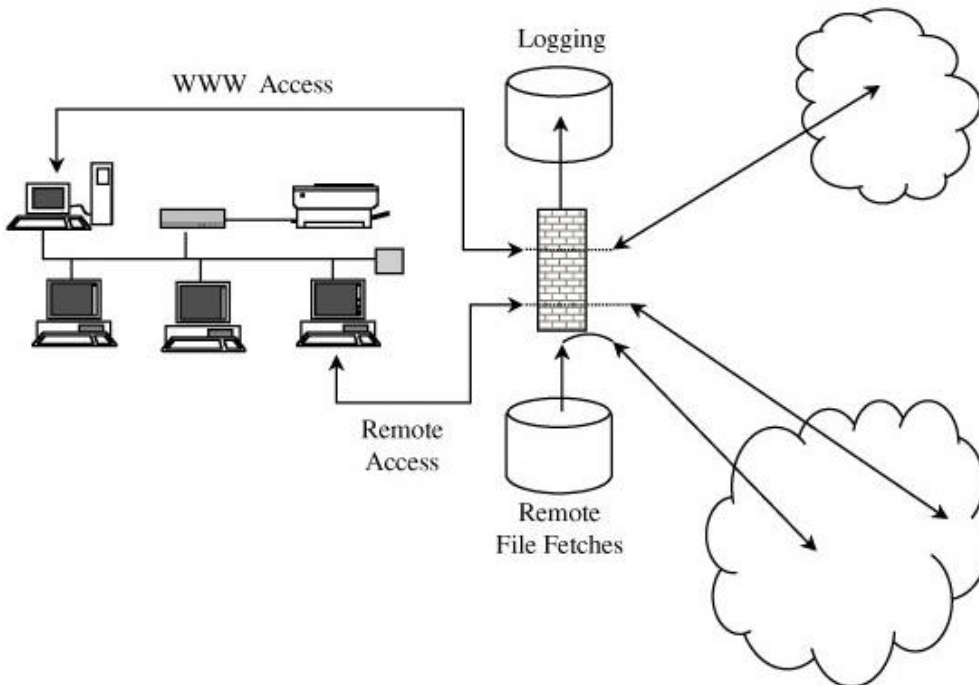


Figure 7-37. Actions of Firewall Proxies.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The firewall proxies can be customized to unique specifications, including logging access data. They even have a standard user interface to which the internal functions can be different. Suppose there is a mix of operating system types in the internal network which does not accept strong authentication with a challenge answer token. This proxy will require strong authentication (name, password and challenger response, confirm the challenger response itself and then only offer simple authentication names and passwords in the form needed by a particular operating system of your internal host.

The difference between a proxy and a screening router is that it interprets the protocol stream through an application to monitor firewall behavior based on the stuff visible within the protocol and not just from external header data.

Guard

A guard is an advanced firewall. Like the proxy firewall it receives, interprets and transmits protocol data units that achieve either the same or changed result through the same or different protocol data units. In accordance with their available information, the guard determines which

services the user has to perform for him, for example something he can know (outside of) his identity, previous experiences etc. Only what is calculable restricts the degree of control a guard can have. Yet firewalls and proxy guards are very close to the difference between them. That is, we can add functionality to a proxy firewall until it starts to look a lot like a guard.

Guard activities could be very sophisticated, just as illustrated in the next examples:

- In the past few days, a university requires its students to be able to use e-mails to a small extent in so many messages. Although e-mail operators can change this result, the common point from which all e-mail flows is the mail transfer protocol can be controlled more easily.
- A school wants the students to access the World Wide Web but only so many characters are allowed per downloaded image because the slow speed of its connection to the web (that is, allowing text mode and simple graphics, but disallowing complex graphics, animation, music, or the like).
- A library would like to make certain documents accessible. However, to ensure that a user can only get the first so many characters in the document to reasonably use copyrighted content. The library demands that the user pay a charge which is transmitted to the author after this amount.
- An enterprise wants to let its employees fetch files through ftp. But it first passes all incoming files through a virus scanner in order to avoid the entry of viruses. Since many of these files are text or graphics which cannot be executed, the company administrator believes the cost to search them (which ought to happen) is negligible.

Personal Firewalls

The (sub)network of multiple host firewalls is usually secured. University students and office workers have a real firewall behind them. Cable modems or DSL connections with unrestricted and on-going access are increasingly used by home users, individual employees and small enterprises. These people need a firewall but a separate firewall device may seem too complicated and costly to secure a single workstation. These people need the power of a firewall at a lower cost.

A personal firewall is a workstation application program which typically blocks unwanted traffic from the network. A personal firewall can add to a traditional firewall, or can make up for the lack of standard firewalls, such as in private DSL or cable modem connections, by filtering the type of data which a host can allow.

Like a network firewall that shows input and output traffic, a personal firewall displays traffic at a single workstation. A workstation might be vulnerable to malicious code or malicious active agents, leaking personal information from the workstation, and vulnerability scans to detect possible vulnerabilities (ActiveX controls or Java applets). Norton Personal Firewall from Symantec, McAfee Human Firewall and Zone Warning from Zone Labs are commercial implementations of personal firewalls (now owned by CheckPoint).

The personal firewall is set up to execute such policies. In particular, the user can choose to highly reliable some sites, such as computers on the network of the organization, but not most other sites. The user determines a policy to allow the download of code from the corporate section, unlimited data sharing and manage access, but not from other sites. Personal firewalls also can create access logs, which can be helpful if the firewall slips into something dangerous.

It is safe and powerful to combine a virus scanner with a personal firewall. Users usually fail to run a virus scan every day, but often during the week they remember to run it. However, keeping the execution of the virus scanner within the user's memory means that the scanner can only identify an issue when a virus is downloaded through an e-mail link. The firewall directs all incoming e-mails, by combining a virus scanner with a personal firewall to the virus scanner, which scans and attachment before it enters the target host and before it opens.

Comparison of Firewall Types

We can summarize the differences among the several types of firewalls we have studied in depth. The comparisons are shown in Table 7.8.

Table 7-8. Comparison of Firewall Types.				
Packet Filtering	Stateful Inspection	Application Proxy	Guard	Personal Firewall
Simplest	More complex	Even more complex	Most complex	Similar to packet filtering firewall
Sees only addresses and service protocol type	Can see either addresses or data	Sees full data portion of packet	Sees full text of communication	Can see full data portion of packet
Auditing difficult	Auditing possible	Can audit activity	Can audit activity	Can and usually does audit activity
Screens based on connection rules	Screens based on information across packets in either header or data field	Screens based on behavior of proxies	Screens based on interpretation of message content	Typically, screens based on information in a single packet, using header or data
Complex addressing rules can make configuration tricky	Usually preconfigured to detect certain attack signatures	Simple proxies can substitute for complex addressing rules	Complex guard functionality can limit assurance	Usually starts in "deny all inbound" mode, to which user adds trusted addresses as they appear

Above Image taken from the book **Security in Computing, Fourth Edition** By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Example Firewall Configurations

See a few examples of how firewalls can be used. We present scenarios that explain how a firewall completes a sensitive architecture and security policy.

Figure 7-38 demonstrates the simplest use of a firewall. The environment is located between the internal LAN and external network connection by a screening router. This installation is always sufficient if we only need to show a router's address.

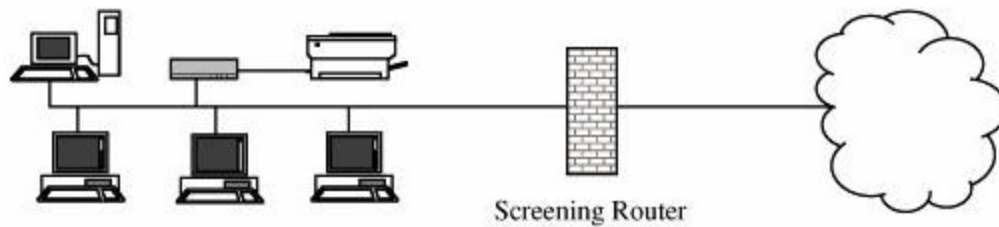


Figure 7.38. Firewall with Screening Router.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

But this company is not suitable for using a proxy machine. Similarly, it is difficult to configure a router for a complex collection of addresses accepted or refused. If the firewall router is attacked successfully, then all LAN traffic that is connected to the firewall can be seen. A proxy firewall is also built on its own LAN to help reduce this exposure as shown in Figure 7-39. Thus, traffic that enters and exits the firewall is the only traffic accessible on that LAN.

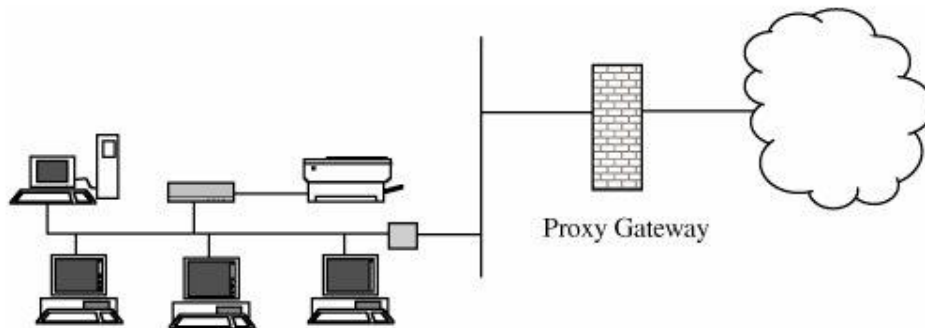


Figure 7.39. Firewall on Separate LAN.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

We can add to this configuration a screening router, as shown in Figure 7-40, for even more security. The proxy firewall proxy filtering router guarantees that the proxy firewall is right (so that an external attacker forges an address from the inside host cannot trick the proxy firewall); the proxy firewall filters traffic according to the proxy laws. Also, only the proxy firewall traffic on the internally secured LAN is visible if the screening router is subverted.

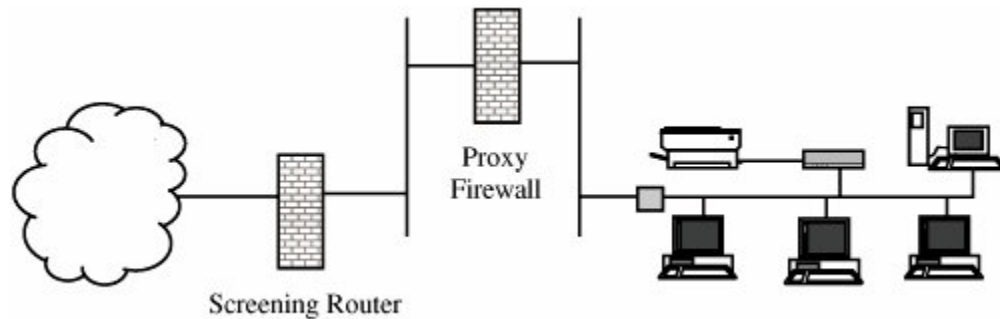


Figure 7.40 Firewall with Proxy and Screening Router.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

What Firewalls Can and Cannot Block

As we have learned, not all computer security issues are solved by firewalls. A firewall only protects its environmental perimeter from attacks by outsiders who are involved in running code in a secure environment or accessing data on their computers. Keep these firewall points in mind.

- Only if the firewalls monitor the entire perimeter will firewalls safeguard an area. Firewalls are only successful when the perimeter is not breached by any unmediated connections. If even an internal host links to an external address, for instance via a modem, the whole internet is vulnerable with the modem and their host.
- Firewalls do not shield data outside the perimeter; firewalls are as exposed as if no firewall had been passed through the firewall.
- Firewalls are the most prominent component of an outer installation, making them the most enticing attack target. Therefore, several different security layers called in-depth defense are better than depending on just one firewall's power.
- Firewalls should be configured properly, this configuration must be modified as internal and external changes in environmental conditions, and firewall notifications for attempted or active intrusions must be checked on a regular basis.
- For penetrators, firewalls are targets. A firewall is designed to prevent attacks, but it isn't impenetrable. Designers purposely keep firewalls tiny so they don't have any other tools, like compilers, linkers, loaders or the like, to continue the attack, even if a penetrator breaks them.
- Just small control over the material admitted inside firewalls exercises, which means that the incorrect data or malicious code is managed inside the perimeter by other methods.

7.5. Intrusion Detection Systems

An Intrusion Detection System (IDS) is a device that tracks network traffic for unusual activities and issues warnings when such an activity is detected. The application scans a network or a device for malicious behavior or violations of policies. Either an administrator or the device centrally collected using security information and event management (SIEM) is usually informed of any

malicious enterprise or violations. SIEM incorporates several sources of alarm performance and uses techniques for filtering alarms to distinguish between malicious activities and false warnings.

While intrusion detection systems track potentially malicious activity networks, they are often able to make false alarms. Therefore, businesses must change their IDS products when they are installed first. It means that the intrusion detection systems are properly established to identify the usual network traffic as opposed to malicious behavior.

The intrusion prevention systems also track the system's incoming network packets to search for suspicious activities and send out alert notices at once.

These preventative controls are complemented by intrusion detection systems as the next protection line. An IDS is an instrument which usually monitors behavior in order to identify malicious and suspicious events. An IDS is another separate computer. An IDS, like a smoke alarm, is an indicator that warns you if certain things happen. Figure 7-41 illustrates an IDS model. The figure is based on the Popular Intrusion Detection Paradigm, the four fundamental elements of an intrusion detection system. An IDS gets raw sensor inputs. It saves, analyzes and tests these inputs and takes some control

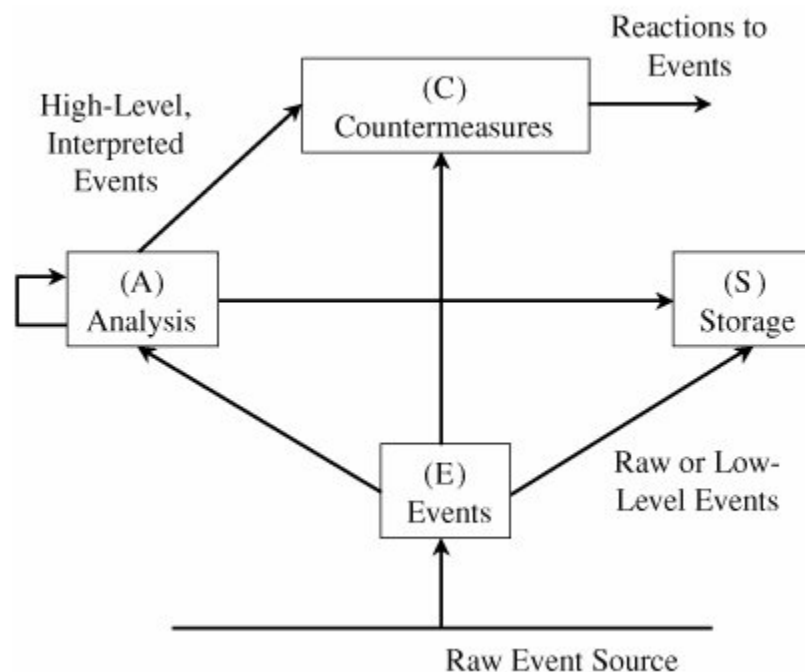


Figure 7.41. Common Components of an Intrusion Detection Framework.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

IDSs perform variety of capabilities:

- monitoring consumers and program activity
- auditing system settings for vulnerabilities and misconfigurations
- evaluating the integrity of important system and documents
- recognizing known harm patterns in technique activity
- identifying abnormal exercise through statistical analysis
- managing audit paths and highlighting person violation of coverage or usual activity
- correcting system settings errors
- putting in and operating traps to report information regarding intruders

Nobody IDS performs many of these functions. Why don't we look more carefully at the forms of IDSs and their used in providing security.

Types of IDSs

Signature-based and heuristic are two common types of intrusion detection systems. Intrusion detection systems based on signature carry out simple pattern matching and reporting situations that corresponds to a pattern of a known type of attack. In the future, the administrator will be able to label the flagging behavior as appropriate so the heuristic IDS will also treat the previously unstated behavior as acceptable. The Heuristic Intrusion Detection Systems, also known as the anomaly, create a model of acceptable conduct and flag exceptions to that model.

Devices for intrusion detection may be network or host based. A network-based IDS is an independent device connected to the Internet monitor traffic throughout that network, and a server-based IDS operates on a single workstation or client or host to secure the traffic of one host.

Signature-Based Intrusion Detection

A simple signature for a known form of attack might identify a series of TCP SYN packets sent successively and often close to each other, like for port scanning. In the first SYN to port 80, for example and then another to port 25, an intrusion detection device will probably find nothing odd. However, because more and more ports, especially ports not open, receive SYN packets, this pattern reflects a potential port scan. Likewise, some protocol stack implementations fail if they receive an ICMP packet that has a data length of 65535 bytes, so this packet is a pattern to view.

The signature detection problem itself is the signatures. An attacker attempts to change a basic attack in such a way that the known signature of the attack does not fit. For example, the attacker may convert lowercase to uppercase letters or convert a 'blank space' symbol into its %20 code. In order to understand that % 20 matches a pattern with a space blank, the IDS must always function from a canonical data stream. A protocol handler stack can discard packets due to malformation, and the attacker will inject the malformed packets that the IDS will deliberately show to cause a pattern mismatch. Each of these variations could be detected by an IDS, but more signatures require additional work for the IDS, which reduces performance.

Naturally, IDSs based on signatures cannot recognize a new attack for whom a signature has still not been installed in the database. Each type of attack begins at some time as a new pattern, and IDS is unable to warn against its presence.

Signatures Intrusion detection systems based on tend to use statistical analysis. This method uses statistical methods both for collecting sample measures of key indicators (e.g. amount of external activity, number of active processes and transaction numbers) and for assessing whether the measurements obtained match the predetermined attack symbols.

Ultimately, signatures should fit every instance of your attack, match understated variations in the attack, however, not match traffic that's not section of an attack. Even so, this goal will be fantastic but unreachable.

Heuristic Intrusion Detection

Since signatures are restricted to some known attack patterns, another type of intrusion detection is important. The heuristic intrusion detection searches for an action that is not typical instead of searching for matches. This original study concentrated upon on individual and sought to find traits that could be useful in knowing normal and abnormal behavior. For instance, a user might start the day often with reading e-mail, writing several documents using a word processor and sometimes storing information. This would be a natural course of action. It does not appear that this user uses many administrator resources. This new activity may provide an indicator that someone else was acting underneath the user's name if he was trying to access critical system management utilities

Thinking of a system that is compromised, it begins clean, with no intrusion, and ends up dirty, absolutely undermined. No trace of usage could be worth changing from clean to dirty; small dirty incidents were more likely, rarely at first, and then increased as the system was more profoundly affected. Any such occurrences could be acceptable on their own, but the accumulation and degree and intensity at which they occurred could have been indicators of an inappropriate occurrence. The system detection inference engine conducts a continuous system analysis, raising an alert when the dirtiness of the system crosses the threshold.

Inference engines work in two different ways. Some, known as state-based intrusion detection systems, could see changes in the system's generalized state or configuration. You can attempt to identify when the device has become unsafe. Others attempt to map current behavior to an appropriate model of activity and raise an alert if the activity is close to the model. These are called model-based intrusion detection systems.. She tried to develop a complex behavioral model, to incorporate changes and transformations over time in a person's actions. The technique compares true operation to a known normality representation.

Intrusion detection can also operate on a known poor activity model. Any other attempt to access a password file is suspicious, for example other than a handful of utilities (login, modify password, create user). This is known as misuse intrusion detection. This work compares the real activity with a known suspicious area.

Any heuristic intrusion detection is categorized into three categories: good/benign, suspicious or unknown. Over time, certain types of actions can switch from one of these to another, which corresponds to the awareness that the IDS may or may not consider certain actions.

Like all pattern matching, the amount of information the system may see (for classifying actions into the correct category), and how well current actions fit into one of those categories, limits heuristic intrusion detection.

Stealth Mode

A network device is an IDS (or, in the case of a host-based IDS, a program running on a network device). Any network device is prone to network attacks. How useful will an IDS be if a Denial-of-Service attack is performed itself? If an attacker managed to log in to a secure network device, will the next step not be to disable IDS?

In order to counter these issues, many IDSs run stealth mode. An IDS includes two network interfaces, one monitoring a network (or a network segment) and one generating alerts and maybe other administrative requirements. The IDS only uses the controlled interface as input; it never sends out packets through it. It also configures the interface such that the device does not have the advertised address through the interface controlled, that is, because that the router does not know that such a device exists, it cannot route anything directly into the address. It's the passive wiretap. When an alarm is needed, the IDS only uses the alarm interface on an individual control network. Figure 7-42 demonstrates such an architecture.

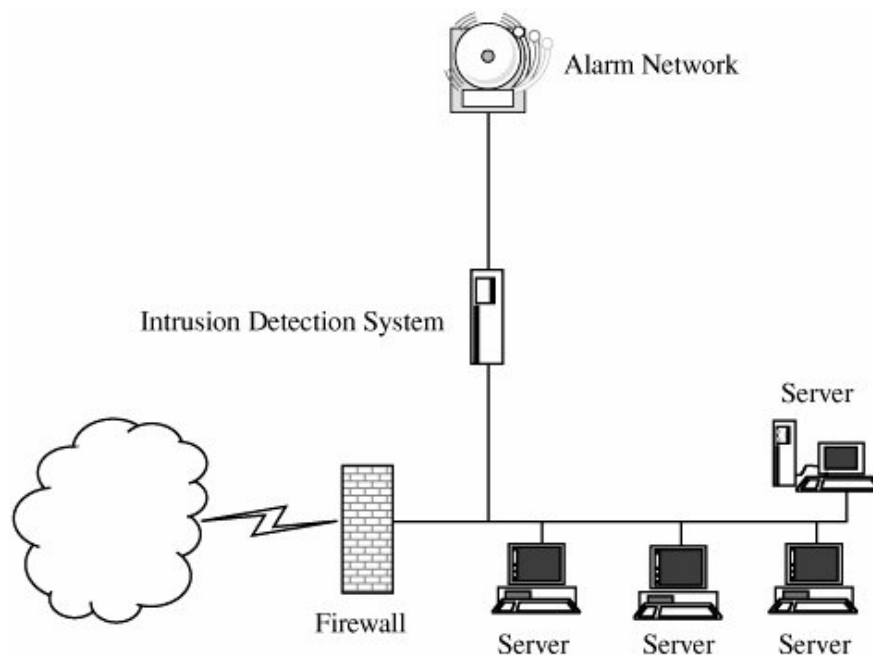


Figure 7.42. Stealth Mode IDS Connected to Two Networks.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Additional IDS Types

Certain engineers often regard other devices as IDSs. For example, programs are able to correlate the newer versions of a software code with a stored version of the digest of the code to identify inappropriate code modification. The tripwire program is the best-known program for comparison software (or static data). On a new system, you run tripwire and create a hash value for every file and then save these hash values to a safe location (offline, so that no intruder can modify them while modifying a system file). If your machine could have been affected later, you run tripwire again and provide it with the saved hash values. Program vulnerability scanners, such as for example ISS Scanning device or Nessus, could be operate against a community. They look for regarded vulnerabilities and survey flaws found. It recalculates the hash value and notes any flaws indicating files which have been updated.

System vulnerability scanners may be performed against the network, such as ISS Scanner or Nessus. They check for identified faults and report observed defects.

Objectives for Intrusion Diagnosis Systems

The two forms of intrusion detection pattern and heuristic approaches, which both have advantages and disadvantages, display different approaches. Current IDS goods also incorporate both methods.

Ideally, an IDS should be easy, clear, precise and complete at the same time. All attacks with poor performance penalties should be detected. The following concept methods may be used by IDS:

- Filtration on packet headers
- Filtration system on packet content
- Maintain network state
- Use organic, multipacket signatures
- Use minimal amount of signatures with utmost effect
- Filter instantly, online
- Hide its presence
- Work with optimal sliding period window size to complement signatures

Responding to Alarms

Whatever form, when a match is identified, an intrusion detection system alerts. The alarm can vary from something simple to something important, such as making a note in an audit log, like paying the system security administrator. Determining what behavior the system can take to what events can be decided by individual implementations.

What are the responses possible? The scope is infinite and the administrator can imagine everything . The responses usually fall into three main categories (which can either be included in one response):

- Monitor, collect info, perhaps increase level of data collected

- Protect, act to lessen exposure
- Contact a human

Monitoring is ideal for a small (initial) attack. Perhaps the true purpose is to track the attacker to see what resources are available or what attempts are being made to attack. Furthermore, all traffic from any given source can be registered for further study. The intruder should find this approach transparent. Attempting to protect can lead to increased access controls and even to a lack of resources (for example, shutting off a network connection or making a file unavailable). The system even can disrupt the attacker's network connection. Defense may be rather clear to the intruder, as opposed to monitoring. The IDS can act instantly in the defensive and also produce a warning for a person who can react for seconds, minutes or longer.

False Results

Intrusion detection systems are not flawless and the main challenge is mistakes. Although an IDS may often accurately identify an attacker, they may struggle by alarming something which is not indeed an attack (referred to as a false positive or type I error in a statistics community) or by not alarming a real attack (a false negative, or type II error). Many such false positive results will make the administrator less trusting in the alerts of IDS, which can lead to the ignorance of real alarm. However, false negative results indicate that actual attacks without intervention move through IDS. We claim that it is the system's sensitivity to the level of false positives and false negatives. The administrator uses most IDS implementations to balance the sensitivity of the system, to find an acceptable balance between false positive and negative ones.

IDS Advantages and Limitations

On the other hand, a rising number of major issues are being identified by IDSs. We may also incorporate their signatures to the IDS model when we find out more about issues. Thus, IDSs are constantly improving over time. They get cheaper and easier to handle at the same time.

To the detriment of preventing IDS, successful attackers have first priority. A badly defended IDS is useless. Thankfully, stealth mode IDS is hard to find, not to mention compromising, even in an internal network.

IDSs search for perceived vulnerabilities, be they known attack patterns or typical behavior models. The same vulnerabilities may occur in similar IDSs and similar attacks can lack their selection criterion. A crucial piece of intelligence within the attacker group is to know how to evade a specific model of IDS. Of course they are trying to correct a deficiency because producers are aware of their goods. Luckily the commercial IDSs classify attacks very well.

The sensitivity of another IDS restriction is difficult to calculate and modify. The IDSs are never fine, so it is important to find the right balance.

A absolute limit is not an IDS per se, but one of use. An IDS is not running itself; someone must monitor their records and respond to their alarms. An administrator is incomprehensible in purchasing and installing an IDS and then ignore it.

Overall, IDSs contribute to network security excellently. Firewalls block access to certain ports or addresses, and often restrict the effect of certain protocols. But firewalls must, by definition, allow such traffic to reach a protected area. It is the work of IDS to see what this traffic actually does inside the protected area.

7.6. Secure E-Mail

The last review we take into account in detail is a secure email. Consider how much email is used and how much it depends on its accuracy. How would you respond if your teacher told you that you were excused for any additional tasks as you have performed that well in your course so far? What if the message was a classmate joke? For sensitive and essential communications we rely on email confidentiality and integrity, even if ordinary email is almost nonconfidential or comprehensive. This segment explores how the privacy and integrity of ordinary e-mails can be added.

Security for E-mail

E-mail is important in today's business and is often comfortable for ordinary individuals to communicate. But, as we have pointed out previously, email is very open, accessible to the recipient at every point from the sender's workstation. You should also acknowledge your e-mail is open and visible for reading to others. Sometimes we wish e-mail to become more secure.

We need e-mails to be secure occasionally. We start by analyzing the exposures of daily and E-mail to evaluate and enforce a more secure form.

Threats to E-mail

Consider threats to electronic mail:

- message interception (confidentiality)
- message interception (blocked delivery)
- message interception and subsequent replay
- message content modification
- message origin modification
- message content forgery by outsider
- message origin forgery by outsider
- message content forgery by recipient
- message origin forgery by recipient
- denial of message transmission

Confidentiality and content falsification are often processed through encryption. Encryption will also aim to safeguard against replay, but we also have to use a protocol to include a specific encrypted text in every document. Symmetric encryption by a recipient cannot protect itself from fraud since both sender and recipient share a similar key, but public key systems can allow a recipient to decrypt but not to encrypt. In general, senders or recipients cannot defend them against a blocked delivery because of the lack of control over the midpoints of the network.

Prerequisites and Solutions

If we mention the security e-mail requirements, the following security will be included in our wish list.

- *message confidentiality* (the message is not exposed en route to the receiver)
- *message integrity* (what the receiver sees is what was sent)
- *sender authenticity* (the receiver is confident who the sender was)
- *Nonrepudiation* (the sender cannot deny having sent the message)

Not every message needs all have these qualities, but an ideal secure email package could selectively invoke these capacities.

Designs

The Internet Society has developed the standard for encrypted emails through its task forces on architecture (IAB) and research (IRTF) and engineering (IETF). In documents 1421, 1422, 1423 and 1424 encrypted email protocols are recorded as an Internet standard. This norm is in reality the third refinement of the original specification.

One of the design goals for encrypted email was to allow enhanced security messages to pass via the existing Internet email networks as normal messages. This needs that there is no improvement in the protection of the broad current e-mail network. Therefore, all security takes place in a message body.

Confidentiality

Due to the fact that the protection has many facets, we will first look at how to enhance confidentiality. The sender selects a symmetric (random) encryption key. The sender then encrypted the whole message, including the headers, TO: SUBJECT: and DATE: The sender will then prepend headers of plaintext. The sender encrypts and attaches the message under the public key of the recipient for key management. Figure 7-43 shows the mechanism by which an encrypted message is produced.

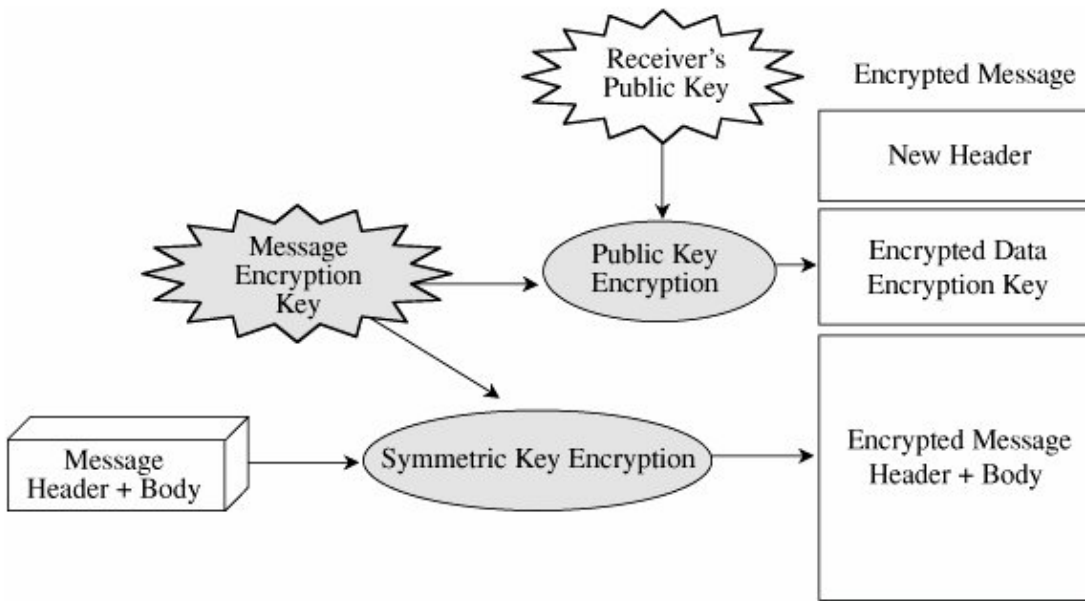


Figure 7-43. Overview of Encrypted E-mail Processing.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Encryption can offer output to any string. Many email managers expect the traffic of the message to be no more than the usual printable characters. Network e-mail administrators use imprinted characters in the traffic stream as control signals. Encoded emails transform the entire cipher text message into printable characters to prevent transmission problems. Figure 7-44 gives an example of an encrypted message. Note the three sections: an external (plaintext) header, a section that allows you to transfer the message encryption key and the message encryption itself. (Shading is

shown

as

encryption.)

```
Header {
  From: noname@net.net
  Received: from post.isp.net by mail.pfleeger.com
    with SMTP id AA9439; Tue, 22 June 15:14:06 GMT
  Date: Tue, 22 Jun 04:12:25 EDT
  From: Anonymous <noname@net.net>
  To: pfleeger@tisuk.co.uk
  Subject: Book example
}

Encrypted Message Key {
  Proc-Type: 4, ENCRYPTED
  DEK-Info: DES-CBC, EE2516152EF97DA2...
  Key-Info: RSA, 1Bk9ac...
}

Encrypted Body {
  From: president@whitehouse.gov
  Received: from post.isp.net by mail.pfleeger.com
    with SMTP id AA9439; Tue, 22 June 15:14:06 GMT
  Date: Tue, 22 Jun 04:12:25 EDT
  From: The President <president@whitehouse.gov>
  To: pfleeger@pfleeger.com
  Subject: Book example

  Hope this works as a convincing example of
  encrypted e-mail for your book.
  We all need security.

  Cheers!

  --The Prez
}
```

Encrypted with Recipient's Public Key

Encrypted with (Random) Message Key

Figure 7.44. Encrypted E-mailSecured Message.

Above Image taken from the book Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

The encrypted standard for e-mail performs as stated more accurately using both symmetric and asymmetric encryption. For symmetric encryption only, the standard is described: the sender and receiver must first have a mutual secret encryption key to use symmetric encryption. The processing type (the "Proc-Type") indicates what services have been implemented to improve privacy. The type of exchange key (symmetric or asymmetrical) will be shown in the data interchange key field ("DEK-Info"). The key exchange (the "Key-Info" field), which is encrypted under this shared encryption key, includes the message encryption key. If asymmetric encryption is used by the key exchange technique, the key exchange field will be encrypted under the public key of the receiver. The sender's certificate may also be included (used for determining authenticity and for generating replies).

The encrypted e-mail standard supports a variety of encryption algorithms, using famous message-related algorithms such as DES, triple DES, AES, and RSA and DiffieHellman.

Other Stability Features

Besides confidentiality, for secure e-mail, we also want different types of integrity.

The digital signature is still present in encrypted e-mails so the sender is authentic and non-repudiable. Integrity is also guaranteed due to a hash function in the digital signature (called a message integrity check or MIC). Encrypted emails for confidentiality can optionally be encrypted.

Notice that the header inside this message is different in Figure 7-44 (in the encrypted portion). In the encrypted part, the identity of a sender or the actual subject of a message may be hidden.

Email processing encrypted in Figure 7-45 can be combined with standard email packages so that an individual can send enhanced and non-enhanced messages. If the sender wishes to add enhancements, the receiver must also uninstall the enhancements by invoking an extra bit of encrypted e-mail processing at the end of the sender. But messages flow as usual through the mail managers without change.

S/MIME can handle an exchange of speech, graphic, video, and other complex message sections other than text messages.

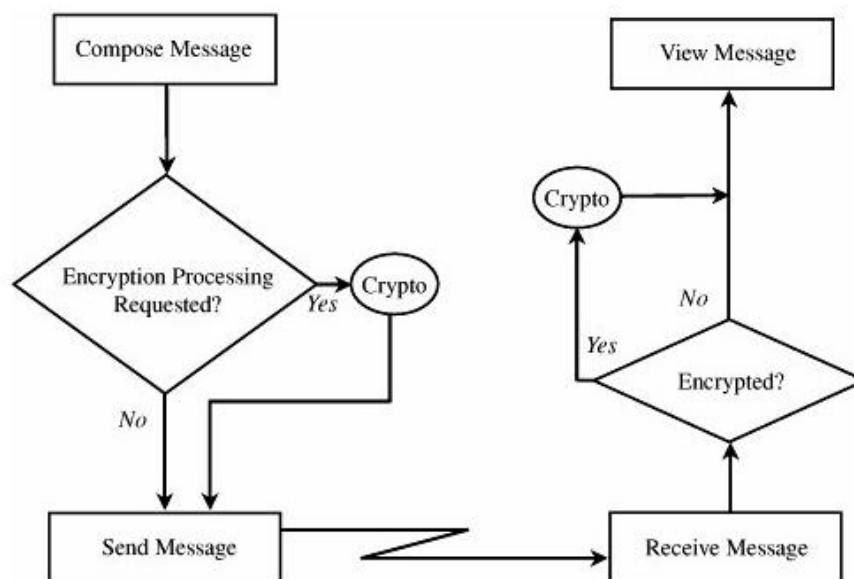


Figure 7.45. Encrypted E-mail Processing in Message Transmission.

Above Image taken from the book *Security in Computing, Fourth Edition* By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger

Encryption for Secure E-mail

Key management is the main challenge with encrypted email. The certificate system is excellent for key sharing and authentication with a public key. The difficulty of building a hierarchy is with certificates. Many organisations are organized hierarchically. The encrypted e-mail problem moves to an interorganizational hierarchy beyond one organisation. PGP has been built as a simplified type of encrypted e-mail, precisely because of the issue of enforcing hierarchy on a non-hierarchical environment.

Encrypted e-mail offers good end-to-end e-mail encryption. Triple DES, AES, and RSA are very strong, in particular if RSA is used with a lengthy key (1024 bits or more). The remaining vulnerabilities of encrypted e-mail are not protected by the following issues: endpoints. If a sender or receiver has access to an intruder, it may circumvent the device, change the code that improves the privacy, or leak a cryptographic key.

PGP

PGP is Pretty Good Privacy. Phil Zimmerman invented it in 1991. Originally a free product, it was purchased by Network Associates in 1996 and became a commercial product. There is also a freeware version. PGP is available extensively in both commercial and freeware versions and is used heavily by private e-mail exchangers.

PGP solves the main issue of delivery for what is known as the 'ring of trust' or a 'key ring' user. One person gives a public key to another person or the other user retrieves the first public key from a server. Few people have public PGP keys at the bottom of e-mails. And a second person may give a third key (and a fourth, and so on).

A policy to create trust is not required by PGP. Instead, each user will determine the amount of trust per key received.

Any or more of the following steps are carried out by the PGP processing, depending on whether confidentiality, integrity, authenticity or a combination is selected:

- Create a random session key for a symmetric algorithm.
- Encrypt the message, using the session key (for message confidentiality).
- Encrypt the session key under the recipient's public key.
- Generate a message digest or hash of the message; sign the hash by encrypting it with the sender's private key (for message integrity and authenticity).
- Attach the encrypted session key to the encrypted message and digest.
- Transmit the message to the recipient.

The recipient reverses these steps to retrieve and validate the message content.

S/MIME

An Internet standard defines how e-mails are sent and received. The MIME general specification specifies the e-mail attachment format and handling. The Internet Standard for secure e-mail attachments is S/MIME (Secure Multipurpose Internet Mail Extensions).

The PGP and its predecessors, PEM (Privacy-Enhanced Mail) and RIPEM are very much like S/MIME. The S/MIME packages, such as Eudora and Microsoft Outlook have been introduced for commercial email.

The main distinction between S/MIME and PGP is the key exchange method. Core PGP depends on each user's exchanging keys and the establishment of a ring of trusting recipients, and on a degree of trust in the validity of the keys for those recipients. S/MIME uses certificates, generally presented in X.509 format, which have been hereby validated for key exchange. With S/MIME, senders and recipients need therefore not exchange keys beforehand as long as they both trust a standard certifier.

S/MIME is performing very similar security transformations as PGP. Initially PGP was for plaintext, but S/MIME handles, for example, data files, and all sorts of attachments (secure) (for example, spreadsheets, graphics, presentations, movies, and sound). Since it is embedded in many commercial email packets, the secure email market will possibly dominate the S/MIME.

7.7 Review Questions

1. Describe a social engineering attack you could use to obtain a user's password.
2. Is a social engineering attack more likely to succeed in person, over the telephone, or through e-mail? Justify your answer
3. A port scanner is a tool useful to an attacker to identify possible vulnerabilities in a potential victim's system. Cite a situation in which someone who is not an attacker could use a port scanner for a nonmalicious purpose.
4. Compare copper wire, microwave, optical fiber, infrared, and (radio frequency) wireless in their resistance to passive and active wiretapping.
5. What is a "man in the middle" attack? Cite a real-life example (not from computer networking) of such an attack. Suggest a means by which sender and receiver can preclude a man-in-the-middle attack. (a) Cite a means not requiring cryptography. (b) Cite a means involving cryptography but also ensuring that the man in the middle cannot get in the middle of the key exchange
6. Signing of mobile code is a suggested approach for addressing the vulnerability of hostile code. Outline what a code-signing scheme would have to do.
7. Does a VPN use link encryption or end-to-end? Justify your answer.
8. Why is a firewall a good place to implement a VPN? Why not implement it at the actual server(s) being accessed?
9. Can encrypted e-mail provide verification to a sender that a recipient has read an e-mail message? Why or why not?
10. Can message confidentiality and message integrity protection be applied to the same message? Why or why not?
11. What are the advantages and disadvantages of an e-mail program (such as Eudora or Outlook) that automatically applies and removes protection to e-mail messages between sender and receiver?

7.8 References

1. Security in Computing, Fourth Edition By Charles P. Pfleeger - Pfleeger Consulting Group, Shari Lawrence Pfleeger - RAND Corporation Publisher: Prentice Hall

2. Cryptography and Network Security - Principles and Practice fifth edition Stallings William
Publisher: Pearson

3. Cryptography And Network Security 3rd Edition behrouz a forouzan and debdeep mukhopadhyay 3/E Publisher: McGraw Hill Education

4. Cryptography and Network Security, 3e Atul Kahate Publisher: McGraw Hill

IDOL

DATABASE SECURITY - I

Unit structure:

8.0 Objectives

8.1 Introduction

8.2 Introduction to Databases

8.2.1. Concept of a Database

8.2.2. Components of Databases

8.2.3. Queries

8.2.4. Advantages of Using Databases

8.3 Security Requirements

8.3.1. Integrity of the Database

8.3.2. Element Integrity

8.3.3. Auditability

8.3.4. Access Control

8.3.5. User Authentication

8.3.6. Availability

8.3.7. Integrity/Confidentiality/Availability

8.4 Reliability and Integrity

8.4.1. Protection Features from the Operating System

8.4.2. Two-Phase Update

8.4.2.1. Update Technique

8.4.2.2. Two-Phase Update Example

8.4.3. Redundancy/Internal Consistency

8.4.3.1. Error Detection and Correction Codes

8.4.3.2. Shadow Fields

8.4.4. Recovery

8.4.5. Concurrency/Consistency

8.4.6. Monitors

8.4.6.1. Range Comparisons

8.4.6.2. State Constraints

8.4.6.3. Transition Constraints

8.5 Sensitive Data

8.5.1. Access Decisions

8.5.1.1. Availability of Data

8.5.1.2. Acceptability of Access

8.5.1.3. Assurance of Authenticity

8.5.2. Types of Disclosures

8.5.2.1. Exact Data

8.5.2.2. Bounds

8.5.2.3. Negative Result

8.5.2.4. Existence

8.5.2.5. Probable Value

8.5.2.6. Summary of Partial Disclosure

8.5.3. Security versus Precision

8.6 Summary

8.7 Review Questions

8.8 Bibliography, References and Further Reading

8.0 OBJECTIVES

We begin this chapter with a brief summary of database terminology. Then we consider the security requirements for database management systems. Two major security problems - integrity and secrecy, are explained in a database context.

8.1 INTRODUCTION

Protecting data is at the heart of many secure systems, and many users (people, programs, or systems) rely on a database management system (DBMS) to manage the protection. There is substantial current interest in DBMS security because databases are newer than programming and operating systems. Databases are essential to many business and government organizations, holding data that reflect the organization's core competencies. Often, when business processes are reengineered to make them more effective and more in tune with new or revised goals, one of the first systems to receive scrutiny is the set of databases supporting the business processes. Thus, databases are more than software-related repositories. Their organization and contents are considered

valuable corporate assets that must be carefully protected. However, the protection provided by database management systems has had mixed results. Over time, we have improved our understanding of database security problems, and several good controls have been developed. But, as you will see, there are still more security concerns for which there are no available controls.

8.2 INTRODUCTION TO DATABASES

We begin by describing a database and defining terminology related to its use. We draw on examples from what is called the relational database because it is one of the most widely used types. However, all the concepts described here apply to any type of database. We first define the basic concepts and then use them to discuss security concerns.

8.2.1 Concept of a Database:

A database is a collection of *data* and a set of *rules* that organize the data by specifying certain relationships among the data. Through these rules, the user describes a *logical* format for the data. The data items are stored in a file, but the precise *physical* format of the file is of no concern to the user. A database administrator is a person who defines the rules that organize the data and controls who should have access to what parts of the data. The user interacts with the database through a program called a database manager or a database management system (DBMS), informally known as a front end.

8.2.2 Components of Databases:

The database file consists of records, each of which contains one related group of data. As shown in the example in Table 8-1, a record in a name and address file consists of one name and address. Each record contains fields or elements, the elementary data items themselves. The fields in the name and address record are NAME, ADDRESS, CITY, STATE, and ZIP (where ZIP is the U.S. postal code). This database can be viewed as a two-dimensional table, where a record is a row and each field of a record is an element of the table.

ADAMS	212 Market St.	Columbus	OH	43210
BENCHLY	501 Union St.	Chicago	IL	60603
CARTER	411 Elm St.	Columbus	OH	43210

Table 8-1 Example of a database

Not every database is easily represented as a single, compact

table. The database in Figure 8-1 logically consists of three files with possibly different uses. These three files could be represented as one large table, but that depiction may not improve the utility of or access to the data.

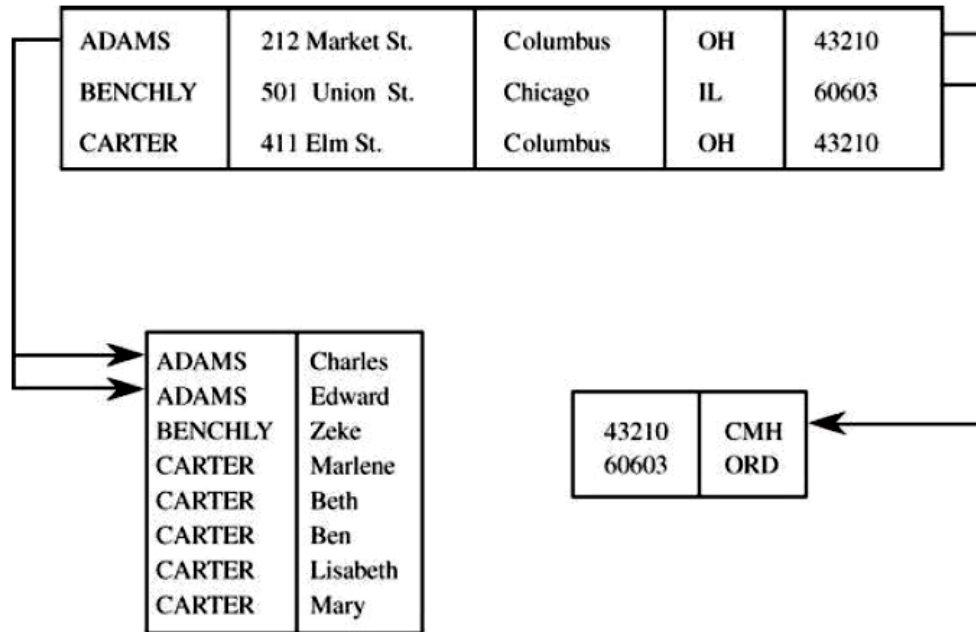


Figure 8-1 Database of Several Related Tables

The logical structure of a database is called a schema. A user may have access to only part of the database, called a subschema. The overall schema of the database in Figure 8-1 is detailed in Table 8-2. The three separate blocks of the figure are examples of subschemas, although other subschemas of this database can be defined. We can use schemas and subschemas to present to users only those elements they wish or need to see. For example, if Table 8-1 represents the employees at a company, the subschema on the lower left can list employee names without revealing personal information such as home address.

Name	First	Address	City	State	Zip	Airport
ADAMS	Charles	212 Market St.	Columbus	OH	43210	CMH
ADAMS	Edward	212 Market St.	Columbus	OH	43210	CMH
BENCHLY	Zeke	501 Union St.	Chicago	IL	60603	ORD
CARTER	Marlene	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Beth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Ben	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Lisabeth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Mary	411 Elm St.	Columbus	OH	43210	CMH

Table 8-2 Schema of Database from Figure 8-1

The rules of a database identify the columns with names. The name of each column is called an attribute of the database. A relation is a set of columns. For example, using the database in Table 8-2, we see that NAME-ZIP is a relation formed by taking the NAME and ZIP columns, as shown in Table 8-3. The relation specifies clusters of related data values in much the same way that the relation "mother of" specifies a relationship among pairs of humans. In this example, each cluster contains a pair of elements, a NAME and a ZIP. Other relations can have more columns, so each cluster may be a triple, a 4-tuple, or an n -tuple (for some value n) of elements.

Name	Zip
ADAMS	43210
BENCHLY	60603
CARTER	43210

Table 8-3 Relation in a Database

8.2.3 Queries:

Users interact with database managers through commands to the DBMS that retrieve, modify, add, or delete fields and records of the database. A command is called a query. Database management systems have precise rules of syntax for queries. Most query languages use an English-like notation, and many are based on SQL, a structured query language originally developed by IBM. We have written the example queries in this chapter to resemble English

sentences so that they are easy to understand. For example, the query

```
SELECT NAME = 'ADAMS'
```

retrieves all records having the value *ADAMS* in the *NAME* field. The result of executing a query is a subschema. One way to form a subschema of a database is by selecting records meeting certain conditions. For example, we might select records in which *ZIP*=43210, producing the result shown in Table 8-4.

Name	First	Address	City	State	Zip	Airport
ADAMS	Charles	212 Market St.	Columbus	OH	43210	CMH
ADAMS	Edward	212 Market St.	Columbus	OH	43210	CMH
CARTER	Marlene	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Beth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Ben	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Lisabeth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Mary	411 Elm St.	Columbus	OH	43210	CMH

Table 8-4 Results of a SELECT Query

8.2.4 Advantages of Using Databases:

The logical idea behind a database is this: A database is a single collection of data, stored and maintained at one central location, to which many people have access as needed. However, the actual implementation may involve some other physical storage arrangement or access. The essence of a good database is that the users are unaware of the physical arrangements; the unified logical arrangement is all they see. As a result, a database offers many advantages over a simple file system:

- **Shared access:** so that many users can use one common, centralized set of data
- **Minimal redundancy:** so that individual users do not have to collect and maintain their own sets of data
- **Data consistency:** so that a change to a data value affects all users of the data value
- **Data integrity:** so that data values are protected against accidental or malicious undesirable changes

- **Controlled access:** so that only authorized users can view or to modify data values

A DBMS is designed to provide these advantages efficiently. However, as often happens, the objectives can conflict with each other. Security interests can conflict with performance. This clash is not surprising because measures taken to enforce security often increase the computing system's size or complexity. What is surprising, though, is that security interests may also reduce the system's ability to provide data to users by limiting certain queries that would otherwise seem innocuous.

8.3 SECURITY REQUIREMENTS

The basic security requirements of database systems are not unlike those of other computing systems we have studied about in the previous chapters. The basic problems are access control, exclusion of spurious data, authentication of users, and reliability which have been discussed in earlier chapters are also valid for database systems, along with the following list of requirements for database security.

- **Physical database integrity:** The data of a database are immune to physical problems, such as power failures, and someone can reconstruct the database if it is destroyed through a catastrophe.
- **Logical database integrity:** The structure of the database is preserved. With logical integrity of a database, a modification to the value of one field does not affect other fields, for example.
- **Element integrity:** The data contained in each element are accurate.
- **Auditability:** It is possible to track who or what has accessed (or modified) the elements in the database.
- **Access control:** A user can access only authorized data, and different users can be restricted to different modes of access (such as read or write).
- **User authentication:** Every user is positively identified, both for the audit trail and for permission to access certain data.
- **Availability:** Users can access the database in general and all the data for which they are authorized.

8.3.1 Integrity of the Database:

If a database is to serve as a central repository of data, users must be able to trust the accuracy of the data values. This condition implies that the database administrator must be assured that updates are performed only by authorized individuals. It also implies that the data must be protected from corruption, either by an outside illegal program action or by an outside force such as fire or a power failure. Two situations can affect the integrity of a database: when the whole database is damaged (as happens, for example, if its storage medium is damaged) or when individual data items are unreadable. Integrity of the database is the responsibility of the DBMS, the operating system, and the (human) computing system manager. From the perspective of the operating system and the computing system manager, databases and DBMSs are files and programs, respectively. Therefore, one way of protecting the database is to regularly back up all files on the system. These periodic backups can be adequate controls against catastrophic failure. Sometimes it is important to be able to reconstruct the database at the point of a failure. For instance, when the power fails suddenly, a bank's clients may be in the middle of making transactions or students may be during registering online for their classes. In these cases, we want to be able to restore the systems to a stable point without forcing users to redo their recently completed transactions. To handle these situations, the DBMS must maintain a log of transactions. For example, suppose the banking system is designed so that a message is generated in a log (electronic or paper or both) each time a transaction is processed. In the event of a system failure, the system can obtain accurate account balances by reverting to a backup copy of the database and reprocessing all later transactions from the log.

8.3.2 Element Integrity:

The integrity of database elements is their correctness or accuracy. Ultimately, authorized users are responsible for entering correct data into databases. However, users and programs make mistakes collecting data, computing results, and entering values. Therefore, DBMSs sometimes take special action to help catch errors as they are made and to correct errors after they are inserted.

This corrective action can be taken in three ways. First, the DBMS can apply field checks, activities that test for appropriate values in a position. A field might be required to be numeric, an uppercase letter, or one of a set of acceptable characters. The check ensures that a value falls within specified bounds or is not greater than the sum of the values in two other fields. These checks prevent simple errors as the data are entered.

A second integrity action is provided by access control. To see why, consider life without databases. Data files may contain data

from several sources, and redundant data may be stored in several different places. For example, a student's home address may be stored in many different campus files: at class registration, for dining hall privileges, at the bookstore, and in the financial aid office. Indeed, the student may not even be aware that each separate office has the address on file. If the student moves from one residence to another, each of the separate files requires correction. Without a database, there are several risks to the data's integrity. First, at a given time, there could be some data files with the old address (they have not yet been updated) and some simultaneously with the new address (they have already been updated). Second, there is always the possibility that the data fields were changed incorrectly, again leading to files with incorrect information. Third, there may be files of which the student is unaware, so he or she does not know to notify the file owner about updating the address information. These problems are solved by databases. They enable collection and control of this data at one central source, ensuring the student and users of having the correct address. However, the centralization is easier said than done. Who owns this shared central file? Who has authorization to update which elements? What if two people apply conflicting modifications? What if modifications are applied out of sequence? How are duplicate records detected? What action is taken when duplicates are found? These are policy questions that must be resolved by the database administrator by having formal processes which are needed for managing changes in databases.

The third means of providing database integrity is maintaining a change log for the database. A change log lists every change made to the database; it contains both original and modified values. Using this log, a database administrator can undo any changes that were made in error. For example, a library fine might erroneously be posted against Charles W. Robertson, instead of Charles M. Robertson, flagging Charles W. Robertson as ineligible to participate in varsity athletics. Upon discovering this error, the database administrator obtains Charles W's original eligibility value from the log and corrects the database.

8.3.3 Auditability:

For some applications it may be desirable to generate an audit record of all access (read or write) to a database. Such a record can help to maintain the database's integrity, or at least to discover after the fact who had affected which values and when. A second advantage is that users can access protected data incrementally; that is, no single access reveals protected data, but a set of sequential accesses viewed together reveals the data, much like discovering the clues in a detective novel. In this case, an audit trail can identify which clues a user has already been given, as a guide to whether to tell the user more. As we noted earlier, granularity

becomes an impediment in auditing. Audited events in operating systems are actions like *open file* or *call procedure*; they are seldom as specific as *write record 3* or *execute instruction 1*. To be useful for maintaining integrity, database audit trails should include accesses at the record, field, and even element levels. This detail is prohibitive for most database applications. Furthermore, it is possible for a record to be accessed but not reported to a user, as when the user performs a select operation. (Accessing a record or an element without transferring to the user the data received is called the pass-through problem.) Also, you can determine the values of some elements without accessing them directly. (For example, you can ask for the average salary in a group of employees when you know the number of employees in the group is only one.) Thus, a log of all records accessed directly may both overstate and understate what a user knows.

8.3.4 Access Control:

Databases are often separated logically by user access privileges. For example, all users can be granted access to general data, but only the personnel department can obtain salary data and only the marketing department can obtain sales data. Databases are very useful because they centralize the storage and maintenance of data. Limited access is both a responsibility and a benefit of this centralization. The database administrator specifies who should be allowed access to which data, at the view, relation, field, record, or even element level. The DBMS must enforce this policy, granting access to all specified data or no access where prohibited. Furthermore, the number of modes of access can be many. A user or program may have the right to read, change, delete, or append to a value, add or delete entire fields or records, or reorganize the entire database. Superficially, access control for a database seems like access control for operating systems or any other component of a computing system. However, the database problem is more complicated. Operating system objects, such as files, are unrelated items, whereas records, fields, and elements are related. Although a user cannot determine the contents of one file by reading others, a user might be able to determine one data element just by reading others. The problem of obtaining data values from others is called inference, and we consider it in depth later in this chapter. It is important to notice that you can access data by inference without needing direct access to the secure object itself. Restricting inference may mean prohibiting certain paths to prevent possible inferences. However, restricting access to control inference also limits queries from users who do not intend unauthorized access to values. Moreover, attempts to check requested accesses for possible unacceptable inferences may degrade the DBMS's performance. Finally, size or granularity is different between operating system objects and database objects. An access control

list of several hundred files is much easier to implement than an access control list for a database with several hundred files of perhaps a hundred fields each. Size affects the efficiency of processing.

8.3.5 User Authentication:

The DBMS can require rigorous user authentication. For example, a DBMS might insist that a user pass both specific password and time-of-day checks. This authentication supplements the authentication performed by the operating system. Typically, the DBMS runs as an application program on top of the operating system. This system design means that there is no trusted path from the DBMS to the operating system, so the DBMS must be suspicious of any data it receives, including user authentication. Thus, the DBMS is forced to do its own authentication.

8.3.6 Availability:

A DBMS has aspects of both a program and a system. It is a program that uses other hardware and software resources, yet to many users it is the only application run. Users often take the DBMS for granted, employing it as an essential tool with which to perform tasks. But when the system is not available, busy serving other users or down to be repaired or upgraded, the users are very aware of a DBMS's unavailability. For example, two users may request the same record, and the DBMS must arbitrate; one user is bound to be denied access for a while. Or the DBMS may withhold unprotected data to avoid revealing protected data, leaving the requesting user unhappy. Problems like these result in high availability requirements for a DBMS.

8.3.7 Integrity/Confidentiality/Availability:

The three aspects of computer security - integrity, confidentiality, and availability, clearly relate to database management systems. As we have described, integrity applies to the individual elements of a database as well as to the database. Thus, integrity is a major concern in the design of database management systems. We look more closely at integrity issues in the next section. Confidentiality is a key issue with databases because of the inference problem, whereby a user can access sensitive data indirectly. Inference and access control are covered later in this chapter. Finally, availability is important because of the shared access motivation underlying database development. However, availability conflicts with confidentiality. The last sections of the chapter address availability in an environment in which confidentiality is also important.

8.4 RELIABILITY AND INTEGRITY

Databases amalgamate data from many sources, and users expect a DBMS to provide access to the data in a reliable way. When software engineers say that software has reliability, they mean that the software runs for very long periods of time without failing. Users certainly expect a DBMS to be reliable, since the data usually are key to business or organizational needs. Moreover, users entrust their data to a DBMS and rightly expect it to protect the data from loss or damage. Concerns for reliability and integrity are general security issues, but they are more apparent with databases. A DBMS guards against loss or damage in several ways. However, the controls we consider are not absolute: No control can prevent an authorized user from inadvertently entering an acceptable but incorrect value. Database concerns about reliability and integrity can be viewed from three dimensions:

Database integrity: *This is concerned that the database is protected against damage, as from the failure of a disk drive or the corruption of the master database index. These concerns are addressed by operating system integrity controls and recovery procedures.*

Element integrity: *This is concerned that the value of a specific data element is written or changed only by authorized users. Proper access controls protect a database from corruption by unauthorized users.*

Element accuracy: *This is concerned that only correct values are written into the elements of a database. Checks on the values of elements can help prevent insertion of improper values. Also, constraint conditions can detect incorrect values.*

8.4.1 Protection Features from the Operating System:

In an earlier chapter, we discussed the protection an operating system provides for its users. A responsible system administrator backs up the files of a database periodically along with other user files. The files are protected during normal execution against outside access by the operating system's standard access control facilities. Finally, the operating system performs certain integrity checks for all data as a part of normal read and write operations for I/O devices. These controls provide basic security for databases, but the database manager must enhance them.

8.4.2 Two-Phase Update:

A serious problem for a database manager is the failure of the computing system in the middle of modifying data. If the data item to

be modified was a long field, half of the field might show the new value, while the other half would contain the old. Even if errors of this type were spotted easily (which they are not), a more subtle problem occurs when several fields are updated, and no single field appears to be in obvious error. The solution to this problem, proposed first by Lampson and Sturgis and adopted by most DBMSs, uses a two-phase update.

8.4.2.1 Update Technique:

During the first phase, called the intent phase, the DBMS gathers the resources it needs to perform the update. It may gather data, create dummy records, open files, lock out other users, and calculate final answers; in short, it does everything to prepare for the update, but it makes no changes to the database. The first phase is repeatable an unlimited number of times because it takes no permanent action. If the system fails during execution of the first phase, no harm is done because all these steps can be restarted and repeated after the system resumes processing. The last event of the first phase, called committing, involves the writing of a commit flag to the database. The commit flag means that the DBMS has passed the point of no return: After committing, the DBMS begins making permanent changes.

The second phase makes the permanent changes. During the second phase, no actions from before the commit can be repeated, but the update activities of phase two can also be repeated as often as needed. If the system fails during the second phase, the database may contain incomplete data, but the system can repair these data by performing all activities of the second phase. After the second phase has been completed, the database is again complete.

8.4.2.2 Two-Phase Update Example:

Suppose a database contains an inventory of a company's office supplies. The company's central stockroom stores paper, pens, paper clips, and the like, and the different departments requisition items as they need them. The company buys in bulk to obtain the best prices. Each department has a budget for office supplies, so there is a charging mechanism by which the cost of supplies is recovered from the department. Also, the central stockroom monitors quantities of supplies on hand to order new supplies when the stock becomes low.

Suppose the process begins with a requisition from the accounting department for 50 boxes of paper clips. Assume that there are 107 boxes in stock and a new order is placed if the quantity in stock ever falls below 100. Here are the steps followed the stockroom receives the requisition.

1. The stockroom checks the database to determine that 50 boxes

of paper clips are on hand. If not, the requisition is rejected, and the transaction is finished.

2. If enough paper clips are in stock, the stockroom deducts 50 from the inventory figure in the database ($107 - 50 = 57$).
3. The stockroom charges accounting's supplies budget (also in the database) for 50 boxes of paper clips.
4. The stockroom checks its remaining quantity on hand (57) to determine whether the remaining quantity is below the reorder point. Because it is, a notice to order more paper clips is generated, and the item is flagged as "on order" in the database.
5. A delivery order is prepared, enabling 50 boxes of paper clips to be sent to accounting.

All five of these steps must be completed in the order listed for the database to be accurate and for the transaction to be processed correctly.

Suppose a failure occurs while these steps are being processed. If the failure occurs before step 1 is complete, there is no harm because the entire transaction can be restarted. However, during steps 2, 3, and 4, changes are made to elements in the database. If a failure occurs then, the values in the database are inconsistent. Worse, the transaction cannot be reprocessed because a requisition would be deducted twice, or a department would be charged twice, or two delivery orders would be prepared.

When a two-phase commit is used, shadow values are maintained for key data points. A shadow data value is computed and stored locally during the intent phase, and it is copied to the actual database during the commit phase. The operations on the database would be performed as follows for a two-phase commit.

Intent:

1. Check the value of COMMIT-FLAG in the database. If it is set, this phase cannot be performed. Halt or loop, checking COMMIT-FLAG until it is not set.
2. Compare number of boxes of paper clips on hand to number requisitioned; if more are requisitioned than are on hand, halt.
3. Compute $TCLIPS = ONHAND - REQUISITION$.
4. Obtain BUDGET, the current supplies budget remaining for accounting department. Compute $TBUDGET = BUDGET - COST$, where COST is the cost of 50 boxes of clips.

5. Check whether TCLIPS is below reorder point; if so, set TREORDER = TRUE; else set TREORDER = FALSE.

Commit:

1. Set COMMIT-FLAG in database.
2. Copy TCLIPS to CLIPS in database.
3. Copy TBUDGET to BUDGET in database.
4. Copy TREORDER to REORDER in database.
5. Prepare notice to deliver paper clips to accounting department. Indicate transaction completed in log.
6. Unset COMMIT-FLAG.

With this example, each step of the intent phase depends only on unmodified values from the database and the previous results of the intent phase. Each variable beginning with T is a shadow variable used only in this transaction. The steps of the intent phase can be repeated an unlimited number of times without affecting the integrity of the database.

Once the DBMS begins the commit phase, it writes a commit flag. When this flag is set, the DBMS will not perform any steps of the intent phase. Intent steps cannot be performed after committing because database values are modified in the commit phase. Notice, however, that the steps of the commit phase can be repeated an unlimited number of times, again with no negative effect on the correctness of the values in the database.

The one remaining flaw in this logic occurs if the system fails after writing the "transaction complete" message in the log but before clearing the commit flag in the database. It is a simple matter to work backward through the transaction log to find completed transactions for which the commit flag is still set and to clear those flags.

8.4.3 Redundancy/Internal Consistency:

Many DBMSs maintain additional information to detect internal inconsistencies in data. The additional information ranges from a few check bits to duplicate or shadow fields, depending on the importance of the data.

8.4.3.1 Error Detection and Correction Codes:

One form of redundancy is error detection and correction

codes, such as parity bits, Hamming codes, and cyclic redundancy checks. These codes can be applied to single fields, records, or the entire database. Each time a data item is placed in the database, the appropriate check codes are computed and stored; each time a data item is retrieved, a similar check code is computed and compared to the stored value. If the values are unequal, they signify to the DBMS that an error has occurred in the database. Some of these codes point out the place of the error; others show precisely what the correct value should be. The more information provided; the more space required to store the codes.

8.4.3.2 Shadow Fields:

Entire attributes or entire records can be duplicated in a database. If the data are irreproducible, this second copy can provide an immediate replacement if an error is detected. Obviously, redundant fields require substantial storage space.

8.4.4 Recovery:

In addition to these error correction processes, a DBMS can maintain a log of user accesses, particularly changes. In the event of a failure, the database is reloaded from a backup copy and all later changes are then applied from the audit log.

8.4.5 Concurrency/Consistency:

Database systems are often multi-user systems. Accesses by two users sharing the same database must be constrained so that neither interferes with the other. Simple locking is done by the DBMS. If two users attempt to read the same data item, there is no conflict because both obtain the same value. If both users try to modify the same data items, we often assume that there is no conflict because each knows what to write; the value to be written does not depend on the previous value of the data item. However, this supposition is not quite accurate.

To see how concurrent modification can get us into trouble, suppose that the database consists of seat reservations for an airline flight. Agent A, booking a seat for passenger Mock, submits a query to find which seats are still available. The agent knows that Mock prefers a right aisle seat, and the agent finds that seats 5D, 11D, and 14D are open. At the same time, Agent B is trying to book seats for a family of three traveling together. In response to a query, the database indicates that 8ABC and 11DEF are the two remaining groups of three adjacent unassigned seats.

Agent A submits the update command
 SELECT (SEAT-NO = '11D')
 ASSIGN 'MOCK,E' TO PASSENGER-NAME

while Agent B submits the update sequence
 SELECT (SEAT-NO = '11D')
 ASSIGN 'EHLERS,P' TO PASSENGER-NAME

as well as commands for seats 11E and 11F. Then two passengers have been booked into the same seat (which would be uncomfortable, to say the least). Both agents have acted properly: Each sought a list of empty seats, chose one seat from the list, and updated the database to show to whom the seat was assigned. The difficulty in this situation is the time delay between reading a value from the database and writing a modification of that value. During the delay time, another user has accessed the same data.

To resolve this problem, a DBMS treats the entire query-update cycle as a single atomic operation. The command from the agent must now resemble "read the current value of seat PASSENGER-NAME for seat 11D; if it is 'UNASSIGNED', modify it to 'MOCK,E' (or 'EHLERS,P')". The read-modify cycle must be completed as an uninterrupted item without allowing any other users access to the PASSENGER-NAME field for seat 11D. The second agent's request to book would not be considered until after the first agent's had been completed; at that time, the value of PASSENGERNAME would no longer be 'UNASSIGNED'.

A final problem in concurrent access is read-write. Suppose one user is updating a value when a second user wishes to read it. If the read is done while the write is in progress, the reader may receive data that are only partially updated. Consequently, the DBMS locks any read requests until a write has been completed.

8.4.6 Monitors:

The monitor is the unit of a DBMS responsible for the structural integrity of the database. A monitor can check values being entered to ensure their consistency with the rest of the database or with characteristics of the field. For example, a monitor might reject alphabetic characters for a numeric field. We discuss several forms of monitors.

8.4.6.1 Range Comparisons:

A range comparison monitors tests each new value to ensure that the value is within an acceptable range. If the data value is outside the range, it is rejected and not entered the database. For example, the range of dates might be 131, "/", 112, "/", 19002099. An even more sophisticated range check might limit the day portion to 130 for months with 30 days, or it might consider leap year for February. Range comparisons are also convenient for numeric quantities. For example, a salary field might be limited to \$200,000,

or the size of a house might be constrained to be between 500 and 5,000 square feet. Range constraints can also apply to other data having a predictable form.

Range comparisons can be used to ensure the internal consistency of a database. When used in this manner, comparisons are made between two database elements. For example, a grade level from A+ would be acceptable if the record described a student at an elementary school, whereas only 912 would be acceptable for a record of a student in high school. Similarly, a person could be assigned a job qualification score of 75100 only if the person had completed college or had had at least ten years of work experience. Filters or patterns are more general types of data form checks. These can be used to verify that an automobile plate is two letters followed by four digits, or the sum of all digits of a credit card number is a multiple of 9. Checks of these types can control the data allowed in the database. They can also be used to test existing values for reasonableness. If you suspect that the data in a database have been corrupted, a range check of all records could identify those having suspicious values.

8.4.6.2 State Constraints:

State constraints describe the condition of the entire database. At no time should the database values violate these constraints. Phrased differently, if these constraints are not met, some value of the database is in error.

In the section on two-phase updates, we saw how to use a commit flag, which is set at the start of the commit phase and cleared at the completion of the commit phase. The commit flag can be considered a state constraint because it is used at the end of every transaction for which the commit flag is not set. Earlier in this chapter, we described a process to reset the commit flags in the event of a failure after a commit phase. In this way, the status of the commit flag is an integrity constraint on the database. For another example of a state constraint, consider a database of employees' classifications. At any time, at most one employee is classified as "president". Furthermore, each employee has an employee number different from that of every other employee. If a mechanical or software failure causes portions of the database file to be duplicated, one of these uniqueness constraints might be violated. By testing the state of the database, the DBMS could identify records with duplicate employee numbers, or two records classified as "president".

8.4.6.3 Transition Constraints:

State constraints describe the state of a correct database. Transition constraints describe conditions necessary before changes can be applied to a database. For example, before a new employee can be added to the database, there must be a position number in

the database with status "vacant." (That is, an empty slot must exist.) Furthermore, after the employee is added, exactly one slot must be changed from "vacant" to the number of the new employee. Simple range checks and filters can be implemented within most database management systems. However, the more sophisticated state and transition constraints can require special procedures for testing. Such user-written procedures are invoked by the DBMS each time an action must be checked.

8.5 SENSITIVE DATA

Some databases contain what is called sensitive data. As a working definition, let us say that sensitive data are data that should not be made public. Determining which data items and fields are sensitive depends both on the individual database and the underlying meaning of the data. Obviously, some databases, such as a public library catalog, contain no sensitive data; other databases, such as defense-related ones, are totally sensitive. These two cases nothing sensitive and everything sensitive are the easiest to handle because they can be covered by access controls to the database. Someone either is or is not an authorized user. These controls are provided by the operating system. The more difficult problem, which is also the more interesting one, is the case in which *some but not all* the elements in the database are sensitive. There may be varying degrees of sensitivity. For example, a university database might contain student data consisting of name, financial aid, dorm, drug use, sex, parking fines, and race. An example of this database is shown in Table 8-5. Name and dorm are probably the least sensitive; financial aid, parking fines, and drug use the most; sex and race somewhere in between. That is, many people may have legitimate access to name, some to sex and race, and relatively few to financial aid, parking fines, or drug use. Indeed, knowledge of the existence of some fields, such as drug use, may itself be sensitive. Thus, security concerns not only the data elements but also their context and meaning. Furthermore, we must consider different degrees of sensitivity. For instance, although they are all highly sensitive, the financial aid, parking fines, and drug-use fields may not have the same kinds of access restrictions. Our security requirements may demand that a few people be authorized to see each field, but no one be authorized to see all three. The challenge of the access control problem is to limit users' access so that they can obtain only the data to which they have legitimate access. Alternatively, the access control problem forces us to ensure that sensitive data are not to be released to unauthorized people. Several factors can make data sensitive.

- **Inherently sensitive:** The value itself may be so revealing that it is sensitive. Examples are the locations of defensive missiles

or the median income of barbers in a town with only one barber.

- ***From a sensitive source:*** The source of the data may indicate a need for confidentiality. An example is information from an informer whose identity would be compromised if the information were disclosed.
- ***Declared sensitive:*** The database administrator or the owner of the data may have declared the data to be sensitive. Examples are classified military data or the name of the anonymous donor of a piece of art.
- ***Part of a sensitive attribute or a sensitive record:*** In a database, an entire attribute or record may be classified as sensitive. Examples are the salary attribute of a personnel database or a record describing a secret space mission.
- ***Sensitive in relation to previously disclosed information:*** Some data become sensitive in the presence of other data. For example, the longitude coordinate of a secret gold mine reveals little, but the longitude coordinate in conjunction with the latitude coordinate pinpoints the mine.

All these factors must be considered to determine the sensitivity of the data.

8.5.1 Access Decisions:

Remember that a database administrator is a person who decides what data should be in the database and who should have access to it. The database administrator considers the need for different users to know certain information and decides who should have what access. Decisions of the database administrator are based on an access policy. The database manager or DBMS is a program that operates on the database and auxiliary control information to implement the decisions of the access policy. We say that the database manager decides to permit user *x* to access data *y*. Clearly, a program or machine cannot decide anything; it is more precise to say that the program performs the instructions by which *x* accesses *y* as a way of implementing the policy established by the database administrator. To keep explanations concise, we occasionally describe programs as if they can carry out human thought processes. The DBMS may consider several factors when deciding whether to permit an access. These factors include availability of the data, acceptability of the access, and authenticity of the user. We expand on these three factors below.

8.5.1.1 Availability of Data:

One or more required elements may be inaccessible. For

example, if a user is updating several fields, other users' accesses to those fields must be blocked temporarily. This blocking ensures that users do not receive inaccurate information, such as a new street address with an old city and state, or a new code component with old documentation. Blocking is usually temporary. When performing an update, a user may have to block access to several fields or several records to ensure the consistency of data for others. Notice, however, that if the updating user aborts the transaction while the update is in progress, the other users may be permanently blocked from accessing the record. This indefinite postponement is also a security problem, resulting in denial of service.

8.5.1.2 Acceptability of Access:

One or more values of the record may be sensitive and not accessible by the general user. A DBMS should not release sensitive data to unauthorized individuals. Deciding what is sensitive, however, is not as simple as it sounds, because the fields may not be directly requested. A user may have asked for certain records that contain sensitive data, but the user's purpose may have been only to project the values from fields that are not sensitive. Even when a sensitive value is not explicitly given, the database manager may deny access on the grounds that it reveals information the user is not authorized to have. Alternatively, the user may want to derive a non-sensitive statistic from the sensitive data; for example, if the average financial aid value does not reveal any individual's financial aid value, the database management system can safely return the average. However, the average of one data value discloses that value.

8.5.1.3 Assurance of Authenticity:

Certain characteristics of the user external to the database may also be considered when permitting access. For example, to enhance security, the database administrator may permit someone to access the database only at certain times, such as during working hours. Previous user requests may also be considered; repeated requests for the same data or requests that exhaust a certain category of information may be used to find out all elements in a set when a direct query is not allowed. Sensitive data can sometimes be revealed by combined results from several fewer sensitive queries.

8.5.2 Types of Disclosures:

Data can be sensitive, but so can their characteristics. In this section, we see that even descriptive information about data (such as their existence or whether they have an element that is zero) is a form of disclosure.

8.5.2.1 Exact Data:

The most serious disclosure is the *exact value of a sensitive data item* itself. The user may know that sensitive data are being

requested, or the user may request general data without knowing that some of it is sensitive. A faulty database manager may even deliver sensitive data by accident, without the user's having requested it. In all these cases the result is the same: The security of the sensitive data has been breached.

8.5.2.2 Bounds:

Another exposure is disclosing bounds on a sensitive value; that is, indicating that a sensitive value, y , is between two values, L and H . Sometimes, by using a narrowing technique not unlike the binary search, the user may first determine that $L \leq y \leq H$ and then see whether $L \leq y \leq H/2$, and so forth, thereby permitting the user to determine y to any desired precision. In another case, merely revealing that a value such as the athletic scholarship budget or the number of CIA agents exceeds a certain amount may be a serious breach of security.

8.5.2.3 Negative Result:

Sometimes we can word a query to determine a negative result. That is, we can learn that z is *not* the value of y . For example, knowing that 0 is not the total number of felony convictions for a person reveals that the person was convicted of a felony. The distinction between 1 and 2 or 46 and 47 felonies is not as sensitive as the distinction between 0 and 1. Therefore, disclosing that a value is not 0 can be a significant disclosure. Similarly, if a student does not appear on the honours list, you can infer that the person's grade point average is below 3.50. This information is not too revealing, however, because the range of grade point averages from 0.0 to 3.49 is rather wide.

8.5.2.4 Existence:

In some cases, the existence of data is itself a sensitive piece of data, regardless of the actual value. For example, an employer may not want employees to know that their use of long-distance telephone lines is being monitored. In this case, discovering a LONG-DISTANCE field in a personnel file would reveal sensitive data.

8.5.2.5 Probable Value:

Finally, it may be possible to determine the probability that a certain element has a certain value. To see how, suppose you want to find out whether the president of the United States is registered in the Tory party. Knowing that the president is in the database, you submit two queries to the database:

- How many people have 1600 Pennsylvania Avenue as their official residence? (Response: 4)
- How many people have 1600 Pennsylvania Avenue as their official residence and have YES as the value of TORY?

(Response: 1)

From these queries you conclude there is a 25 percent likelihood that the president is a registered Tory.

8.5.2.6 Summary of Partial Disclosure:

We have seen several examples of how a security problem can result if characteristics of sensitive data are revealed. Notice that some of the techniques we presented used information *about* the data, rather than direct access to the data, to infer sensitive results. A successful security strategy must protect from both direct and indirect disclosure.

8.5.3 Security versus Precision:

Our examples have illustrated how difficult it is to determine which data are sensitive and how to protect them. The situation is complicated by a desire to share non-sensitive data. For reasons of confidentiality we want to disclose only those data that are not sensitive. Such an outlook encourages a conservative philosophy in determining what data to disclose: less is better than more. On the other hand, consider the users of the data. The conservative philosophy suggests rejecting any query that mentions a sensitive field. We may thereby reject many reasonable and non-disclosing queries. For example, a researcher may want a list of grades for all students using drugs, or a statistician may request lists of salaries for all men and for all women. These queries probably do not compromise the identity of any individual. We want to disclose as much data as possible so that users of the database have access to the data they need. This goal, called precision, aims to protect all sensitive data while revealing as much non-sensitive data as possible.

We can depict the relationship between security and precision with concentric circles. As Figure 8-2 shows, the sensitive data in the central circle should be carefully concealed. The outside band represents data we willingly disclose in response to queries. But we know that the user may put together pieces of disclosed data and infer other, more deeply hidden, data. The figure shows us that beneath the outer layer may be yet more non-sensitive data that the user cannot infer. The ideal combination of security and precision allows us to maintain perfect confidentiality with maximum precision; in other words, we disclose all and only the non-sensitive data. But achieving this goal is not as easy as it might seem.

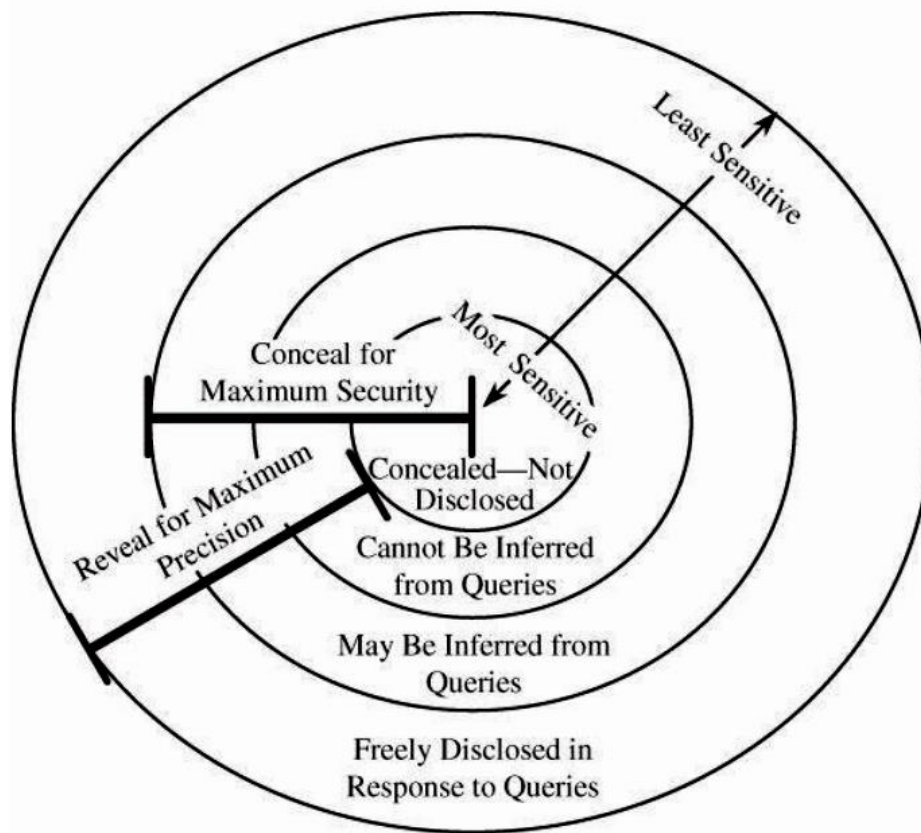


Figure 8-2 Security versus Precision

8.6 SUMMARY

- Reliability, correctness, and integrity are three closely related concepts in databases. Users trust the DBMS to maintain their data correctly, so integrity issues are very important to database security. This chapter has addressed the confidentiality and integrity problems specific to database applications for database management systems.
- Both confidentiality and integrity are important to users of databases. Confidentiality can be broken by indirect disclosure of a negative result or of the bounds of a value. Integrity of the entire database is a responsibility of the DBMS software; this problem is handled by most major commercial systems through backups, redundancy, change logs, and two-step updates.
- Integrity of an individual element of the database is the responsibility of the database administrator who defines the access policy.

8.7 REVIEW QUESTIONS

- a) What are the components of a database?
- b) Discuss the advantages of using databases.
- c) What are the requirements for database security?
- d) Write a short note on the element integrity for database security.
- e) Explain the two-phase update technique for database integrity.
- f) How can concurrent access and consistency be maintained in databases?
- g) Write a short note on monitors.
- h) What is sensitive data? Explain the factors that make data sensitive.
- i) What are the different types of disclosures of data?
- j) Differentiate between security and precision.

8.8 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- *Security in Computing* by C. P. Pfleeger, and S. L. Pfleeger, Pearson Education.
- *Computer Security: Art and Science* by Matt Bishop, Pearson Education.
- *Cryptography and Network Security: Principles and practice* by Stallings
- *Network Security* by Kaufman, Perlman, Speciner
- *Network Security : A Beginner's Guide* by Eric Maiwald, TMH
- *Java Network Security* by Macro Pistoia, Pearson Education
- *Principles of information security* by Whitman, Mattord, Thomson.



DATABASE SECURITY - II

Unit Structure:

9.0 Objectives

9.1 Introduction

9.2 Inference

9.2.1. Direct Attack

9.2.2. Indirect Attack

9.2.2.1. Sum

9.2.2.2. Count

9.2.2.3. Mean

9.2.2.4. Median

9.2.2.5. Tracker Attacks

9.2.2.6. Linear System Vulnerability

9.2.2.7. Controls for Statistical Inference Attacks

9.2.2.8. Limited Response Suppression

9.2.2.9. Combined Results

9.2.2.10. Conclusion on the Inference Problem

9.2.3. Aggregation

9.3 Multilevel Databases

9.3.1. The Case for Differentiated Security

9.3.2. Granularity

9.3.3. Security Issues

9.3.3.1. Integrity

9.3.3.2. Confidentiality

9.4 Proposals for Multilevel Security

9.4.1. Separation

9.4.1.1. Partitioning

9.4.1.2. Encryption

- 9.4.1.3. Integrity Lock
- 9.4.1.4. Sensitivity Lock
- 9.4.2. Designs of Multilevel Secure Databases
 - 9.4.2.1. Integrity Lock
 - 9.4.2.2. Trusted Front End
 - 9.4.2.3. Commutative Filters
 - 9.4.2.4. Distributed Databases
 - 9.4.2.5. Window/View
- 9.4.3. Practical Issues
- 9.5 Summary
- 9.6 Review Questions
- 9.7 Bibliography, References and Further Reading

9.0 OBJECTIVES

In this chapter, we study two major (but related) database security problems, the inference problem and the multilevel problem. Both problems are complex, and there are no immediate solutions. However, by understanding the problems, we become more sensitive to ways of reducing potential threats to the data.

9.1 INTRODUCTION

Protecting data is at the heart of many secure systems, and many users (people, programs, or systems) rely on a database management system (DBMS) to manage the protection. There is substantial current interest in DBMS security because databases are newer than programming and operating systems. However, the protection provided by database management systems has had mixed results. Over time, we have improved our understanding of database security problems, and several good controls have been developed. But, as you will see, there are still more security concerns for which there are no available controls.

9.2 INFERENCE

Inference is a way to infer or derive sensitive data from non-sensitive data. The inference problem is a subtle vulnerability in database security. The database in Table 9-1 can help illustrate the inference problem. AID is the amount of financial aid a student is receiving. FINES is the amount of parking fines still owed. DRUGS is the result of a drug-use survey: 0 means never used and 3 means

frequent user. Obviously, this information should be kept confidential. We assume that AID, FINES, and DRUGS are sensitive fields, although only when the values are related to a specific individual. In this section, we look at ways to determine sensitive data values from the database.

Name	Sex	Race	Aid	Fines	Drugs	Dorm
Adams	M	C	5000	45.	1	Holmes
Bailey	M	B	0	0.	0	Grey
Chin	F	A	3000	20.	0	West
Dewitt	M	B	1000	35.	3	Grey
Earhart	F	C	2000	95.	1	Holmes
Fein	F	C	1000	15.	0	West
Groff	M	C	4000	0.	3	West
Hill	F	B	5000	10.	2	Holmes
Koch	F	C	0	0.	1	West
Liu	F	A	0	10.	2	Grey
Majors	M	C	2000	0.	2	Grey

Table 9-1 Database to illustrate inferences

9.2.1 Direct Attack:

In a direct attack, a user tries to determine values of sensitive fields by seeking them directly with queries that yield few records. The most successful technique is to form a query so specific that it matches exactly one data item. In Table 9-1, a sensitive query might be

List NAME where
SEX=M \wedge DRUGS=1

This query discloses that for record ADAMS, DRUGS=1. However, it is an obvious attack because it selects people for whom DRUGS=1, and the DBMS might reject the query because it selects records for a specific value of the sensitive attribute DRUGS. A less obvious query is

List NAME where
 $(SEX = M \wedge DRUGS = 1) \vee$
 $(SEX \neq M \wedge SEX \neq F) \vee$
 $(DORM = AYRES)$

On the surface, this query looks as if it should conceal drug usage by selecting other non-drug-related records as well. However, this query still retrieves only one record, revealing a name that corresponds to the sensitive DRUG value. The DBMS needs to know that SEX has only two possible values so that the second clause will select no records. Even if that were possible, the DBMS would also need to know that no records exist with DORM=AYRES, even though AYRES might in fact be an acceptable value for DORM. Organizations that publish personal statistical data, such as the U.S. Census Bureau, do not reveal results when a small number of people make up a large proportion of a category. The rule of “ n items over k percent” means that data should be withheld if n items represent over k percent of the result reported. In the previous case, the one person selected represents 100 percent of the data reported, so there would be no ambiguity about which person matches the query.

9.2.2 Indirect Attack:

Another procedure, used by the U.S. Census Bureau and other organizations that gather sensitive data, is to release only statistics. The organizations suppress individual names, addresses, or other characteristics by which a single individual can be recognized. Only neutral statistics, such as sum, count, and mean, are released. The indirect attack seeks to infer a result based on one or more intermediate statistical results. But this approach requires work outside the database itself. A statistical attack seeks to use some apparently anonymous statistical measure to infer individual data. In the following sections, we present several examples of indirect attacks on databases that report statistics.

9.2.2.1 Sum:

An attack by sum tries to infer a value from a reported sum. For example, with the sample database in Table 9-1, it might seem safe to report student aid total by sex and dorm. Such a report is shown in Table 9-2. This seemingly innocent report reveals that no female living in Grey is receiving financial aid. Thus, we can infer that any female living in Grey (such as Liu) is certainly not receiving financial aid. This approach often allows us to determine a negative result.

	Holmes	Grey	West	Total
M	5000	3000	4000	12000
F	7000	0	4000	11000
Total	12000	3000	8000	23000

Table 9-2 Table showing negative result

9.2.2.2 Count:

The count can be combined with the sum to produce some even more revealing results. Often these two statistics are released for a database to allow users to determine average values. (Conversely, if count and mean are released, sum can be deduced.) Table 9-3 shows the count of records for students by dorm and sex. This table is innocuous by itself. Combined with the sum table, however, this table demonstrates that the two males in Holmes and West are receiving financial aid in the amount of \$5000 and \$4000, respectively. We can obtain the names by selecting the subschema of NAME, DORM, which is not sensitive because it delivers only low-security data on the entire database.

Sex	Holmes	Grey	West	Total
M	1	3	1	5
F	2	1	3	6
Total	3	4	4	11

Table 9-3 Inference from Count and Sum Results

9.2.2.3 Mean:

The arithmetic mean (average) allows exact disclosure if the attacker can manipulate the subject population. As a trivial example, consider salary. Given the number of employees, the mean salary for a company and the mean salary of all employees except the president, it is easy to compute the president's salary.

9.2.2.4 Median:

By a slightly more complicated process, we can determine an individual value from the **median**, the midpoint of an ordered list of values. The attack requires finding selections having one point of intersection that happens to be exactly in the middle. For example,

in our sample database, there are five males and three persons whose drug use value is 2. Arranged in order of aid, these lists are shown in table 9-4. Notice that Majors is the only name common to both lists, and conveniently that name is in the middle of each list. Someone working at the Health Clinic might be able to find out that Majors is a white male whose drug-use score is 2. That information identifies Majors as the intersection of these two lists and pinpoints Majors' financial aid as \$2000. In this example, the queries

$q = \text{median}(\text{AID where SEX} = \text{M})$
 $p = \text{median}(\text{AID where DRUGS} = 2)$

reveal the exact financial aid amount for Majors.

Name	Sex	Drugs	Aid
Bailey	M	0	0
Dewitt	M	3	1000
Majors	M	2	2000
Groff	M	3	4000
Adams	M	1	5000
Liu	F	2	0
Majors	M	2	2000
Hill	F	2	5000

Table 9-4 Drug Use and Aid Result

9.2.2.5 Tracker Attacks:

Database management systems may conceal data when a small number of entries make up a large proportion of the data revealed. A tracker attack can fool the database manager into locating the desired data by using additional queries that produce small results. The tracker adds additional records to be retrieved for two different queries; the two sets of records cancel each other out, leaving only the statistic or data desired. The approach is to use intelligent padding of two queries. In other words, instead of trying to identify a unique value, we request $n - 1$ other values (where there are n values in the database). Given n and $n - 1$, we can easily compute the desired single element.

9.2.2.6 Linear System Vulnerability:

A tracker is a specific case of a more general vulnerability. With a little logic, algebra, and luck in the distribution of the database contents, it may be possible to construct a series of queries that returns results relating to several different sets. For example, the following system of five queries does not overtly reveal any single c value from the database. However, the queries' equations can be solved for each of the unknown c values, revealing them all. In fact, this attack can also be used to obtain results *other than* numerical ones. Recall that we can apply logical rules to *and* (\wedge) and *or* (\vee), typical operators for database queries, to derive values from a series of logical expressions. For example, each expression might represent a query asking for precise data instead of counts.

9.2.2.7 Controls for Statistical Inference Attacks:

Denning and Schlörer present a very good survey of techniques for maintaining security in databases. The controls for all statistical attacks are similar. Essentially, there are two ways to protect against inference attacks: Either controls are applied to the queries or controls are applied to individual items within the database. As we have seen, it is difficult to determine whether a given query discloses sensitive data. Thus, query controls are effective primarily against direct attacks.

Suppression and concealing are two controls applied to data items. With suppression, sensitive data values are not provided; the query is rejected without response. With concealing, the answer provided is *close to* but not exactly the actual value. These two controls reflect the contrast between security and precision. With suppression, any results provided are correct, yet many responses must be withheld to maintain security. With concealing, more results can be provided, but the precision of the results is lower. The choice between suppression and concealing depends on the context of the database.

9.2.2.8 Limited Response Suppression:

The n -item k -percent rule eliminates certain low-frequency elements from being displayed. It is not enough to delete them, however, if their values can also be inferred. When one cell is suppressed in a table with totals for rows and columns, it is necessary to suppress at least one additional cell on the row and one on the column to provide some confusion. Using this logic, all cells (except totals) would have to be suppressed in this small sample table. When totals are not provided, single cells in a row or column can be suppressed.

9.2.2.9 Combined Results:

Another control combines rows or columns to protect sensitive values. For example, Table 9-5 shows several sensitive results that identify single individuals. These counts, combined with other results such as sum, permit us to infer individual drug-use values for the three males, as well as to infer that no female was rated 3 for drug use. To suppress such sensitive information, it is possible to combine the attribute values for 0 and 1, and for 2 and 3, producing the less sensitive results shown in Table 9-6. In this instance, it is impossible to identify any single value.

Table 9-5 Students by Sex and Drug Use

Sex	Drug Use			
	0	1	2	3
M	1	1	1	2
F	2	2	2	0

Table 9-6 Suppression by Combining Revealing Values

Sex	Drug Use	
	0 or 1	2 or 3
M	2	3
F	4	2

9.2.2.10 Conclusion on the Inference Problem:

There are no perfect solutions to the inference problem. The approaches to controlling it follow the three paths listed below. The first two methods can be used either to limit queries accepted or to limit data provided in response to a query. The last method applies only to data released.

- *Suppress obviously sensitive information:* This action can be taken easily. The tendency is to err on the side of suppression, thereby restricting the usefulness of the database.
- *Track what the user knows:* Although possibly leading to the greatest safe disclosure, this approach is extremely costly. Information must be maintained on all users, even though most are not trying to obtain sensitive data. Moreover, this approach seldom considers what any two people may know together and cannot address what a single user can accomplish by using

multiple IDs.

- *Disguise the data:* Random perturbation and rounding can inhibit statistical attacks that depend on exact values for logical and algebraic manipulation. The users of the database receive slightly incorrect or possibly inconsistent results.

It is unlikely that research will reveal a simple, easy-to-apply measure that determines exactly which data can be revealed without compromising sensitive data. Nevertheless, an effective control for the inference problem is just knowing that it exists. As with other problems in security, recognition of the problem leads to understanding of the purposes of controlling the problem and to sensitivity to the potential difficulties caused by the problem.

9.2.3 Aggregation:

Related to the inference problem is aggregation, which means building sensitive results from less sensitive inputs. We saw earlier that knowing either the latitude or longitude of a gold mine does you no good. But if you know both latitude and longitude, you can pinpoint the mine. For a more realistic example, consider how police use aggregation frequently in solving crimes: They determine who had a motive for committing the crime, when the crime was committed, who had alibis covering that time, who had the skills, and so forth. Typically, you think of police investigation as starting with the entire population and narrowing the analysis to a single person. But if the police officers work in parallel, one may have a list of possible suspects, another may have a list with possible motive, and another may have a list of capable persons. When the intersection of these lists is a single person, the police have their prime suspect.

Addressing the aggregation problem is difficult because it requires the database management system to track which results each user had already received and conceal any result that would let the user derive a more sensitive result. Aggregation is especially difficult to counter because it can take place outside the system. For example, suppose the security policy is that anyone can have *either* the latitude or longitude of the mine, but not both. Nothing prevents you from getting one, your friend from getting the other, and the two of you talking to each other.

Recent interest in data mining has raised concern again about aggregation. Data mining is the process of sifting through multiple databases and correlating multiple data elements to find useful information. Marketing companies use data mining extensively to find consumers likely to buy a product.

Aggregation was of interest to database security researchers at the same time as was inference. As we have seen, some

approaches to inference have proven useful and are currently being used. But there have been few proposals for countering aggregation.

9.3 MULTILEVEL DATABASES

So far, we have considered data in only two categories: either sensitive or non-sensitive. We have alluded to some data items being more sensitive than others, but we have allowed only yes-or-no access. Our presentation may have implied that sensitivity was a function of the *attribute*, the column in which the data appeared, although nothing we have done depended on this interpretation of sensitivity. In fact, though, sensitivity is determined not just by attribute but also in ways that we investigate in the next section.

9.3.1 The Case for Differentiated Security:

Consider a database containing data on U.S. government expenditures. Some of the expenditures are for paper clips, which is not sensitive information. Some salary expenditures are subject to privacy requirements. Individual salaries are sensitive, but the aggregate (for example, the total Agriculture Department payroll, which is a matter of public record) is not sensitive. Expenses of certain military operations are more sensitive; for example, the total amount the United States spends for ballistic missiles, which is not public. There are even operations known only to a few people, and so the amount spent on these operations, or even the fact that anything was spent on such an operation, is highly sensitive. From this description, three characteristics of database security emerge.

- The security of a single element may be different from the security of other elements of the same record or from other values of the same attribute. That is, the security of one element may differ from that of other elements of the same row or column. This situation implies that security should be implemented for each individual element.
- Two levels sensitive and non-sensitive are inadequate to represent some security situations. Several grades of security may be needed. These grades may represent ranges of allowable knowledge, which may overlap. Typically, the security grades form a lattice.
- The security of an aggregate, a sum, a count, or a group of values in a database may differ from the security of the individual elements. The security of the aggregate may be higher or lower than that of the individual elements.

These three principles lead to a model of security not unlike the military model of security encountered earlier, in which the sensitivity of an object is defined as one of n levels and is further separated into compartments by category.

9.3.2 Granularity:

Recall that the military classification model applied originally to paper documents and was adapted to computers. It is easy to classify and track a single sheet of paper or, for that matter, a paper file, a computer file, or a single program or process. It is entirely different to classify individual data items. For obvious reasons, an entire sheet of paper is classified at one level, even though certain words, such as *and*, *the*, or *of*, would be innocuous in any context, and other words, such as codewords like *Manhattan project*, might be sensitive in any context. But defining the sensitivity of each value in a database is like applying a sensitivity level to each individual word of a document.

And the problem is still more complicated. The word *Manhattan* by itself is not sensitive, nor is *project*. However, the combination of these words produces the sensitive codeword *Manhattan project*. A similar situation occurs in databases. Therefore, not only can every *element* of a database have a distinct sensitivity, every *combination of elements* can also have a distinct sensitivity. Furthermore, the combination can be sensitive than any of its elements. So, what would we need in order to associate a sensitivity level with each value of a database? First, we need an access control policy to dictate which users may have access to what data. Typically, to implement this policy each data item is marked to show its access limitations. Second, we need a means to guarantee that the value has not been changed by an unauthorized person. These two requirements address both confidentiality and integrity.

9.3.3 Security Issues:

In Chapter 1, we introduced three general security concerns: integrity, confidentiality, and availability. In this section, we extend the first two of these concepts to include their special roles for multilevel databases.

9.3.3.1 Integrity:

Even in a single-level database in which all elements have the same degree of sensitivity, integrity is a tricky problem. In the case

of multilevel databases, integrity becomes both more important and more difficult to achieve. Because of the *-property for access control, a process that reads high-level data is not allowed to write a file at a lower level. Applied to databases, however, this principle says that a high-level user should not be able to write a lower-level data element. The problem with this interpretation arises when the DBMS must be able to read all records in the database and write new records for any of the following purposes: to do backups, to scan the database to answer queries, to reorganize the database according to a user's processing needs, or to update all records of the database.

When people encounter this problem, they handle it by using trust and common sense. People who have access to sensitive information are careful not to convey it to uncleared individuals. In a computing system, there are two choices: Either the process cleared at a high level cannot write to a lower level or the process must be a "trusted process," the computer equivalent of a person with a security clearance.

9.3.3.2 Confidentiality:

Users trust that a database will provide correct information, meaning that the data are consistent and accurate. As indicated earlier, some means of protecting confidentiality may result in small changes to the data. Although these perturbations should not affect statistical analyses, they may produce two different answers representing the same underlying data value in response to two differently formed queries. In the multilevel case, two different users operating at two different levels of security might get two different answers to the same query. To preserve confidentiality, precision is sacrificed.

Enforcing confidentiality also leads to unknowing redundancy. Suppose a personnel specialist works at one level of access permission. The specialist knows that Bob Hill works for the company. However, Bob's record does not appear on the retirement payment roster. The specialist assumes this omission is an error and creates a record for Bob. The reason that no record for Bob appears is that Bob is a secret agent, and his employment with the company is not supposed to be public knowledge. A record on Bob is in the file but, because of his special position, his record is not accessible to the personnel specialist. The DBMS cannot reject the record from the personnel specialist because doing so would reveal that there already is such a record at a sensitivity too high for the specialist to see. The creation of the new record means that there are now two records for Bob Hill: one sensitive and one not. This situation is called polyinstantiation, meaning that one record can appear (be instantiated) many times, with a different level of confidentiality each

time. In our zeal to reduce polyinstantiation, we must be careful not to eliminate legitimate records such as these.

9.4 PROPOSALS FOR MULTILEVEL SECURITY

As you can already tell, implementing multilevel security for databases is difficult, probably more so than in operating systems, because of the small granularity of the items being controlled. In the remainder of this section, we study approaches to multilevel security for databases.

9.4.1 Separation:

As we have already seen, separation is necessary to limit access. In this section, we study mechanisms to implement separation in databases. Then, we see how these mechanisms can help to implement multilevel security for databases.

9.4.1.1 Partitioning:

The obvious control for multilevel databases is partitioning. The database is divided into separate databases, each at its own level of sensitivity. This approach is like maintaining separate files in separate file cabinets. This control destroys a basic advantage of databases: elimination of redundancy and improved accuracy through having only one field to update. Furthermore, it does not address the problem of a high-level user who needs access to some low-level data combined with high-level data. Nevertheless, because of the difficulty of establishing, maintaining, and using multilevel databases, many users with data of mixed sensitivities handle their data by using separate, isolated databases.

9.4.1.2 Encryption:

If sensitive data are encrypted, a user who accidentally receives them cannot interpret the data. Thus, each level of sensitive data can be stored in a table encrypted under a key unique to the level of sensitivity. But encryption has certain disadvantages. First, a user can mount a chosen plaintext attack. Suppose party affiliation of REP or DEM is stored in encrypted form in each record. A user who achieves access to these encrypted fields can easily decrypt them by creating a new record with party=DEM and comparing the resulting encrypted version to that element in all other records. Worse, if authentication data are encrypted, the malicious user can substitute the encrypted form of his or her own data for that of any other user. Not only does this provide access for the malicious user, but it also excludes the legitimate user whose authentication data have been changed to that of the malicious user.

Using a different encryption key for each record overcomes these defects. Each record's fields can be encrypted with a different key, or all fields of a record can be cryptographically linked, as with cipher block chaining. The disadvantage, then, is that each field must be decrypted when users perform standard database operations such as "select all records with SALARY > 10,000." Decrypting the SALARY field, even on rejected records, increases the time to process a query. Thus, encryption is not often used to implement separation in databases.

9.4.1.3 Integrity Lock:

The integrity lock was first proposed at the U.S. Air Force Summer Study on Data Base Security. The lock is a way to provide both integrity and limited access for a database. The operation was nicknamed "spray paint" because each element is figuratively painted with a colour that denotes its sensitivity. The colouring is maintained with the element, not in a master database table. Each apparent data item consists of three pieces: the actual data item itself, a sensitivity label, and a checksum.

The sensitivity label defines the sensitivity of the data, and the checksum is computed across both data and sensitivity label to prevent unauthorized modification of the data item or its label. The actual data item is stored in plaintext, for efficiency because the DBMS may need to examine many fields when selecting records to match a query. The sensitivity label should be

- *unforgeable*, so that a malicious subject cannot create a new sensitivity level for an element
- *unique*, so that a malicious subject cannot copy a sensitivity level from another element
- *concealed*, so that a malicious subject cannot even determine the sensitivity level of an arbitrary element

The third piece of the integrity lock for a field is an error-detecting code, called a cryptographic checksum. To guarantee that a data value or its sensitivity classification has not been changed, this checksum must be unique for a given element, and must contain both the element's data value and something to tie that value to a position in the database. An appropriate cryptographic checksum includes something unique to the record (the record number), something unique to this data field within the record (the field attribute name), the value of this element, and the sensitivity classification of the element. These four components guard against anyone's changing, copying, or moving the data. The checksum can be computed with a strong encryption algorithm or hash function.

9.4.1.4 Sensitivity Lock:

A sensitivity lock is a combination of a unique identifier (such as the record number) and the sensitivity level. Because the identifier is unique, each lock relates to one record. Many different elements will have the same sensitivity level. A malicious subject should not be able to identify two elements having identical sensitivity levels or identical data values just by looking at the sensitivity level portion of the lock. Because of the encryption, the lock's contents, especially the sensitivity level, are concealed from plain view. Thus, the lock is associated with one specific record, and it protects the secrecy of the sensitivity level of that record.

9.4.2 Designs of Multilevel Secure Databases:

This section covers different designs for multilevel secure databases. These designs show the trade-offs among efficiency, flexibility, simplicity, and trustworthiness.

9.4.2.1 Integrity Lock:

The integrity lock DBMS was invented as a short-term solution to the security problem for multilevel databases. The intention was to be able to use any (untrusted) database manager with a trusted procedure that handles access control. The sensitive data were obliterated or concealed with encryption that protected both a data item and its sensitivity. In this way, only the access procedure would need to be trusted because only it would be able to achieve or grant access to sensitive data.

The efficiency of integrity locks is a serious drawback. The space needed for storing an element must be expanded to contain the sensitivity label. Because there are several pieces in the label and one label for every element, the space required is significant. Problematic, too, is the processing time efficiency of an integrity lock. The sensitivity label must be decoded every time a data element is passed to the user to verify that the user's access is allowable. Also, each time a value is written or modified; the label must be recomputed. Thus, substantial processing time is consumed. If the database file can be sufficiently protected, the data values of the individual elements can be left in plaintext. That approach benefits select and project queries across sensitive fields because an element need not be decrypted just to determine whether it should be selected. A final difficulty with this approach is that the untrusted database manager sees all data, so it is subject to Trojan horse attacks by which data can be leaked through covert channels.

9.4.2.2 Trusted Front End:

A trusted front end is also known as a guard and operates much like the reference monitor. This approach, originated by Hinke and Schaefer, recognizes that many DBMSs have been built and put into use without consideration of multilevel security. Staff members are already trained in using these DBMSs, and they may in fact use them frequently. The front-end concept takes advantage of existing tools and expertise, enhancing the security of these existing systems with minimal change to the system. The interaction between a user, a trusted front end, and a DBMS involves the following steps:

- a) A user identifies himself or herself to the front end; the front end authenticates the user's identity.
- b) The user issues a query to the front end.
- c) The front end verifies the user's authorization to data.
- d) The front end issues a query to the database manager.
- e) The database manager performs I/O access, interacting with low-level access control to achieve access to actual data.
- f) The database manager returns the result of the query to the trusted front end.
- g) The front end analyzes the sensitivity levels of the data items in the result and selects those items consistent with the user's security level.
- h) The front end transmits selected data to the untrusted front end for formatting.
- i) The untrusted front end transmits formatted data to the user.

The trusted front end serves as a one-way filter, screening out results the user should not be able to access. But the scheme is inefficient because potentially much data is retrieved and then discarded as inappropriate for the user.

9.4.2.3 Commutative Filters:

A commutative filter is a process that forms an interface between the user and a DBMS. However, unlike the trusted front end, the filter tries to capitalize on the efficiency of most DBMSs. The filter reformats the query so that the database manager does as much of the work as possible, screening out many unacceptable records. The filter then provides a second screening to select only data to which the user has access. Filters can be used for security at the record, attribute, or element level.

- When used at the record level, the filter requests desired data plus cryptographic checksum information; it then verifies the accuracy and accessibility of data to be passed to the user.

- At the attribute level, the filter checks whether all attributes in the user's query are accessible to the user and, if so, passes the query to the database manager. On return, it deletes all fields to which the user has no access rights.
- At the element level, the system requests desired data plus cryptographic checksum information. When these are returned, it checks the classification level of every element of every record retrieved against the user's level.

The commutative filter re-forms the original query in a trustable way so that sensitive information is never extracted from the database. The filter works by restricting the query to the DBMS and then restricting the results before they are returned to the user. The advantage of the commutative filter is that it allows query selection, some optimization, and some subquery handling to be done by the DBMS. This delegation of duties keeps the size of the security filter small, reduces redundancy between it and the DBMS, and improves the overall efficiency of the system.

9.4.2.4 Distributed Databases:

The distributed or federated database is a fourth design for a secure multilevel database. In this case, a trusted front-end controls access to two unmodified commercial DBMSs: one for all low-sensitivity data and one for all high-sensitivity data. The front end takes a user's query and formulates single-level queries to the databases as appropriate. For a user cleared for high-sensitivity data, the front end submits queries to both the high- and low-sensitivity databases. But if the user is not cleared for high-sensitivity data, the front end submits a query to only the low-sensitivity database. If the result is obtained from either back-end database alone, the front end passes the result back to the user. If the result comes from both databases, the front end must combine the results appropriately. For example, if the query is a join query having some high-sensitivity terms and some low, the front end must perform the equivalent of a database join itself.

The distributed database design is not popular because the front end, which must be trusted, is complex, potentially including most of the functionality of a full DBMS itself. In addition, the design does not scale well to many degrees of sensitivity; each sensitivity level of data must be maintained in its own separate database.

9.4.2.5 Window/View:

Traditionally, one of the advantages of using a DBMS for multiple users of different interests (but not necessarily different sensitivity levels) is the ability to create a different view for each user. That is, each user is restricted to a picture of the data reflecting only what the user needs to see. For example, the registrar may see only

the class assignments and grades of each student at a university, not needing to see extracurricular activities or medical records. The university health clinic, on the other hand, needs medical records and drug-use information but not scores on standardized academic tests.

The notion of a window or a view can also be an organizing principle for multilevel database access. A window is a subset of a database, containing exactly the information that a user is entitled to access. A view can represent a single user's subset database so that all a user's queries access only that database. This subset guarantees that the user does not access values outside the permitted ones, because non permitted values are not even in the user's database. The view is specified as a set of relations in the database, so the data in the view subset change as data change in the database.

A view may involve computation or complex selection criteria to specify subset data. The data presented to a user is obtained by filtering of the contents of the original database. Attributes, records, and elements are stripped away so that the user sees only acceptable items. Any attribute (column) is withheld unless the user is authorized to access at least one element. Any record (row) is withheld unless the user is authorized to access at least one element. Then, for all elements that remain, if the user is not authorized to access the element, it is replaced by UNDEFINED. This last step does not compromise any data because the user knows the existence of the attribute (there is at least one element that the user can access) and the user knows the existence of the record (again, at least one accessible element exists in the record). In addition to elements, a view includes relations on attributes. Furthermore, a user can create new relations from new and existing attributes and elements. These new relations are accessible to other users, subject to the standard access rights. A user can operate on the subset database defined in a view only as allowed by the operations authorized in the view. As an example, a user might be allowed to retrieve records specified in one view or to retrieve and update records as specified in another view.

9.4.3 Practical Issues:

The multilevel security problem for databases has been studied since the 1970s. Several promising research results have been identified, as we have seen in this chapter. However, as with trusted operating systems, the consumer demand has not been enough to support many products. Civilian users have not liked the inflexibility of the military multilevel security model, and there have

been too few military users. Consequently, multilevel secure databases are primarily of research and historical interest.

The general concepts of multilevel databases are important. We do need to be able to separate data according to their degree of sensitivity. Similarly, we need ways of combining data of different sensitivities into one database (or at least into one virtual database or federation of databases). And these needs will only increase over time as larger databases contain more sensitive information, especially for privacy concerns.

9.5 SUMMARY

- This chapter has addressed two aspects of security for database management systems: the inference problem for statistical databases, and problems of including users and data of different sensitivity levels in one database.
- The inference problem in a statistical database arises from the mathematical relationships between data elements and query results. We studied controls for preventing statistical inference, including limited response suppression, perturbation of results, and query analysis.
- One very complex control involves monitoring all data provided to a user in order to prevent inference from independent queries.
- Multilevel secure databases must provide both confidentiality and integrity. Separation can be implemented physically, logically, or cryptographically. We explored five approaches for ensuring confidentiality in multilevel secure databases: integrity lock, trusted front end, commutative filters, distributed databases, and restricted views. Other solutions are likely to evolve as the problem is studied further.

9.6 REVIEW QUESTIONS

- a) Write a short note on inference.
- b) Explain direct attack of inference.
- c) What are indirect attacks of inference? What are its types?
- d) What is aggregation? Differentiate between aggregation and inference.
- e) Write a short note on multilevel databases.
- f) How can separation limit access in databases?
- g) List and explain the designs for multilevel secure databases.

9.7 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- *Security in Computing* by C. P. Pfleeger, and S. L. Pfleeger, Pearson Education.
- *Computer Security: Art and Science* by Matt Bishop, Pearson Education.
- *Cryptography and Network Security: Principles and practice* by Stallings
- *Network Security* by Kaufman, Perlman, Speciner
- *Network Security : A Beginner's Guide* by Eric Maiwald, TMH
- *Java Network Security* by Macro Pistoia, Pearson Education
- *Principles of information security* by Whitman, Mattord, Thomson

SECURITY IN NETWORKS - I

Unit Structure:

- 10.1 Objectives
- 10.2 Introduction
- 10.3 Network Concepts
 - 10.3.1. The Network
 - 10.3.2. Media
 - 10.3.3. Protocols
 - 10.3.4. Types of Networks
- 10.4 Threats in Networks
 - 10.4.1. What makes a network vulnerable?
 - 10.4.2. Who attacks networks?
 - 10.4.3. Reconnaissance
 - 10.4.4. Threats in Transit: Eavesdropping and Wiretapping
 - 10.4.5. Protocol Flaws
 - 10.4.6. Impersonation
 - 10.4.7. Message Confidentiality Threats
 - 10.4.8. Message Integrity Threats
 - 10.4.9. Format Failures
 - 10.4.10. Website Vulnerabilities
 - 10.4.11. Denial of Service
 - 10.4.12. Distributed Denial of Service
 - 10.4.13. Threats in Active or Mobile Code
 - 10.4.14. Complex Attacks
- 10.5 Summary
- 10.6 Review Questions
- 10.7 Bibliography, References and Further Reading

10.1 OBJECTIVES

At the end of this chapter, you will be able to understand:

- Networks vs. stand-alone applications and environments: differences and similarities
- Threats against networked applications, including denial of service, web site defacements, malicious mobile code, and protocol attacks

10.2 INTRODUCTION

Networks, their design, development, and usage are critical to our style of computing. We interact with networks daily, when we perform banking transactions, make telephone calls, or ride trains and planes. The utility companies use networks to track electricity or water usage and bill for it. When we pay for groceries or gasoline, networks enable our credit or debit card transactions and billing. Life without networks would be considerably less convenient, and many activities would be impossible. Not surprisingly, then, computing networks are attackers' targets of choice. Because of their actual and potential impact, network attacks attract the attention of journalists, managers, auditors, and the general public.

In this chapter we describe what makes a network like and different from an application program or an operating system. In investigating networks, you will learn how the concepts of confidentiality, integrity, and availability apply in networked settings. At the same time, you will see that the basic notions of identification and authentication, access control, accountability, and assurance are the basis for network security, just as they have been in other settings.

10.3 NETWORK CONCEPTS

To study network threats and controls, we first must review some of the relevant networking terms and concepts. Networks are both fragile and strong. To see why, think about the power, cable television, telephone, or water network that serves your home. If a falling tree branch breaks the power line to your home, you are without electricity until that line is repaired; you are vulnerable to what is called a single point of failure, because one cut to the network destroys electrical functionality for your entire home. From the user's perspective, a network is sometimes designed so that it looks like two endpoints with a single connection in the middle. For example, the municipal water supply may appear to be little more than a reservoir (the source), the pipes (the transmission or communication medium), and your water faucet (the destination). Although this simplistic view is functionally correct, it ignores the complex design,

implementation, and management of the "pipes." In a similar way, we describe computer networks in this chapter in ways that focus on the security concepts but present the networks themselves in a simplistic way, to highlight the role of security and prevent the complexity of the networks from distracting our attention.

10.3.1 The Network:

A network in its simplest form, is two devices connected across some medium by hardware and software that enable the communication. In some cases, one device is a computer (sometimes called a "server") and the other is a simpler device (sometimes called a "client") enabled only with some means of input (such as a keyboard) and some means of output (such as a screen).

Although this model defines a basic network, the actual situation is frequently significantly more complicated.

- The simpler client device, employed for user-to-computer communication, is often a PC or workstation, so the client has considerable storage and processing capability.
- A network can be configured as just a single client connected to a single server. But more typically, many clients interact with many servers.
- The network's services are often provided by many computers. As a single user's communication travels back and forth from client to server, it may merely pass through some computers but pause at others for significant interactions.
- The end user is usually unaware of many of the communications and computations taking place in the network on the user's behalf.

A single computing system in a network is often called a node, and its processor (computer) is called a host. A connection between two hosts is known as a link. Network computing consists of users, communications media, visible hosts, and systems not generally visible to end users. Users communicate with networked systems by interacting directly with terminals, workstations, and computers. A workstation is an end-user computing device, usually designed for a single user at a time.

Networks can be described by several typical characteristics:

- **Anonymity:** You may have seen the cartoon image that shows a dog typing at a workstation, and saying to another dog, "On the Internet, nobody knows you're a dog." A network removes most of the clues, such as appearance, voice, or context, by which we

recognize acquaintances.

- **Automation:** In some networks, one or both endpoints, as well as all intermediate points, involved in each communication may be machines with only minimal human supervision.
- **Distance:** Many networks connect endpoints that are physically far apart. Although not all network connections involve distance, the speed of communication is fast enough that humans usually cannot tell whether a remote site is near or far.
- **Opacity:** Because the dimension of distance is hidden, users cannot tell whether a remote host is in the room next door or in a different country. In the same way, users cannot distinguish whether they are connected to a node in an office, school, home, or warehouse, or whether the node's computing system is large or small, modest or powerful. In fact, users cannot tell if the current communication involves the same host with which they communicated the last time.
- **Routing diversity:** To maintain or improve reliability and performance, routings between two endpoints are usually dynamic. That is, the same interaction may follow one path through the network the first time and a very different path the second time. In fact, a query may take a different path from the response that follows a few seconds later.
- **Boundary:** The boundary distinguishes an element of the network from an element outside it. For a simple network, we can easily list all the components and draw an imaginary line around it to separate what is in the network from what is outside. But listing all the hosts connected to the Internet is practically impossible. For example, a line surrounding the Internet would have to surround the entire globe today, and Internet connections also pass through satellites in orbit around the earth. Moreover, as people and organizations choose to be connected or not, the number and type of hosts change almost second by second, with the number generally increasing over time.
- **Ownership:** It is often difficult to know who owns each host in a network. The network administrator's organization may own the network infrastructure, including the cable and network devices. However, certain hosts may be connected to a network for convenience, not necessarily implying ownership.
- **Control:** Finally, if ownership is uncertain, control must be, too. To see how, pick an arbitrary host. Is it part of network A? If yes, is it under the control of network A's administrator? Does that administrator establish access control policies for the network, or determine when its software must be upgraded and to what version? Indeed, does the administrator even know what version

of software that host runs?

10.3.2 Media:

Communication is enabled by several kinds of media. We can choose among several types, such as along copper wires or optical fibre or through the air, as with cellular phones. Let us look at each type in turn.

Cable: Because much of our computer communication has historically been done over telephone lines, the most common network communication medium today is wire. Inside our homes and offices, we use a pair of insulated copper wires, called a twisted pair or unshielded twisted pair (UTP). Copper has good transmission properties at a relatively low cost. The bandwidth of UTP is limited to under 10 megabits per second (Mbps), so engineers cannot transmit many communications simultaneously on a single line. Moreover, the signal strength degrades as it travels through the copper wire, and it cannot travel long distances without a boost.

Another choice for network communication is coaxial (coax) cable, the kind used for cable television. Coax cable is constructed with a single wire surrounded by an insulation jacket. The jacket is itself surrounded by a braided or spiral-wound wire. The inner wire carries the signal, and the outer braid acts as a ground. The most widely used computer communication coax cable is Ethernet, carrying up to 100 Mbps over distances of up to 1500 feet. Coax cable also suffers from degradation of signal quality over distance. Repeaters (for digital signals) or amplifiers (for analog signals) can be spaced periodically along the cable to pick up the signal, amplify it, remove spurious signals called "noise," and retransmit it.

Optical Fibre : A newer form of cable is made of very thin strands of glass. Instead of carrying electrical energy, these fibres carry pulses of light. The bandwidth of optical fiber is up to 1000 Mbps, and the signal degrades less over fiber than over wire or coax; the fiber is good for a run of approximately 2.5 miles. Optical fiber involves less interference, less crossover between adjacent media, lower cost, and less weight than copper. Thus, optical fiber is generally a much better transmission medium than copper.

Wireless : Radio signals can also carry communications. Like pagers, wireless microphones, garage door openers, and portable telephones, wireless radio can be used in networks, following a protocol developed for short-range telecommunications, designated the 802.11 family of standards. The wireless medium is used for short distances; it is especially useful for networks in which the nodes are physically close together, such as in an office building or at home.

Many 802.11 devices are becoming available for home and office wireless networks.

Microwave : Microwave is a form of radio transmission especially well-suited for outdoor communication. Microwave has a channel capacity like coax cable; that is, it carries similar amounts of data. Its principal advantage is that the signal is strong from point of transmission to point of receipt. Therefore, microwave signals do not need to be regenerated with repeaters, as do signals on cable.

Infrared: Infrared communication carries signals for short distances (up to 9 miles) and requires a clear line of sight. Because it does not require cabling, it is convenient for portable objects, such as laptop computers and connections to peripherals. An infrared signal is difficult to intercept because it is a point-to-point signal. Because of line-of-sight requirements and limited distance, infrared is typically used in a protected space, such as an office, in which in-the-middle attacks would be difficult to conceal.

Satellite :Many communications, such as international telephone calls, must travel around the earth. In the early days of telephone technology, telephone companies ran huge cables along the ocean's bottom, enabling calls to travel from one continent to another. Today, we have other alternatives. The communication companies place satellites in orbits that are synchronized with the rotation of the earth (called geosynchronous orbits).

Table 10-1 Communication Media Strengths and Weakness

Medium	Strengths	Weaknesses
Wire	<ul style="list-style-type: none"> • Widely used • Inexpensive to buy, install, maintain 	<ul style="list-style-type: none"> • Susceptible to emanation • Susceptible to physical wiretapping
Optical fiber	<ul style="list-style-type: none"> • Immune to emanation • Difficult to wiretap 	<ul style="list-style-type: none"> • Potentially exposed at connection points
Microwave	<ul style="list-style-type: none"> • Strong signal, not seriously affected by weather 	<ul style="list-style-type: none"> • Exposed to interception along path of transmission • Requires line of sight location • Signal must be repeated approximately every 30 miles (50 kilometers)
Wireless (radio, WiFi)	<ul style="list-style-type: none"> • Widely available • Built into many computers 	<ul style="list-style-type: none"> • Signal degrades over distance; suitable for short range • Signal interceptable in circular pattern around transmitter
Satellite	<ul style="list-style-type: none"> • Strong, fast signal 	<ul style="list-style-type: none"> • Delay due to distance signal travels up and down • Signal exposed over wide area at receiving end

10.3.3 Protocols:

When we use a network, the communication media are usually transparent to us. The communication medium may change from one transmission to the next. This ambiguity is a positive feature of a network: its *independence*. That is, the communication is separated from the actual medium of communication. Independence is possible because we have defined protocols that allow a user to view the network at a high, abstract level of communication (viewing it in terms of user and data); the details of *how* the communication is accomplished are hidden within software and hardware at both ends. The software and hardware enable us to implement a network according to a protocol stack, a layered architecture for communications. Each layer in the stack is much like a language for communicating information relevant at that layer. Two popular protocol stacks are used frequently for implementing networks: The Open Systems Interconnection (OSI) and the Transmission Control Protocol and Internet Protocol (TCP/IP) architecture.

ISO OSI Reference Model:

The International Standards Organization (ISO) Open Systems Interconnection model consists of layers by which a network communication occurs. The OSI reference model contains the seven layers. This seven-layer model starts with an application that prepares data to be transmitted through a network. The data move down through the layers, being transformed and repackaged; at the lower layers, control information is added in headers and trailers. Finally, the data are ready to travel on a physical medium,

such as a cable or through the air on a microwave or satellite link. On the receiving end, the data enter the bottom of the model and progress up through the layers where control information is examined and removed, and the data are reformatted. Finally, the data arrive at an application at the top layer of the model for the receiver.

TCP/IP :

The OSI model is a conceptual one; it shows the different activities required for sending a communication. However, full implementation of a seven-layer transmission carries too much overhead for megabit-per-second communications; the OSI protocol slows things down to unacceptable levels. For this reason, TCP/IP (Transmission Control Protocol/Internet Protocol) is the protocol stack used for most wide area network communications. TCP/IP was invented for what became the Internet. TCP/IP is defined by protocols, not layers, but we can think of it in terms of four layers: application, host-to-host (end-to-end) transport, Internet, and physical. An application program deals only with abstract data items meaningful to the application user. Although TCP/IP is often used as a single acronym, it really denotes two different protocols: TCP implements a connected communications session on top of the more basic IP transport protocol. In fact, a third protocol, UDP (user datagram protocol) is also an essential part of the suite. The transport layer receives variable-length messages from the application layer; the transport layer breaks them down into units of manageable size, transferred in packets. The Internet layer transmits application layer packets in data grams, passing them to different physical connections based on the data's destination. The physical layer consists of device drivers to perform the actual bit-by-bit data communication.

10.3.4 Types of Network:

A network is a collection of communicating hosts. But to understand the network and how it works, we have several key questions to ask, such as How many hosts? Communicating by what means? To answer these questions, we are helped by an understanding of several types of subclasses of networks, since they commonly combine into larger networks. The subclasses are general notions, not definitive distinctions. But since the terms are commonly used, we present several common network subclasses that have significant security properties.

Local Area Networks:

As the name implies, a local area network (or LAN) covers a small distance, typically within a single building. Usually a LAN

connects several small computers, such as personal computers, as well as printers and perhaps some dedicated file storage devices. The primary advantage of a LAN is the opportunity for its users to share data and programs and to share access to devices such as printers.

Wide Area Networks:

A wide area network, or WAN, differs from a local area network in terms of both size or distance (as its name implies, it covers a wider geographic area than does a LAN) and control or ownership (it is more likely *not* to be owned or controlled by a single body). Still, there tends to be some unifying principle to a WAN. The hosts on a WAN may all belong to a company with many offices, perhaps even in different cities or countries, or they may be a cluster of independent organizations within a few miles of each other, who share the cost of networking hardware. These examples also show how WANs themselves differ. Some are under close control and maintain a high degree of logical and physical isolation (typically, these are WANs controlled by one organization), while others are only marriages of convenience.

Internet works (Internets) :

Networks of networks, or internet work networks, are sometimes called internets. An internet is a connection of two or more separate networks, in that they are separately managed and controlled. The most significant internet work is known as the Internet, because it connects so many of the other public networks. The Internet is, in fact, a federation of networks, loosely controlled by the Internet Society (ISOC) and the Internet Corporation for Assigned Names and Numbers (ICANN). These organizations enforce certain minimal rules of fair play to ensure that all users are treated equitably, and they support standard protocols so that users can communicate.

10.4 THREATS IN NETWORKS

This section describes some of the threats you have already hypothesized and perhaps presents you with some new ones. But the general thrust is the same: threats aimed to compromise confidentiality, integrity, or availability, applied against data, software, and hardware by nature, accidents, non-malicious humans, and malicious attackers.

10.4.1 What makes a network vulnerable:

An isolated home user or a stand-alone office with a few employees is an unlikely target for many attacks. But add a network to the mix and the risk rises sharply. Consider how a network differs from a stand-alone environment:

Anonymity: An attacker can mount an attack from thousands of miles away and never come into direct contact with the system, its administrators, or users. The potential attacker is thus safe behind an electronic shield. The attack can be passed through many other hosts to disguise the attack's origin. And computer-to-computer authentication is not the same for computers as it is for humans.

Many points of attack both targets and origins: A simple computing system is a self-contained unit. Access controls on one machine preserve the confidentiality of data on that processor. However, when a file is stored in a network host remote from the user, the data or the file itself may pass through many hosts to get to the user. One host's administrator may enforce rigorous security policies, but that administrator has no control over other hosts in the network. Thus, the user must depend on the access control mechanisms in each of these systems. An attack can come from any host to any host, so that a large network offers many points of vulnerability.

Sharing: Because networks enable resource and workload sharing, more users have the potential to access networked systems than on single computers. Perhaps worse, access is afforded to *more systems*, so that access controls for single systems may be inadequate in networks.

Complexity of system: A network combines two or more possibly dissimilar operating systems. Therefore, a network operating/control system is likely to be more complex than an operating system for a single computing system. Furthermore, the ordinary desktop computer today has greater computing power than did many office computers in the last two decades. The attacker can use this power to advantage by causing the victim's computer to perform part of the attack's computation. And because an average computer is so powerful, most users do not know what their computers are really doing at any moment: What processes are active in the background while you are playing *Invaders from Mars*? This complexity diminishes confidence in the network's security.

Unknown perimeter: A network's expandability also implies uncertainty about the network boundary. One host may be a node on two different networks, so resources on one network are accessible to the users of the other network as well. Although wide accessibility is an advantage, this unknown or uncontrolled group of possibly malicious users is a security disadvantage. A similar problem occurs when new hosts can be added to the network. Every network node must be able to react to the possible presence of new, untrustable hosts.

Unknown path: There may be many paths from one host to another. Suppose that a user on host A1 wants to send a message to a user on host B3. That message might be routed through hosts C or D before arriving at host B3. Host C may provide acceptable security, but not D. Network users seldom have control over the routing of their messages.

10.4.2 Who attacks networks:

Who are the attackers? To have some idea of who the attackers might be, we return to concepts introduced in earlier chapter, where we described the three necessary components of an attack: method, opportunity, and motive. Here we consider first the motives of attackers. Focusing on motive may give us some idea of who might attack a networked host or user. Four important motives are challenge or power, fame, money, and ideology.

Challenge:

Why do people do dangerous or daunting things, like climb mountains or swim the English Channel or engage in extreme sports? Because of the challenge. The situation is no different for someone skilled in writing or using programs. The single most significant motivation for a network attacker is the intellectual challenge. He or she is intrigued with knowing the answers to Can I defeat this network? What would happen if I tried this approach or that technique? Some attackers enjoy the intellectual stimulation of defeating the supposedly undefeatable. For example, Robert Morris, who perpetrated the Internet worm in 1988, attacked supposedly as an experiment to see if he could exploit a vulnerability. Other attackers, such as the Cult of the Dead Cow, seek to demonstrate weaknesses in security defences so that others will pay attention to strengthening security. Still other attackers are unnamed, unknown individuals working persistently just to see how far they can go in performing unwelcome activities. However, only a few attackers find previously unknown flaws. Most attackers repeat well-known and even well-documented attacks, sometimes only to see if they work against different hosts.

Fame:

The challenge of accomplishment is enough for some attackers. But other attackers seek recognition for their activities. That is, part of the challenge is doing the deed; another part is taking credit for it. In many cases, we do not know who the attackers really are, but they leave behind a "calling card" with a name or moniker: Mafia boy, Kevin Mitnick, Fluffy Bunny, and members of the Chaos Computer Club, for example. The actors often retain some anonymity by using pseudonyms, but they achieve fame, nevertheless. They may not be able to brag too openly, but they enjoy the personal thrill of seeing their attacks written up in the news media.

Money and Espionage:

As in other settings, financial reward motivates attackers, too. Some attackers perform industrial espionage, seeking information on a company's products, clients, or long-range plans. We know industrial espionage has a role when we read about laptops and sensitive papers having been lifted from hotel rooms when other more valuable items were left behind. Some countries are notorious for using espionage to aid their state-run industries. Industrial espionage, leading to loss of intellectual property, is clearly a problem.

Organized Crime:

With the growth in commercial value of the Internet, participation by organized crime has also increased. In October 2004, police arrested members of a 28-person gang of Internet criminals, called the Shadow crew, who operated out of six foreign countries and eight states in the United States. Six leaders of that group pled guilty to charges, closing an illicit business that trafficked in at least 1.5 million stolen credit and bank card numbers and resulted in losses in excess of \$4 million. These more sophisticated attacks require more than one person working out of a bedroom, and so organization and individual responsibilities follow. With potential revenue in the millions of dollars and operations involving hundreds of thousands of credit card numbers and other pieces of identity, existing organized crime units are sure to take notice.

Ideology:

In the last few years, we are starting to find cases in which attacks are perpetrated to advance ideological ends. For example, many security analysts believe that the Code Red worm of 2001 was launched by a group motivated by the tension in U.S. China relations. Denning has distinguished between two types of related behaviours, hacktivism and cyberterrorism. Hacktivism involves "operations that use hacking techniques against a target's [network] with the intent of disrupting normal operations but not causing serious damage." In some cases, the hacking is seen as giving voice to a constituency that might otherwise not be heard by the company or government organization. Cyberterrorism is more dangerous than hacktivism: "politically motivated hacking operations intended to cause grave harm such as loss of life or severe economic damage." Security and terrorism experts are seeing increasing use of the Internet as an attack vector, as a communications medium among attackers, and as a point of attack.

10.4.3 Reconnaissance:

Now that we have listed many motives for attacking, we turn to how attackers perpetrate their attacks. Attackers do not ordinarily

sit down at a terminal and launch an attack. A clever attacker investigates and plans before acting. A network attacker learns a lot about a potential target before beginning the attack. We study the precursors to an attack so that if we can recognize characteristic behaviour, we may be able to block the attack before it is launched. Because most vulnerable networks are connected to the Internet, the attacker begins preparation by finding out as much as possible about the target.

Port Scan:

An easy way to gather network information is to use a port scan, a program that, for an IP address, reports which ports respond to messages and which of several known vulnerabilities seem to be present. Port scanning tells an attacker three things: which standard ports or services are running and responding on the target system, what operating system is installed on the target system, and what applications and versions of applications are present. This information is readily available for the asking from a networked system; it can be obtained quietly, anonymously, without identification or authentication, drawing little or no attention to the scan.

Social Engineering:

The port scan gives an external picture of a network, where are the doors and windows, of what are they constructed, to what kinds of rooms do they open? The attacker also wants to know what is inside the building. What better way to find out than to ask? Suppose while sitting at your workstation, you receive a phone call. "Hello, this is John Davis from IT support. We need to test some connections on the internal network. Could you please run the command `ipconfig/all` on your workstation and read to me the addresses it displays?" The request sounds innocuous. But unless you know John Davis and his job responsibilities well, the caller could be an attacker gathering information on the inside architecture. Social engineering involves using social skills and personal interaction to get someone to reveal security-relevant information and perhaps even to do something that permits an attack. The point of social engineering is to persuade the victim to be helpful. The attacker often impersonates someone inside the organization who is in a bind: "My laptop has just been stolen and I need to change the password I had stored on it," or "I have to get out a very important report quickly and I can't get access to the following thing." This attack works especially well if the attacker impersonates someone in a high position, such as the division vice president or the head of IT security. (Their names can sometimes be found on a public web site, in a network registration with the Internet registry, or in publicity and articles.) The attack is often directed at someone low enough to be intimidated or impressed by the high-level person. A direct phone call and expressions of great urgency can override any instinct to check

out the story. Because the victim has helped the attacker, the victim will think nothing is wrong and not report the incident. Thus, the damage may not be known for some time. An attacker has little to lose in trying a social engineering attack. At worst it will raise awareness of a possible target. But if the social engineering is directed against someone who is not sceptical, especially someone not involved in security management, it may well succeed.

Intelligence:

From a port scan the attacker knows what is open. From social engineering, the attacker knows certain internal details. But a more detailed floor plan would be nice. Intelligence is the general term for collecting information. In security it often refers to gathering discrete bits of information from various sources and then putting them together like the pieces of a puzzle. One commonly used intelligence technique is called "dumpster diving." It involves looking through items that have been discarded in rubbish bins or recycling boxes. It is amazing what we throw away without thinking about it. Mixed with the remains from lunch might be network diagrams, printouts of security device configurations, system designs and source code, telephone and employee lists, and more. Even outdated printouts may be useful. Seldom will the configuration of a security device change completely. More often only one rule is added or deleted or modified, so an attacker has a high probability of a successful attack based on the old information.

Gathering intelligence may also involve eavesdropping. Trained spies may follow employees to lunch and listen in from nearby tables as co-workers discuss security matters. Or spies may befriend key personnel to co-opt, coerce, or trick them into passing on useful information. Most intelligence techniques require little training and minimal investment of time. If an attacker has targeted an organization, spending a little time to collect background information yields a big payoff.

Operating System and Application Fingerprinting:

The port scan supplies the attacker with very specific information. For instance, an attacker can use a port scan to find out that port 80 is open and supports HTTP, the protocol for transmitting web pages. But the attacker is likely to have many related questions, such as which commercial server application is running, what version, and what the underlying operating system and version are. Once armed with this additional information, the attacker can consult a list of specific software's known vulnerabilities to determine which weaknesses to try to exploit. How can the attacker answer these questions? The network protocols are standard and vendor independent. Still, each vendor's code is implemented independently, so there may be minor variations in interpretation and behaviour. The variations do not make the software noncompliant

with the standard, but they are different enough to make each version distinctive. A new version will implement a new feature, but an old version will reject the request. All these peculiarities, sometimes called the operating system or application fingerprint, can mark the manufacturer and version.

Bulletin Boards and Chats:

The Internet is probably the greatest tool for sharing knowledge since the invention of the printing press. It is probably also the most dangerous tool for sharing knowledge. Numerous underground bulletin boards and chat rooms support exchange of information. Attackers can post their latest exploits and techniques, read what others have done, and search for additional information on systems, applications, or sites. Remember that, as with everything on the Internet, anyone can post anything, so there is no guarantee that the information is reliable or accurate. And you never know who is reading from the Internet.

Availability of Documentation:

The vendors themselves sometimes distribute information that is useful to an attacker. For example, Microsoft produces a resource kit by which application vendors can investigate a Microsoft product to develop compatible, complementary applications. This toolkit also gives attackers tools to use in investigating a product that can subsequently be the target of an attack.

10.4.4 Threats in Transit: Eavesdropping and Wiretapping:

By now, you can see that an attacker can gather a significant amount of information about a victim before beginning the actual attack. Once the planning is done, the attacker is ready to proceed. The easiest way to attack is simply to listen in. An attacker can pick off the content of a communication passing in the clear. The term eavesdrop implies overhearing without expending any extra effort. For example, we might say that an attacker (or a system administrator) is eavesdropping by monitoring all traffic passing through a node. The administrator might have a legitimate purpose, such as watching for inappropriate use of resources (for instance, visiting non-work-related web sites from a company network) or communication with inappropriate parties (for instance, passing files to an enemy from a military computer).

A more hostile term is wiretap, which means intercepting communications through some effort. Passive wiretapping is just "listening," much like eavesdropping. But active wiretapping means injecting something into the communication. Wiretapping works differently depending on the communication medium used.

Cable:

At the most local level, all signals in an Ethernet or other LAN are available on the cable for anyone to intercept. Each LAN connector (such as a computer board) has a unique address; each board and its drivers are programmed to label all packets from its host with its unique address (as a sender's "return address") and to take from the net only those packets addressed to its host. But removing only those packets addressed to a given host is mostly a matter of politeness; there is little to stop a program from examining each packet as it goes by. A device called a packet sniffer can retrieve all packets on the LAN. Alternatively, one of the interface cards can be reprogrammed to have the supposedly unique address of another existing card on the LAN so that two different cards will both fetch packets for one address. (To avoid detection, the rogue card will have to put back on the net copies of the packets it has intercepted.) Fortunately (for now), LANs are usually used only in environments that are friendly, so these kinds of attacks occur infrequently.

Wireless:

Wireless networking is becoming very popular, with good reason. With wireless (also known as Wi-Fi), people are not tied to a wired connection; they are free to roam throughout an office, house, or building while maintaining a connection. Universities, offices, and even home users like being able to connect to a network without the cost, difficulty, and inconvenience of running wires. The difficulties of wireless arise in the ability of intruders to intercept and spoof a connection. But the major threat is not interference; it is interception. A wireless signal is strong for approximately 100 to 200 feet. A strong signal can be picked up easily. And with an inexpensive, tuned antenna, a wireless signal can be picked up several miles away. In other words, someone who wanted to pick up your signal could do so from several streets away. Parked in a truck or van, the interceptor could monitor your communications for quite some time without arousing suspicion.

Interception:

Interception of wireless traffic is always a threat, through either passive or active wiretapping. You may react to that threat by assuming that encryption will address it. Unfortunately, encryption is not always used for wireless communication, and the encryption built into some wireless devices is not as strong as it should be to deter a dedicated attacker.

Theft of Service:

Wireless also admits a second problem: the possibility of rogue use of a network connection. Many hosts run the Dynamic Host Configuration Protocol (DHCP), by which a client negotiates a one-time IP address and connectivity with a host. This protocol is useful in office or campus settings, where not all users (clients) are

active at any time. A small number of IP addresses can be shared among users. Essentially the addresses are available in a pool. A new client requests a connection and an IP address through DHCP, and the server assigns one from the pool. This scheme admits a big problem with authentication. Unless the host authenticates users before assigning a connection, any requesting client is assigned an IP address and network access.

10.4.5 Protocol Flaws:

Internet protocols are publicly posted for scrutiny by the entire Internet community. Each accepted protocol is known by its Request for Comment (RFC) number. Many problems with protocols have been identified by sharp reviewers and corrected before the protocol was established as a standard. But protocol definitions are made and reviewed by fallible humans. Likewise, protocols are implemented by fallible humans. For example, TCP connections are established through sequence numbers. The client (initiator) sends a sequence number to open a connection, the server responds with that number and a sequence number of its own, and the client responds with the server's sequence number. Suppose someone can guess a client's next sequence number. That person could impersonate the client in an interchange. Sequence numbers are incremented regularly, so it can be easy to predict the next number.

10.4.6 Impersonation:

In many instances, there is an easier way than wiretapping for obtaining information on a network: Impersonate another person or process. Why risk tapping a line, or why bother extracting one communication out of many, if you can obtain the same data directly? Impersonation is a more significant threat in a wide area network than in a local one. Local individuals often have better ways to obtain access as another user; they can, for example, simply sit at an unattended workstation. Still, impersonation attacks should not be ignored even on local area networks, because local area networks are sometimes attached to wider area networks without anyone's first thinking through the security implications. In an impersonation, an attacker has several choices:

- Guess the identity and authentication details of the target.
- Pick up the identity and authentication details of the target from a previous communication or from wiretapping.
- Circumvent or disable the authentication mechanism at the target computer.
- Use a target that will not be authenticated.
- Use a target whose authentication data are known.

Authentication Foiled by Guessing:

The results of several studies showing that many users choose easy-to-guess passwords. The Internet worm of 1988 capitalized on exactly that flaw. Morris's worm tried to impersonate each user on a target machine by trying, in order, a handful of variations of the username, a list of about 250 common passwords and, finally, the words in a dictionary. Sadly, many users' accounts are still open to these easy attacks. A second source of password guesses is default passwords. Many systems are initially configured with default accounts having GUEST or ADMIN as login IDs; accompanying these IDs are well-known passwords such as "guest" or "null" or "password" to enable the administrator to set up the system. Administrators often forget to delete or disable these accounts, or at least to change the passwords. Dead accounts offer a final source of guessable passwords. The attacker can try several passwords until the password guessing limit is exceeded. The system then locks the account administratively, and the attacker uses a social engineering attack. In all these ways the attacker may succeed in resetting or discovering a password.

Authentication Thwarted by Eavesdropping or Wiretapping:

Because of the rise in distributed and client-server computing, some users have access privileges on several connected machines. To protect against arbitrary outsiders using these accesses, authentication is required between hosts. This access can involve the user directly, or it can be done automatically on behalf of the user through a host-to-host authentication protocol. In either case, the account and authentication details of the subject are passed to the destination host. When these details are passed on the network, they are exposed to anyone observing the communication on the network. These same authentication details can be reused by an impersonator until they are changed.

Authentication Foiled by Avoidance:

Obviously, authentication is effective only when it works. A weak or flawed authentication allows access to any system or person who can circumvent the authentication. Many network hosts, especially those that connect to wide area networks, run variants of Unix System V or BSD Unix. In a local environment, many users are not aware of which networked operating system is in use; still fewer would know of, be capable of, or be interested in exploiting flaws. However, some hackers regularly scan wide area networks for hosts running weak or flawed operating systems. Thus, connection to a wide area network, especially the Internet, exposes these flaws to a wide audience intent on exploiting them.

Non-existent Authentication:

If two computers are used by the same users to store data and run processes and if each has authenticated its users on first

access, you might assume that computer-to-computer or local user-to-remote process authentication is unnecessary. These two computers and their users are a trustworthy environment in which the added complexity of repeated authentication seems excessive. These "trusted hosts" can also be exploited by outsiders who obtain access to one system through an authentication weakness (such as a guessed password) and then transfer to another system that accepts the authenticity of a user who comes from a system on its trusted list. An attacker may also realize that a system has some identities requiring no authentication. Some systems have "guest" or "anonymous" accounts to allow outsiders to access things the systems want to release to anyone.

Well-Known Authentication:

Authentication data should be unique and difficult to guess. But unfortunately, the convenience of one well known authentication scheme sometimes usurps the protection. The system network management protocol (SNMP) is widely used for remote management of network devices, such as routers and switches, that support no ordinary users. SNMP uses a "community string," essentially a password for the community of devices that can interact with one another. But network devices are designed especially for quick installation with minimal configuration, and many network administrators do not change the default community string installed on a router or switch. This laxity makes these devices on the network perimeter open to many SNMP attacks. Some vendors still ship computers with one system administration account installed, having a default password. Or the systems come with a demonstration or test account, with no required password. Some administrators fail to change the passwords or delete these accounts.

Trusted Authentication:

Finally, authentication can become a problem when identification is delegated to other trusted sources. For instance, a file may indicate who can be trusted on a host. Or the authentication mechanism for one system can "vouch for" a user. We noted earlier how the Unix *.rhosts*, *.rlogin*, and */etc/hosts/equiv* files indicate hosts or users that are trusted on other hosts. While these features are useful to users who have accounts on multiple machines or for network management, maintenance, and operation, they must be used very carefully. Each of them represents a potential hole through which a remote user or a remote attacker can achieve access.

Spoofing:

Guessing or otherwise obtaining the network authentication credentials of an entity (a user, an account, a process, a node, a device) permits an attacker to create a full communication under the entity's identity. Impersonation falsely represents a valid entity in a communication. Closely related is spoofing, when an attacker falsely

carries on one end of a networked interchange. Examples of spoofing are masquerading, session hijacking, and man-in-the-middle attacks.

Masquerade:

In a masquerade one host pretends to be another. A common example is URL confusion. Domain names can easily be confused, or someone can easily mistype certain names. Thus xyz.com, xyz.org, and xyz.net might be three different organizations, or one bona fide organization (for example, xyz.com) and two masquerade attempts from someone who registered the similar domain names. From the attacker's point of view, the fun in masquerading comes before the mask is removed.

A variation of this attack is called phishing. You send an e-mail message, perhaps with the real logo of Blue Bank, and an enticement to click on a link, supposedly to take the victim to the Blue Bank web site. The enticement might be that your victim's account has been suspended or that you offer your victim some money for answering a survey (and need the account number and PIN to be able to credit the money), or some other legitimate-sounding explanation.

In another version of a masquerade, the attacker exploits a flaw in the victim's web server and can overwrite the victim's web pages. Although there is some public humiliation at having one's site replaced, perhaps with obscenities or strong messages opposing the nature of the site (for example, a plea for vegetarianism on a slaughterhouse web site), most people would not be fooled by a site displaying a message absolutely contrary to its aims. However, a clever attacker can be more subtle. Instead of differentiating from the real site, the attacker can try to build a false site that resembles the real one, perhaps to obtain sensitive information (names, authentication numbers, credit card numbers) or to induce the user to enter into a real transaction.

Session Hijacking:

Session hijacking is intercepting and carrying on a session begun by another entity. Suppose two entities have entered a session but then a third entity intercepts the traffic and carries on the session in the name of the other. A different type of example involves an interactive session, for example, using Telnet. If a system administrator logs in remotely to a privileged account, a session hijack utility could intrude in the communication and pass commands as if they came from the administrator.

Man-in-the-Middle Attack:

Our hijacking example requires a third party involved in a session between two entities. A man-in-the-middle attack is a similar

form of attack, in which one entity intrudes between two others. The difference between man-in-the-middle and hijacking is that a man-in-the-middle usually participates from the start of the session, whereas a session hijacking occurs after a session has been established. The difference is largely semantic and not too significant.

10.4.7 Message Confidentiality Threats:

An attacker can easily violate message confidentiality (and perhaps integrity) because of the public nature of networks. Eavesdropping and impersonation attacks can lead to a confidentiality or integrity failure. Here we consider several other vulnerabilities that can affect confidentiality.

Mis-delivery:

Sometimes messages are misdelivered because of some flaw in the network hardware or software. Most frequently, messages are lost entirely, which is an integrity or availability issue. Occasionally, however, a destination address is modified or some handler malfunctions, causing a message to be delivered to someone other than the intended recipient. All of these "random" events are quite uncommon. More frequent than network flaws are human errors. It is far too easy to mistype an address. There is simply no justification for a computer network administrator to identify people by meaningless long numbers or cryptic initials.

Exposure:

To protect the confidentiality of a message, we must track it all the way from its creation to its disposal. Along the way, the content of a message may be exposed in temporary buffers; at switches, routers, gateways, and intermediate hosts throughout the network; and in the workspaces of processes that build, format, and present the message. In earlier chapters, we considered confidentiality exposures in programs and operating systems. All these exposures apply to networked environments as well. Furthermore, a malicious attacker can use any of these exposures as part of a general or focused attack on message confidentiality. Passive wiretapping is one source of message exposure. So, also is subversion of the structure by which a communication is routed to its destination. Finally, intercepting the message at its source, destination, or at any intermediate node can lead to its exposure.

Traffic Flow Analysis:

Sometimes not only is the message itself sensitive but the fact that a message *exists* is also sensitive. For example, if the enemy during wartime sees a large amount of network traffic between headquarters and a unit, the enemy may be able to infer that significant action is being planned to involve that unit. In a commercial setting, messages sent from the president of one

company to the president of a competitor could lead to speculation about a takeover or conspiracy to fix prices. Or communications from the prime minister of one country to another with whom diplomatic relations were suspended could lead to inferences about a rapprochement between the countries. In these cases, we need to protect both the *content* of messages and the *header* information that identifies sender and receiver.

10.4.8 Message Integrity Threats:

In many cases, the *integrity* or correctness of a communication is at least as important as its confidentiality. In fact, for some situations, such as passing authentication data, the integrity of the communication is paramount. In other cases, the need for integrity is less obvious. Next, we consider threats based on failures of integrity in communication.

Falsification of Messages:

Increasingly, people depend on electronic messages to justify and direct actions. For example, if you receive a message from a good friend asking you to meet at the pub for a drink next Tuesday evening, you will probably be there at the appointed time. If it is reasonable, we tend to act on an electronic message just as we would on a signed letter, a telephone call, or a face-to-face communication. However, an attacker can take advantage of our trust in messages to mislead us. An attacker may

- change some or all the content of a message
- replace a message entirely, including the date, time, and sender/receiver identification
- reuse (replay) an old message
- combine pieces of different messages into one
- change the apparent source of a message
- redirect a message
- destroy or delete a message

These attacks can be perpetrated in the ways we have already examined, including

- active wiretap
- Trojan horse
- Impersonation
- pre-empted host
- pre-empted workstation

Noise:

Signals sent over communications media are subject to interference from other traffic on the same media, as well as from natural sources, such as lightning, electric motors, and animals. Such unintentional interference is called noise. These forms of noise are inevitable, and they can threaten the integrity of data in a message. Fortunately, communications protocols have been intentionally designed to overcome the negative effects of noise. For example, the TCP/IP protocol suite ensures detection of almost all transmission errors. Processes in the communications stack detect errors and arrange for retransmission, all invisible to the higher-level applications. Thus, noise is scarcely a consideration for users in security-critical applications.

10.4.9 Format Failures:

Network communications work because of well-designed protocols that define how two computers communicate with a minimum of human intervention. The format of a message, size of a data unit, sequence of interactions, even the meaning of a single bit is precisely described in a standard. The whole network works only because everyone obeys these rules. Almost everyone, that is. Attackers purposely break the rules to see what will happen. Or the attacker may seek to exploit an undefined condition in the standard. Software may detect the violation of structure and raise an error indicator. Sometimes, however, the malformation causes a software failure, which can lead to a security compromise, just what the attacker wants. In this section we look at several kinds of malformation.

Malformed Packets:

Packets and other data items have specific formats, depending on their use. Field sizes, bits to signal continuations, and other flags have defined meanings and will be processed appropriately by network service applications called protocol handlers. These services do not necessarily check for errors, however. For example, in 2003 Microsoft distributed a patch for its RPC (Remote Procedure Call) service. If a malicious user initiated an RPC session and then sent an incorrectly formatted packet, the entire RPC service failed, as well as some other Microsoft services. Attackers try all sorts of malformations of packets. Of course, many times the protocol handler detects the malformation and raises an error condition, and other times the failure affects only the user (the attacker). But when the error causes the protocol handler to fail, the result can be denial of service, complete failure of the system, or some other serious result.

Protocol Failures and Implementation Flaws:

Each protocol is a specification of a service to be provided;

the service is then implemented in software, which may be flawed. Network protocol software is basic to the operating system, so flaws in that software can cause widespread harm because of the privileges with which the software runs and the impact of the software on many users at once. Certain network protocol implementations have been the source of many security flaws; especially troublesome have been SNMP (network management), DNS (addressing service), and e-mail services such as SMTP and S/MIME. Although different vendors have implemented the code for these services themselves, they often are based on a common (flawed) prototype. Or the protocol itself may be incomplete. If the protocol does not specify what action to take in a situation, vendors may produce different results. So, an interaction on Windows, for example, might succeed while the same interaction on a Unix system would fail. The protocol may have an unknown security flaw. Attackers can exploit all these kinds of errors.

10.4.10 Website Vulnerabilities:

A web site is especially vulnerable because it is almost completely exposed to the user. If you use an application program, you do not usually get to view the program's code. With a web site, the attacker can download the site's code for offline study over time. With a program, you have little ability to control in what order you access parts of the program, but a web attacker gets to control in what order pages are accessed. The attacker can also choose what data to supply and can run experiments with different data values to see how the site will react. In short, the attacker has some advantages that can be challenging to control.

Web Site Defacement:

One of the most widely known attacks is the web site defacement attack. Because of the large number of sites that have been defaced and the visibility of the result, the attacks are often reported in the popular press. A defacement is common not only because of its visibility but also because of the ease with which one can be done. Web sites are designed so that their code is downloaded, enabling an attacker to obtain the full hypertext document and all programs directed to the client in the loading process. An attacker can even view programmers' comments left in as they built or maintained the code. The download process essentially gives the attacker the blueprints to the web site.

Buffer Overflows:

Buffer overflow is alive and well on web pages, too. The attacker simply feeds a program far more data than it expects to receive. A buffer size is exceeded, and the excess data spill over into adjoining code and data locations. Perhaps the best-known web server buffer overflow is the file name problem known as iishack. This

attack is so well known that it has been written into a procedure (see <http://www.technotronic.com>). To execute the procedure, an attacker supplies as parameters the site to be attacked and the URL of a program the attacker wants that server to execute.

Dot-Dot-Slash:

Web server code should always run in a constrained environment. Ideally, the web server should never have editors, xterm and Telnet programs, or even most system utilities loaded. By constraining the environment in this way, even if an attacker escapes from the web server application, no other executable programs will help the attacker use the web server's computer and operating system to extend the attack. The code and data for web applications can be transferred manually to a web server or pushed as a raw image. But many web applications programmers are naïve. They expect to need to edit a web application in place, so they install editors and system utilities on the server to give them a complete environment in which to program.

A second, less desirable, condition for preventing an attack is to create a fence confining the web server application. With such a fence, the server application cannot escape from its area and access other potentially dangerous system areas (such as editors and utilities). The server begins in a directory subtree, and everything the server needs is in that same subtree.

Enter the dot-dot. In both Unix and Windows, '..' is the directory indicator for "predecessor." And '../..' is the grandparent of the current location. So, someone who can enter file names can travel back up the directory tree one .. at a time. Cerberus Information Security analysts found just that vulnerability in the webhits.dll extension for the Microsoft Index Server.

Application Code Errors:

A user's browser carries on an intricate, undocumented protocol interchange with applications on the web server. To make its job easier, the web server passes context strings to the user, making the user's browser reply with full context. A problem arises when the user can modify that context. An example is the time-of-check to time-of-use flaw that we discussed in earlier chapters. The server sets (checks) the price of the item when you first display the price, but then it loses control of the checked data item and never checks it again. This situation arises frequently in server application code because application programmers are generally not aware of security and typically do not anticipate malicious behaviour.

Server-Side Include:

A potentially more serious problem is called a server-side include. The problem takes advantage of the fact that web pages can be organized to invoke a function automatically. For example, many

pages use web commands to send an e-mail message in the "contact us" part of the displayed page. The commands, such as `e-mail`, `if`, `goto`, and `include`, are placed in a field that is interpreted in HTML. One of the server-side include commands is `exec`, to execute an arbitrary file on the server. For instance, the server-side include command opens a Telnet session from the server running in the name of the server. An attacker may find it interesting to execute commands such as `chmod` (change access rights to an object), `sh` (establish a command shell), or `cat` (copy to a file).

10.4.11 Denial of Service:

Availability attacks, sometimes called denial-of-service or DOS attacks, are much more significant in networks than in other contexts. There are many accidental and malicious threats to availability or continued service.

Transmission Failure:

Communications fail for many reasons. For instance, a line is cut. Or network noise makes a packet unrecognizable or undeliverable. A machine along the transmission path fails for hardware or software reasons. A device is removed from service for repair or testing. A device is saturated and rejects incoming data until it can clear its overload. Many of these problems are temporary or automatically fixed (circumvented) in major networks, including the Internet. However, some failures cannot be easily repaired. From a malicious standpoint, you can see that anyone who can sever, interrupt, or overload capacity to you can deny you service. The physical threats are obvious. We consider instead several electronic attacks that can cause a denial of service.

Connection Flooding:

The most primitive denial-of-service attack is flooding a connection. If an attacker sends you as much data as your communications system can handle, you are prevented from receiving any other data. Even if an occasional packet reaches you from someone else, communication to you will be seriously degraded. More sophisticated attacks use elements of Internet protocols. In addition to TCP and UDP, there is a third class of protocols, called ICMP or Internet Control Message Protocols. Normally used for system diagnostics, these protocols do not have associated user applications.

Echo-Chargen:

This attack works between two hosts. Chargen is a protocol that generates a stream of packets; it is used to test the network's capacity. The attacker sets up a chargen process on host A that generates its packets as echo packets with a destination of host B.

Then, host A produces a stream of packets to which host B replies by echoing them back to host A. This series puts the network infrastructures of A and B into an endless loop. If the attacker makes B both the source and destination address of the first packet, B hangs in a loop, constantly creating and replying to its own messages.

Ping of Death:

A ping of death is a simple attack. Since ping requires the recipient to respond to the ping request, all the attacker needs to do is send a flood of pings to the intended victim. The attack is limited by the smallest bandwidth on the attack route. The attacker cannot mathematically flood the victim alone. But the attack succeeds if the numbers are reversed. The ping packets will saturate the victim's bandwidth.

Smurf:

The smurf attack is a variation of a ping attack. It uses the same vehicle, a ping packet, with two extra twists. First, the attacker chooses a network of unwitting victims. The attacker spoofs the source address in the ping packet so that it appears to come from the victim. Then, the attacker sends this request to the network in broadcast mode by setting the last byte of the address to all 1s; broadcast mode packets are distributed to all hosts on the network.

Syn Flood:

Another popular denial-of-service attack is the syn flood. This attack uses the TCP protocol suite, making the session-oriented nature of these protocols work against the victim. The attacker can deny service to the target by sending many SYN requests and never responding with ACKs, thereby filling the victim's SYN_RECV queue.

Teardrop:

The teardrop attack misuses a feature designed to improve network communication. In the teardrop attack, the attacker sends a series of datagrams that cannot fit together properly. One datagram might say it is position 0 for length 60 bytes, another position 30 for 90 bytes, and another position 41 for 173 bytes. These three pieces overlap, so they cannot be reassembled properly. In an extreme case, the operating system locks up with these partial data units it cannot reassemble, thus leading to denial of service.

Traffic Redirection:

A router is a device that forwards traffic on its way through intermediate networks between a source host's network and a destination's network. So, if an attacker can corrupt the routing, traffic can disappear. To see how, keep in mind that, despite its sophistication, a router is simply a computer with two or more network interfaces. Suppose a router advertises to its neighbours that it has the best path to every other address in the whole network. Soon all routers will direct all traffic to that one router. The one router

may become flooded, or it may simply drop much of its traffic. In either case, a lot of traffic never makes it to the intended destination.

DNS Attacks:

Our final denial-of-service attack is a class of attacks based on the concept of domain name server. By overtaking a name server or causing it to cache spurious entries (called DNS cache poisoning), an attacker can redirect the routing of any traffic, with an obvious implication for denial of service.

10.4.12 Distributed Denial of Service:

The denial-of-service attacks we have listed are powerful by themselves, but an attacker can construct a two-stage attack that multiplies the effect many times. This multiplicative effect gives power to distributed denial of service. To perpetrate a distributed denial-of-service (or DDoS) attack, an attacker does two things. In the first stage, the attacker uses any convenient attack to plant a Trojan horse on a target machine. That Trojan horse does not necessarily cause any harm to the target machine, so it may not be noticed. The attacker repeats this process with many targets. Each of these target systems then becomes what is known as a zombie. The target systems carry out their normal work, unaware of the resident zombie. At some point the attacker chooses a victim and sends a signal to all the zombies to launch the attack. Then, instead of the victim's trying to defend against one denial-of-service attack from one malicious host, the victim must try to counter n attacks from the n zombies all acting at once.

10.4.13 Threats in Active or Mobile Code:

Active code or mobile code is a general name for code that is pushed to the client for execution. There are many kinds of active code.

Cookies:

Strictly speaking, cookies are not active code. They are data files that can be stored and fetched by a remote server. However, cookies can be used to cause unexpected data transfer from a client to a server, so they have a role in a loss of confidentiality. So, a cookie is something that takes up space on your disk, holding information about you that you cannot see, forwarded to servers you do not know whenever the server wants it, without informing you. The philosophy behind cookies seems to be "Trust us, it's good for you."

Scripts:

Clients can invoke services by executing scripts on servers. Typically, a web browser displays a page. As the user interacts with the web site via the browser, the browser organizes user inputs into parameters to a defined script; it then sends the script and

parameters to a server to be executed. But all communication is done through HTML. The server cannot distinguish between commands generated from a user at a browser completing a web page and a user's handcrafting a set of orders. The malicious user can monitor the communication between a browser and a server to see how changing a web page entry affects what the browser sends and then how the server reacts. With this knowledge, the malicious user can manipulate the server's actions.

Active Code:

Displaying web pages started simply with a few steps: generate text, insert images, and register mouse clicks to fetch new pages. Soon, people wanted more elaborate action at their web sites: toddlers dancing atop the page, a three-dimensional rotating cube, images flashing on and off, colours changing, totals appearing. Some of these tricks, especially those involving movement, take significant computing power; they require a lot of time and communication to download from a server. But typically, the client has a capable and underutilized processor, so the timing issues are irrelevant. To take advantage of the processor's power, the server may download code to be executed on the client. This executable code is called active code. The two main kinds of active code are Java code and ActiveX controls.

A hostile applet is downloadable Java code that can cause harm on the client's system. Because an applet is not screened for safety when it is downloaded and because it typically runs with the privileges of its invoking user, a hostile applet can cause serious damage. Using ActiveX controls, objects of arbitrary type can be downloaded to a client. If the client has a viewer or handler for the object's type, that viewer is invoked to present the object. To prevent arbitrary downloads, Microsoft uses an authentication scheme under which downloaded code is cryptographically signed and the signature is verified before execution. But the authentication verifies only the source of the code, not its correctness or safety.

Auto Exec by Type:

Data files are processed by programs. For some products, the file type is implied by the file extension, such as .doc for a Word document, .pdf (Portable Document Format) for an Adobe Acrobat file, or .exe for an executable file. On many systems, when a file arrives with one of these extensions, the operating system automatically invokes the appropriate processor to handle it. A malicious agent might send you a file named innocuous.doc, which you would expect to be a Word document. Because of the .doc extension, Word would try to open it. An attacker can disguise a malicious active file under a nonobvious file type.

Bots:

Bots, hackerese for robots, are pieces of malicious code under remote control. These code objects are Trojan horses that are distributed to large numbers of victims' machines. Because they may not interfere with or harm a user's computer (other than consuming computing and network resources), they are often undetected. Structured as a loosely coordinated web, a network of bots, called a botnet, is not subject to failure of any one bot or group of bots, and with multiple channels for communication and coordination, they are highly resilient. Botnets are used for distributed denial-of-service attacks, launching attacks from many sites in parallel against a victim. They are also used for spam and other bulk email attacks, in which an extremely large volume of e-mail from any one point might be blocked by the sending service provider.

10.4.14 Complex Attacks:

As if these vulnerabilities were not enough, two other phenomena multiply the risk. Scripts let people perform attacks even if the attackers do not understand what the attack is or how it is performed. Building blocks let people combine components of an attack, almost like building a house from prefabricated parts.

Script Kiddies:

Attacks can be scripted. A simple smurf denial-of-service attack is not hard to implement. But an underground establishment has written scripts for many of the popular attacks. With a script, attackers need not understand the nature of the attack or even the concept of a network. The hacker community is active in creating scripts for known vulnerabilities. People who download and run attack scripts are called script kiddies. As the rather derogatory name implies, script kiddies are not well respected in the attacker community because the damage they do requires almost no creativity or innovation. Nevertheless, script kiddies can cause serious damage, sometimes without even knowing what they do.

Building Blocks:

This chapter's attack types do not form an exhaustive list, but they represent the kinds of vulnerabilities being exploited, their sources, and their severity. A good attacker knows these vulnerabilities and many more. An attacker simply out to cause minor damage to a randomly selected site could use any of the techniques we have described, perhaps under script control. A dedicated attacker who targets one location can put together several pieces of an attack to compound the damage. Often, the attacks are done in series so that each part builds on the information gleaned from previous attacks. For example, a wiretapping attack may yield reconnaissance information with which to form an ActiveX attack that transfers a Trojan horse that monitors for sensitive data in transmission. Putting the attack pieces together like building blocks

expands the number of targets and increases the degree of damage.

10.5 SUMMARY

This chapter covers a very large and important area of computer security: networks and distributed applications. As the world becomes more connected by networks, the significance of network security will certainly continue to grow. Security issues for networks are visible and important, but their analysis is like the analysis done for other aspects of security. That is, we ask questions about what we are protecting and why we are protecting it.

A network has many different vulnerabilities, but all derive from an underlying model of computer, communications, and information systems security. Threats are raised against the key aspects of security: confidentiality, integrity, and availability.

Network assets include the network infrastructure, applications programs and, most importantly, data. Recall that threats are actions or situations that offer potential harm to or loss of confidentiality, integrity, or availability, in the form of interception (eavesdropping or passive wiretapping), modification (active wiretapping, falsification, and compromise of authenticity), and denial of service. In stand-alone computing, most agents have a strong motive for an attack. But in networks we see new threat agents; anyone can be a victim of essentially a random attack. The strongest network controls are solid authentication, access control, and encryption.

Networks usually employ many copies of the same or similar software, with a copy on each of several (or all) machines in the network. This similarity, combined with connectivity, means that any fault in one copy of a program can create vulnerabilities spread across many machines. Mass-market software often has flaws, and each flaw can be studied and exploited by an attacker. In large networks, a huge number of potential attackers can probe the software extensively; the result is that a network often includes many identified faults and software patches to counter them.

10.6 REVIEW QUESTIONS

- a) Explain the characteristics of the network.
- b) Explain the different types of media.
- c) List the strengths and weaknesses of different types of communication medium.
- d) What are the different types of networks?

- e) What are the differences between network and stand-alone environment?
- f) What are the motives to attack a network?
- g) How do attackers perpetrate their attacks?
- h) Write a short note on eavesdropping and wiretapping.
- i) What is impersonation? What choices does an attacker have during impersonation?
- j) Explain the threats to message confidentiality.
- k) Explain the threats to message integrity.
- l) Write a short note on website vulnerabilities.
- m) Explain the various availability attacks (DOS attacks).
- n) Explain the different threats in active code.

10.7 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- *Security in Computing* by C. P. Pfleeger, and S. L. Pfleeger, Pearson Education.
- *Computer Security: Art and Science* by Matt Bishop, Pearson Education.
- *Cryptography and Network Security: Principles and practice* by Stallings
- *Network Security* by Kaufman, Perlman, Speciner
- *Network Security : A Beginner's Guide* by Eric Maiwald, TMH
- *Java Network Security* by Macro Pistoia, Pearson Education
- *Principles of information security* by Whitman, Mattord, Thomson



SECURITY IN NETWORKS - II

Unit Structure:

- 11.0 Objectives
- 11.1 Introduction
- 11.2 Network Security Controls
 - 11.2.1. Security Threat Analysis
 - 11.2.2. Design and Implementation
 - 11.2.3. Architecture
 - 11.2.4. Encryption
 - 11.2.5. Content Integrity
 - 11.2.6. Strong Authentication
 - 11.2.7. Access Controls
 - 11.2.8. Wireless Security
 - 11.2.9. Alarms and Alerts
 - 11.2.10. Honeypots
 - 11.2.11. Traffic Flow Security
- 11.3 Firewalls
 - 11.3.1. What is a Firewall?
 - 11.3.2. Design of Firewalls
 - 11.3.3. Types of Firewalls
 - 11.3.4. Personal Firewalls
 - 11.3.5. Comparison of Firewall Types
 - 11.3.6. What Firewalls Can- and Cannot- Block
- 11.4 Intrusion Detection Systems
 - 11.4.1. Types of IDSs
 - 11.4.2. Goals for Intrusion Detection Systems
 - 11.4.3. IDS Strengths and Limitations
- 11.5 Secure E-mail
 - 11.5.1. Security for E-mail
 - 11.5.2. Requirements and Solutions
 - 11.5.3. Designs
 - 11.5.4. Example Secure E-mail Systems

- 11.6 Example Protocols
 - 11.6.1. SSL
 - 11.6.2. PEM
 - 11.6.3. IPSec
- 11.7 Summary
- 11.8 Review Questions
- 11.9 Bibliography, References and Further Reading

11.0 OBJECTIVES

At the end of this chapter, you will be able to understand:

- Controls against network attacks: physical security, policies and procedures, and a range of technical controls
- Firewalls: design, capabilities, limitations
- Intrusion detection systems
- Private e-mail: PGP and S/MIME

11.1 INTRODUCTION

In this chapter we consider three categories of controls: First, as you can well imagine, the familiar control of encryption is a strong tool for preserving both confidentiality and integrity in networks. We describe architecturally how encryption can be used and then introduce two specific applications of cryptography to networking: encrypted communication between a browser and its websites, called SSL encryption, and encrypted links within a network, called a virtual private network or VPN. Then we introduce a network-protection tool called a firewall, which is just an instantiation of the familiar reference monitor. We end the study of controls with another device, called an intrusion detection or protection system, that monitors network traffic to identify and counter specific malicious network threats.

11.2 NETWORK SECURITY CONTROLS

The list of security attacks is long, and the news media carry frequent accounts of serious security incidents. Previous chapters have presented several strategies for addressing security concerns, such as encryption for confidentiality and integrity, reference monitors for access control, and overlapping controls for defence in depth. These strategies are also useful in protecting networks. This section presents many excellent defences available to the network security engineer. Subsequent sections provide detailed

explanations for three particularly important controls, firewalls, intrusion detection systems, and encrypted e-mail.

11.2.1 Security Threat Analysis:

Recall the three steps of a security threat analysis in other situations. First, we scrutinize all the parts of a system so that we know what each part does and how it interacts with other parts. Next, we consider possible damage to confidentiality, integrity, and availability. Finally, we hypothesize the kinds of attacks that could cause this damage. We can take the same steps with a network. Why are all these attacks possible? Size, anonymity, ignorance, misunderstanding, complexity, dedication, and programming all contribute. But we have help at hand; we look next at specific threats and their countermeasures.

11.2.2 Design and Implementation:

Concepts from the work of the early trusted operating systems projects have natural implications for networks as well. And assurance relates to networked systems. In general, the Open Web Applications project has documented many of the techniques people can use to develop secure web applications. Thus, having addressed secure programming from several perspectives.

11.2.3 Architecture:

When we build or modify computer-based systems, we can give some thought to their overall architecture and plan to "build in" security as one of the key constructs. Similarly, the architecture or design of a network can have a significant effect on its security.

Segmentation:

Just as segmentation was a powerful security control in operating systems, it can limit the potential for harm in a network in two important ways: Segmentation reduces the number of threats, and it limits the amount of damage a single vulnerability can allow. Separate access is another way to segment the network.

Redundancy:

Another key architectural control is redundancy: allowing a function to be performed on more than one node, to avoid "putting all the eggs in one basket."

Single Points of Failure:

Ideally, the architecture should make the network immune to failure. In fact, the architecture should at least make sure that the system tolerates failure in an acceptable way (such as slowing down but not stopping processing or recovering and restarting incomplete

transactions). One way to evaluate the network architecture's tolerance of failure is to look for single points of failure.

Mobile Agents:

Mobile code and hostile agents are potential methods of attack. However, they can also be forces for good. Good agents might look for unsecured wireless access, software vulnerabilities, or embedded malicious code.

11.2.4 Encryption:

Encryption is probably the most important and versatile tool for a network security expert. We have seen in earlier chapters that encryption is powerful for providing privacy, authenticity, integrity, and limited access to data. Because networks often involve even greater risks, they often secure data with encryption, perhaps in combination with other controls. Before we begin to study the use of encryption to counter network security threats, let us consider these points. First, remember that encryption is not a panacea or silver bullet. A flawed system design with encryption is still a flawed system design. Second, notice that encryption protects only what is encrypted. Finally, encryption is no more secure than its key management. If an attacker can guess or deduce a weak encryption key, the game is over.

Link Encryption:

In link encryption, data are encrypted just before the system places them on the physical communications link. In this case, encryption occurs at layer 1 or 2 in the OSI model. (A similar situation occurs with TCP/IP protocols.) Similarly, decryption occurs just as the communication arrives at and enters the receiving computer. Link encryption is invisible to the user. The encryption becomes a transmission service performed by a low-level network protocol layer, just like message routing or transmission error detection.

End-to-End Encryption:

As its name implies, end-to-end encryption provides security from one end of a transmission to the other. The encryption can be applied by a hardware device between the user and the host. Alternatively, the encryption can be done by software running on the host computer. In either case, the encryption is performed at the highest levels (layer 7, application, or perhaps at layer 6, presentation) of the OSI model. Since the encryption precedes all the routing and transmission processing of the layer, the message is transmitted in encrypted form throughout the network.

Table 11-1 Comparison of Link and End-to-End Encryption

Link Encryption	End-to-End Encryption
Security within hosts	
Data partially exposed in sending host	Data protected in sending host
Data partially exposed in intermediate nodes	Data protected through intermediate nodes
Role of user	
Applied by sending host	Applied by user application
Invisible to user	User application encrypts
Host administrators select encryption	User selects algorithm
One facility for all users	Each user selects
Can be done in software or hardware	Usually software implementation; occasionally performed by user add-on hardware
All or no data encrypted	User can selectively encrypt individual data items
Implementation considerations	
Requires one key per pair of hosts	Requires one key per pair of users
Provides node authentication	Provides user authentication

Virtual Private Networks:

Link encryption can be used to give a network's users the sense that they are on a private network, even when it is part of a public network. For this reason, the approach is called a virtual private network (or VPN). Typically, physical security and administrative security are strong enough to protect transmission inside the perimeter of a network. Thus, the greatest exposure for a user is between the user's workstation or client and the perimeter of the host network or server. A firewall is an access control device that sits between two networks or two network segments. It filters all traffic between the protected or "inside" network and a less trustworthy or "outside" network or segment. With the VPN, we say that the communication passes through an encrypted tunnel or tunnel.

PKI and Certificates:

A public key infrastructure, or PKI, is a process created to enable users to implement public key cryptography, usually in a large (and frequently, distributed) setting. PKI offers each user a set of services, related to identification and access control, as follows:

- Create certificates associating a user's identity with a (public) cryptographic key
- Give out certificates from its database

- Sign certificates, adding its credibility to the authenticity of the certificate
- Confirm (or deny) that a certificate is valid
- Invalidate certificates for users who no longer are allowed access or whose private key has been exposed

PKI sets up entities, called certificate authorities, that implement the PKI policy on certificates. The general idea is that a certificate authority is trusted, so users can delegate the construction, issuance, acceptance, and revocation of certificates to the authority. The security involved in protecting the certificates involves administrative procedures. For example, more than one operator should be required to authorize certification requests. Controls should be put in place to detect hackers and prevent them from issuing bogus certificate requests. These controls might include digital signatures and strong encryption. Finally, a secure audit trail is necessary for reconstructing certificate information should the system fail and for recovering if a hacking attack does indeed corrupt the authentication process.

SSH Encryption:

SSH (secure shell) is a pair of protocols (versions 1 and 2), originally defined for Unix but also available under Windows 2000, that provides an authenticated and encrypted path to the shell or operating system command interpreter. Both SSH versions replace Unix utilities such as Telnet, *rlogin*, and *rsh* for remote access. SSH protects against spoofing attacks and modification of data in communication. The SSH protocol involves negotiation between local and remote sites for encryption algorithm (for example, DES, IDEA, AES) and authentication (including public key and Kerberos).

SSL Encryption:

The SSL (Secure Sockets Layer) protocol was originally designed by Netscape to protect communication between a web browser and server. It is also known now as TLS, for transport layer security. SSL interfaces between applications (such as browsers) and the TCP/IP protocols to provide server authentication, optional client authentication, and an encrypted communications channel between client and server. Client and server negotiate a mutually supported suite of encryption for session encryption and hashing; possibilities include triple DES and SHA1, or RC4 with a 128-bit key and MD5. The protocol is simple but effective, and it is the most widely used secure communication protocol on the Internet. However, remember that SSL protects only from the client's browser to the server's decryption point (which is often only to the server's firewall or, slightly stronger, to the computer that runs the web

application). Data are exposed from the user's keyboard to the browser and throughout the recipient's company.

IPSec:

IPSec is somewhat like SSL, in that it supports authentication and confidentiality in a way that does not necessitate significant change either above it (in applications) or below it (in the TCP protocols). Like SSL, it was designed to be independent of specific cryptographic protocols and to allow the two communicating parties to agree on a mutually supported set of protocols. The basis of IPSec is what is called a security association, which is essentially the set of security parameters for a secured communication channel. As with most cryptographic applications, the critical element is key management. IPSec addresses this need with ISAKMP or Internet Security Association Key Management Protocol.

Signed Code:

Someone can place malicious active code on a web site to be downloaded by unsuspecting users. A partial not complete approach to reducing this risk is to use signed code. A trustworthy third party appends a digital signature to a piece of code, supposedly connoting more trustworthy code. A signature structure in a PKI helps to validate the signature.

Encrypted E-mail:

An electronic mail message is much like the back of a post card. The mail carrier (and everyone in the postal system through whose hands the card passes) can read not just the address but also everything in the message field. To protect the privacy of the message and routing information, we can use encryption to protect the confidentiality of the message and perhaps its integrity.

11.2.5 Content Integrity:

Content integrity comes as a bonus with cryptography. No one can change encrypted data in a meaningful way without breaking the encryption. This does not say, however, that encrypted data cannot be modified. Changing even one bit of an encrypted data stream affects the result after decryption, often in a way that seriously alters the resulting plaintext. We need to consider three potential threats:

- malicious modification that changes content in a meaningful way
- malicious or nonmalicious modification that changes content in a way that is not necessarily meaningful

- nonmalicious modification that changes content in a way that will not be detected

Encryption addresses the first of these threats very effectively. To address the others, we can use other controls.

Error Correcting Codes:

We can use error detection and error correction codes to guard against modification in a transmission. The codes work as their names imply: Error detection codes detect when an error has occurred, and error correction codes can correct errors without requiring retransmission of the original message. The error code is transmitted along with the original data, so the recipient can recompute the error code and check whether the received result matches the expected value. The simplest error detection code is a parity check. An extra bit is added to an existing group of data bits depending on their sum or an exclusive OR. The two kinds of parity are called even and odd.

There are other kinds of error detection codes, such as hash codes and Huffman codes. Some of the more complex codes can detect multiple-bit errors (two or more bits changed in a data group) and may be able to pinpoint which bits have been changed. Parity and simple error detection and correction codes are used to detect nonmalicious changes in situations in which there may be faulty transmission equipment, communications noise and interference, or other sources of spurious changes to data.

Cryptographic Checksum:

Malicious modification must be handled in a way that prevents the attacker from modifying the error detection mechanism as well as the data bits themselves. One way to do this is to use a technique that shrinks and transforms the data, according to the value of the data bits. A cryptographic checksum (sometimes called a message digest) is a cryptographic function that produces a checksum. The cryptography prevents the attacker from changing the data block (the plaintext) and changing the checksum value (the ciphertext) to match. Two major uses of cryptographic checksums are code tamper protection and message integrity protection in transit.

11.2.6 Strong Authentication:

Networked environments need authentication, too. In the network case, however, authentication may be more difficult to achieve securely because of the possibility of eavesdropping and wiretapping, which are less common in nonnetworked environments. Also, both ends of a communication may need to be authenticated to each other: Before you send your password across a network, you want to know that you are really communicating with the remote host

you expect.

One-Time Password:

The wiretap threat implies that a password could be intercepted from a user who enters a password across an unsecured network. A one-time password can guard against wiretapping and spoofing of a remote host. As the name implies, a one-time password is good for one use only. What are the advantages and disadvantages of this approach? First, it is easy to use. It largely counters the possibility of a wiretapper reusing a password. With a strong password-generating algorithm, it is immune to spoofing. However, the system fails if the user loses the generating device or, worse, if the device falls into an attacker's hands. Because a new password is generated only once a minute, there is a small (one-minute) window of vulnerability during which an eavesdropper can reuse an intercepted password.

Challenge Response Systems:

To counter the loss and reuse problems, a more sophisticated one-time password scheme uses challenge and response. A challenge and response device look like a simple pocket calculator. The user first authenticates to the device, usually by means of a PIN. The remote system sends a random number, called the "challenge," which the user enters the device. The device responds to that number with another number, which the user then transmits to the system. The system prompts the user with a new challenge for each use. Thus, this device eliminates the small window of vulnerability in which a user could reuse a time-sensitive authenticator.

Digital Distributed Authentication:

In the 1980s, Digital Equipment Corporation recognized the problem of needing to authenticate nonhuman entities in a computing system. For example, a process might retrieve a user query, which it then reformats, perhaps limits, and submits to a database manager. Both the database manager and the query processor want to be sure that a communication channel is authentic between the two. Neither of these servers is running under the direct control or supervision of a human (although each process was, of course, somehow initiated by a human). Human forms of access control are thus inappropriate. Digital created a simple architecture for this requirement, effective against the following threats:

- *impersonation* of a server by a rogue process, for either of the two servers involved in the authentication
- *interception or modification* of data exchanged between servers
- *replay* of a previous authentication

The architecture assumes that each server has its own private

key and that the corresponding public key is available to or held by every other process that might need to establish an authenticated channel.

Kerberos:

Kerberos is a system that supports authentication in distributed systems. Originally designed to work with secret key encryption, Kerberos, in its latest version, uses public key technology to support key exchange. The Kerberos system was designed at Massachusetts Institute of Technology. Kerberos is used for authentication between intelligent processes, such as client-to-server tasks, or a user's workstation to other hosts. Kerberos is based on the idea that a central server provides authenticated tokens, called tickets, to requesting applications. A ticket is an unforgeable, nonrepayable, authenticated object. That is, it is an encrypted data structure naming a user and a service that user can obtain. It also contains a time value and some control information.

Kerberos was carefully designed to withstand attacks in distributed environments:

- ***No passwords communicated on the network:*** A user's password is stored only at the Kerberos server. The user's password is not sent from the user's workstation when the user initiates a session.
- ***Cryptographic protection against spoofing:*** Each access request is mediated by the ticket-granting server, which knows the identity of the requester, based on the authentication performed initially by the Kerberos server and on the fact that the user was able to present a request encrypted under a key that had been encrypted under the user's password.
- ***Limited period of validity:*** Each ticket is issued for a limited time period; the ticket contains a timestamp with which a receiving server will determine the ticket's validity. In this way, certain long-term attacks, such as brute force cryptanalysis, will usually be neutralized because the attacker will not have time to complete the attack.
- ***Timestamps to prevent replay attacks:*** Kerberos requires reliable access to a universal clock. Each user's request to a server is stamped with the time of the request. A server receiving a request compares this time to the current time and fulfills the request only if the time is reasonably close to the current time. This time-checking prevents most replay attacks, since the attacker's presentation of the ticket will be delayed too long.
- ***Mutual authentication:*** The user of a service can be assured of

any server's authenticity by requesting an authenticating response from the server. The user sends a ticket to a server and then sends the server a request encrypted under the session key for that server's service; the ticket and the session key were provided by the ticket-granting server. The server can decrypt the ticket only if it has the unique key it shares with the ticket-granting server.

Kerberos is not a perfect answer to security problems in distributed systems.

- *Kerberos requires continuous availability of a trusted ticket-granting server:* Because the ticket-granting server is the basis of access control and authentication, constant access to that server is crucial. Both reliability (hardware or software failure) and performance (capacity and speed) problems must be addressed.
- *Authenticity of servers requires a trusted relationship between the ticket-granting server and every server:* The ticket-granting server must share a unique encryption key with each "trustworthy" server. The ticket-granting server (or that server's human administrator) must be convinced of the authenticity of that server. In a local environment, this degree of trust is warranted. In a widely distributed environment, an administrator at one site can seldom justify trust in the authenticity of servers at other sites.
- *Kerberos requires timely transactions:* To prevent replay attacks, Kerberos limits the validity of a ticket. A replay attack could succeed during the period of validity, however. And setting the period fairly is hard: Too long increases the exposure to replay attacks, while too short requires prompt user actions and risks providing the user with a ticket that will not be honoured when presented to a server. Similarly, subverting a server's clock allows reuse of an expired ticket.
- *A subverted workstation can save and later replay user passwords:* This vulnerability exists in any system in which passwords, encryption keys, or other constant, sensitive information is entered in the clear on a workstation that might be subverted.
- *Password guessing works:* A user's initial ticket is returned under the user's password. An attacker can submit an initial authentication request to the Kerberos server and then try to decrypt the response by guessing at the password.
- *Kerberos does not scale well:* The architectural model of Kerberos assumes one Kerberos server and one ticket-granting

server, plus a collection of other servers, each of which shares a unique key with the ticket-granting server. Adding a second ticket-granting server, for example, to enhance performance or reliability, would require duplicate keys or a second set for all servers. Duplication increases the risk of exposure and complicates key updates, and second keys more than double the work for each server to act on a ticket.

- *Kerberos is a complete solution:* All applications must use Kerberos authentication and access control. Currently, few applications use Kerberos authentication, and so integration of Kerberos into an existing environment requires modification of existing applications, which is not feasible.

11.2.7 Access Controls:

Authentication deals with the *who* of security policy enforcement; access controls enforce the *what* and *how*.

ACLs on Routers:

Routers perform the major task of directing network traffic either to subnetworks they control or to other routers for subsequent delivery to other subnetworks. Routers convert external IP addresses into internal MAC addresses of hosts on a local subnetwork. Suppose a host is being spammed (flooded) with packets from a malicious rogue host. Routers can be configured with access control lists to deny access to hosts from hosts. So, a router could delete all packets with a source address of the rogue host and a destination address of the target host.

This approach has three problems, however. First, routers in large networks perform a lot of work: They must handle every packet coming into and going out of the network. Adding ACLs to the router requires the router to compare every packet against the ACLs. The second problem is also an efficiency issue: Because of the volume of work they perform, routers are designed to perform only essential services. Logging of activity is usually not done on a router because of the volume of traffic and the performance penalty logging would entail. With ACLs, it would be useful to know how many packets were being deleted, to know if a ACL could be removed (thereby improving performance). But without logging it is impossible to know whether an ACL is being used. The final limitation on placing ACLs on routers concerns the nature of the threat. A router inspects only source and destination addresses. An attacker usually does not reveal an actual source address. To reveal the real source address would be equivalent to a bank robber's leaving his home address and a description of where he plans to store the stolen money. Because someone can easily forge any source address on a UDP datagram, many attacks use UDP protocols with false source addresses so that

the attack cannot be blocked easily by a router with an ACL.

Firewalls:

A firewall does the screening that is less appropriate for a router to do. A router's primary function is addressing, whereas a firewall's primary function is filtering. Firewalls can also do auditing. Even more important, firewalls can examine an entire packet's contents, including the data portion, whereas a router is concerned only with source and destination MAC and IP addresses.

11.2.8 Wireless Security:

Because wireless computing is so exposed, it requires measures to protect communications between a computer (called the client) and a wireless base station or access point. Remembering that all these communications are on predefined radio frequencies, you can expect an eavesdropping attacker to try to intercept and impersonate. Pieces to protect are finding the access point, authenticating the remote computer to the access point, and vice versa, and protecting the communication stream.

SSID:

The Service Set Identifier or SSID is the identification of an access point; it is a string of up to 32 characters. Obviously the SSIDs need to be unique in each area to distinguish one wireless network from another. A client and an access point engage in a handshake to locate each other. In what is called "open mode," an access point can continually broadcast its appeal, indicating that it is open for the next step in establishing a connection. Open mode is a poor security practice because it advertises the name of an access point to which an attacker might attach. "Closed" or "stealth mode" reverses the order of the protocol: The client must send a signal seeking an access point with a SSID before the access point responds to that one query with an invitation to connect.

WEP:

The second step in securing a wireless communication involves use of encryption. The original 802.11 wireless standard relied upon a cryptographic protocol called wired equivalent privacy or WEP. WEP was meant to provide users privacy equivalent to that of a dedicated wire, that is, immunity to most eavesdropping and impersonation attacks. WEP uses an encryption key shared between the client and the access point. First, the WEP standard uses either a 64- or 128-bit encryption key. The user enters the key in any convenient form, usually in hexadecimal or as an alphanumeric string that is converted to a number.

WPA and WPA2:

The alternative to WEP is Wi-Fi Protected Access or WPA,
[Unedited Version: Information Security pg.1](#)

approved in 2003. The IEEE standard 802.11i is now known as WPA2, approved in 2004, and is an extension of WPA. How does WPA improve upon WEP? The setup protocol for WPA and WPA2 is much more robust than that for WEP. Setup for WPA involves three protocol steps: authentication, a four-way handshake (to ensure that the client can generate cryptographic keys and to generate and install keys for both encryption and integrity on both ends), and an optional group key handshake (for multicast communication.)

11.2.9 Alarms and Alerts:

The logical view of network protection has both a router and a firewall which provides layers of protection for the internal network. Now let us add one more layer to this defence. An intrusion detection system is a device that is placed inside a protected network to monitor what occurs within the network. If an attacker passes through the router and passes through the firewall, an intrusion detection system offers the opportunity to detect the attack at the beginning, in progress, or after it has occurred. Intrusion detection systems activate an alarm, which can take defensive action.

11.2.10 Honeypots:

How do you catch a mouse? You set a trap with bait (food the mouse finds attractive) and catch the mouse after it is lured into the trap. You can catch a computer attacker the same way. You put up a honeypot for several reasons:

- to watch what attackers, do, in order to learn about new attacks (so that you can strengthen your defences against these new attacks)
- to lure an attacker to a place in which you may be able to learn enough to identify and stop the attacker
- to provide an attractive but diversionary playground, hoping that the attacker will leave your real system alone

A honeypot has no special features. It is just a computer system or a network segment, loaded with servers and devices and data. It may be protected with a firewall, although you want the attackers to have some access. There may be some monitoring capability, done carefully so that the monitoring is not evident to the attacker. The two difficult features of a honeypot are putting up a believable, attractive false environment and confining and monitoring the attacker surreptitiously.

11.2.11 Traffic Flow Security:

If the attacker can detect an exceptional volume of traffic

between two points, the attacker may infer the location of an event about to occur. The countermeasure to traffic flow threats is to disguise the traffic flow. One way to disguise traffic flow, albeit costly and perhaps crude, is to ensure a steady volume of traffic between two points. A more sophisticated approach to traffic flow security is called onion routing. Packages for onion routing can be any network transmissions. The most popular uses, however, are covert email, and private web browsing. The Tor project distributes free software and enlists an open network that uses onion routing to defend against traffic analysis. Tor (which stands for The Onion Router) protects by transferring communications around a distributed network of over 5,000 relays run by volunteers all around the world: It prevents outsiders watching Internet connections from learning what sites a user visits, and it prevents sites from learning the user's physical location.

11.3 FIREWALLS

Firewalls were officially invented in the early 1990s, but the concept really reflects the reference monitor from two decades earlier.

11.3.1 What is a Firewall?

A firewall is a device that filters all traffic between a protected or "inside" network and a less trustworthy or "outside" network. Usually a firewall runs on a dedicated device; because it is a single point through which traffic is channelled, performance is important, which means non-firewall functions should not be done on the same machine. Because a firewall is executable code, an attacker could compromise that code and execute from the firewall's device. Thus, the fewer pieces of code on the device, the fewer tools the attacker would have by compromising the firewall. Firewall code usually runs on a proprietary or carefully minimized operating system. The purpose of a firewall is to keep "bad" things outside a protected environment. To accomplish that, firewalls implement a security policy that is specifically designed to address what bad things might happen. For example, the policy might be to prevent any access from outside (while still allowing traffic to pass *from* the inside *to* the outside). Alternatively, the policy might permit accesses only from certain places, from certain users, or for certain activities. Part of the challenge of protecting a network with a firewall is determining which security policy meets the needs of the installation.

We can describe the two schools of thought as "that which is not expressly forbidden is permitted" (default permit) and "that which is not expressly permitted is forbidden" (default deny). Users, always interested in new features, prefer the former. Security experts,

relying on several decades of experience, strongly counsel the latter. An administrator implementing or configuring a firewall must choose one of the two approaches, although the administrator can often broaden the policy by setting the firewall's parameters.

11.3.2 Design of Firewalls:

Remember that a reference monitor must be

- always invoked
- tamperproof
- small and simple enough for rigorous analysis

A firewall is a special form of reference monitor. By carefully positioning a firewall within a network, we can ensure that all network accesses that we want to control must pass through it. This restriction meets the "always invoked" condition. A firewall is typically well isolated, making it highly immune to modification. Usually a firewall is implemented on a separate computer, with direct connections only to the outside and inside networks. This isolation is expected to meet the "tamperproof" requirement. And firewall designers strongly recommend keeping the functionality of the firewall simple.

11.3.3 Types of Firewalls:

Firewalls have a wide range of capabilities. Types of firewalls include

- packet filtering gateways or screening routers
- stateful inspection firewalls
- application proxies
- guards
- personal firewalls

Each type does different things; no one is necessarily "right" and the others "wrong." In this section, we examine each type to see what it is, how it works, and what its strengths and weaknesses are. Simplicity in a security policy is not a bad thing; the important question to ask when choosing a type of firewall is what threats an installation needs to counter.

Packet Filtering Gateway:

A packet filtering gateway or screening router is the simplest, and in some situations, the most effective type of firewall. A packet filtering gateway controls access to packets based on packet address (source or destination) or specific transport protocol type (such as HTTP web traffic). As described earlier in this chapter, putting ACLs on routers may severely impede their performance. But a separate firewall behind (on the local side) of the router can screen

traffic before it gets to the protected network.

Packet filters do not "see inside" a packet; they block or accept packets solely based on the IP addresses and ports. Thus, any details in the packet's data field (for example, allowing certain Telnet commands while blocking other services) is beyond the capability of a packet filter. Packet filters can perform the very important service of ensuring the validity of inside addresses. A packet filter sits between the inside network and the outside net, so it can know if a packet from the outside is forging an inside address. A screening packet filter might be configured to block all packets from the *outside* that claimed their source address was an *inside* address.

Stateful Inspection Firewall:

Filtering firewalls work on packets one at a time, accepting or rejecting each packet and moving on to the next. They have no concept of "state" or "context" from one packet to the next. A stateful inspection firewall maintains state information from one packet to another in the input stream. One classic approach used by attackers is to break an attack into multiple packets by forcing some packets to have very short lengths so that a firewall cannot detect the signature of an attack split across two or more packets. A stateful inspection firewall would track the sequence of packets and conditions from one packet to another to thwart such an attack.

Application Proxy:

An application proxy gateway, also called a bastion host, is a firewall that simulates the (proper) effects of an application so that the application receives only requests to act properly. A proxy gateway is a two-headed device: It looks to the inside as if it is the outside (destination) connection, while to the outside it responds just as the insider would.

An application proxy runs pseudo applications. For instance, when electronic mail is transferred to a location, a sending process at one site and a receiving process at the destination communicate by a protocol that establishes the legitimacy of a mail transfer and then actually transfers the mail message. The protocol between sender and destination is carefully defined. A proxy gateway essentially intrudes in the middle of this protocol exchange, seeming like a destination in communication with the sender that is outside the firewall, and seeming like the sender in communication with the real destination on the inside. The proxy in the middle can screen the mail transfer, ensuring that only acceptable e-mail protocol commands are sent to the destination. The proxies on the firewall can be tailored to specific requirements, such as logging details about accesses. They can even present a common user interface to what may be dissimilar internal functions. Suppose the internal network has a mixture of operating system types, none of which

support strong authentication through a challenge response token. The proxy can demand strong authentication (name, password, and challenge response), validate the challenge response itself, and then pass on only simple name and password authentication details in the form required by a specific internal host's operating system.

Guard:

A guard is a sophisticated firewall. Like a proxy firewall, it receives protocol data units, interprets them, and passes through the same or different protocol data units that achieve either the same result or a modified result. The guard decides what services to perform on the user's behalf in accordance with its available knowledge, such as whatever it can reliably know of the (outside) user's identity, previous interactions, and so forth. The degree of control a guard can provide is limited only by what is computable. But guards and proxy firewalls are similar enough that the distinction between them is sometimes fuzzy. Since the security policy implemented by the guard is somewhat more complex than the action of a proxy, the guard's code is also more complex and therefore more exposed to error. Simpler firewalls have fewer possible ways to fail or be subverted.

11.3.4 Personal Firewalls:

A personal firewall is an application program that runs on a workstation to block unwanted traffic, usually from the network. A personal firewall can complement the work of a conventional firewall by screening the kind of data a single host will accept, or it can compensate for the lack of a regular firewall, as in a private DSL or cable modem connection. Just as a network firewall screens incoming and outgoing traffic for that network, a personal firewall screens traffic on a single workstation. A workstation could be vulnerable to malicious code or malicious active agents (ActiveX controls or Java applets), leakage of personal data stored on the workstation, and vulnerability scans to identify potential weaknesses.

The personal firewall is configured to enforce some policy. For example, the user may decide that certain sites, such as computers on the company network, are highly trustworthy, but most other sites are not. Combining a virus scanner with a personal firewall is both effective and efficient. A personal firewall runs on the very computer it is trying to protect. Thus, a clever attacker is likely to attempt an undetected attack that would disable or reconfigure the firewall for the future. Still, especially for cable modem, DSL, and other "always on" connections, the static workstation is a visible and vulnerable target for an ever-present attack community. A personal firewall can provide reasonable protection to clients that are not behind a network firewall.

11.3.5 Comparison of Firewall Types:

We can summarize the differences among the several types of firewalls we have studied. The comparisons are shown in Table 10-3. Firewall types are arranged generally from least sophisticated on the left to more so on the right, except for personal firewalls, which are more like an enterprise packet filter.

Table 11-2 Comparison of Firewall Types

Packet Filter	Stateful Inspection	Application Proxy	Circuit Gateway	Guard	Personal Firewall
Simplest decision-making rules, packet by packet	Correlates data across packets	Simulates effect of an application program	Joins two subnetworks	Implements any conditions that can be programmed	Similar to packet filter, but getting more complex
Sees only addresses and service protocol type	Can see addresses and data	Sees and analyzes full data portion of pack	Sees addresses and data	Sees and analyzes full content of data	Can see full data portion
Auditing limited because of speed limitations	Auditing possible	Auditing likely	Auditing likely	Auditing likely	Auditing likely
Screens based on connection rules	Screens based on information across multiple packets—in either headers or data	Screens based on behavior of application	Screens based on address	Screens based on interpretation of content	Typically, screens based on content of each packet individually, based on address or content
Complex addressing rules can make configuration tricky	Usually preconfigured to detect certain attack signatures	Simple proxies can substitute for complex decision rules, but proxies must be aware of application's behavior	Relatively simple addressing rules; make configuration straightforward	Complex guard functionality; can be difficult to define and program accurately	Usually starts in mode to deny all inbound traffic; adds addresses and functions to trust as they arise

11.3.6 What Firewalls Can- and Cannot- Block:

As we have seen, firewalls are not complete solutions to all computer security problems. A firewall protects only the perimeter of its environment against attacks from outsiders who want to execute code or access data on the machines in the protected environment. Keep in mind these points about firewalls.

- Firewalls can protect an environment only if the firewalls control the entire perimeter. That is, firewalls are effective only if no unmediated connections breach the perimeter. If even one inside host connects to an outside address, by a modem for example, the entire inside net is vulnerable through the modem and its host.
- Firewalls do not protect data outside the perimeter; data that have

properly passed (outbound) through the firewall are just as exposed as if there were no firewall.

- Firewalls are the most visible part of an installation to the outside, so they are the most attractive target for attack. For this reason, several different layers of protection, called defence in depth, are better than relying on the strength of just a single firewall.
- Firewalls must be correctly configured, that configuration must be updated as the internal and external environment changes, and firewall activity reports must be reviewed periodically for evidence of attempted or successful intrusion.
- Firewalls are targets for penetrators. While a firewall is designed to withstand attack, it is not impenetrable. Designers intentionally keep a firewall small and simple so that even if a penetrator breaks it, the firewall does not have further tools, such as compilers, linkers, loaders, and the like, to continue an attack.
- Firewalls exercise only minor control over the content admitted to the inside, meaning that inaccurate data or malicious code must be controlled by other means inside the perimeter.

Firewalls are important tools in protecting an environment connected to a network. However, the environment must be viewed as a whole, all possible exposures must be considered, and the firewall must fit into a larger, comprehensive security strategy. Firewalls alone cannot secure an environment.

11.4 INTRUSION DETECTION SYSTEMS

After the perimeter controls, firewall, and authentication and access controls block certain actions, some users are admitted using a computing system. Most of these controls are preventive: They block known bad things from happening. Many studies have shown that most computer security incidents are caused by insiders, people who would not be blocked by a firewall. And insiders require access with significant privileges to do their daily jobs. Most of the harm from insiders is not malicious; it is honest people making honest mistakes. Then, too, there are the potential malicious outsiders who have somehow passed the screens of firewalls and access controls. Prevention, although necessary, is not a complete computer security control; detection during an incident copes with harm that cannot be prevented in advance.

Intrusion detection systems complement these preventive controls as the next line of defence. An intrusion detection system (IDS) is a device, typically another separate computer, that monitors activity to identify malicious or suspicious events. An IDS is a sensor, like a smoke detector, that raises an alarm if specific things occur. An IDS receives raw inputs from sensors. It saves those inputs,

[Unedited Version: Information Security pg.1](#)

analyzes them, and takes some controlling action.

IDSs perform a variety of functions:

- monitoring users and system activity
- auditing system configuration for vulnerabilities and misconfigurations
- assessing the integrity of critical system and data files
- recognizing known attack patterns in system activity
- identifying abnormal activity through statistical analysis
- managing audit trails and highlighting user violation of policy or normal activity
- correcting system configuration errors
- installing and operating traps to record information about intruders

No one IDS perform all these functions. Let us look more closely at the kinds of IDSs and their use in providing security.

11.4.1 Types of IDSs:

The two general types of intrusion detection systems are signature based and heuristic. Signature-based intrusion detection systems perform simple pattern-matching and report situations that match a pattern corresponding to a known attack type. Heuristic intrusion detection systems, also known as anomaly, based, build a model of acceptable behaviour and flag exceptions to that model; for the future, the administrator can mark a flagged behaviour as acceptable so that the heuristic IDS will now treat that previously unclassified behaviour as acceptable.

Intrusion detection devices can be network based or host based. A network-based IDS is a stand-alone device attached to the network to monitor traffic throughout that network; a host-based IDS runs on a single workstation or client or host, to protect that one host.

Signature-Based Intrusion Detection:

A simple signature for a known attack type might describe a series of TCP SYN packets sent to many different ports in succession and at times close to one another, as would be the case for a port scan. An intrusion detection system would probably find nothing unusual in the first SYN, say, to port 80, and then another (from the same source address) to port 25. But as more and more ports receive SYN packets, especially ports that are not open, this pattern reflects a possible port scan. Similarly, some implementations of the protocol stack fail if they receive an ICMP packet with a data length of 65535 bytes, so such a packet would be a pattern for which to

watch.

The problem with signature-based detection is the signatures themselves. An attacker will try to modify a basic attack in such a way that it will not match the known signature of that attack. For example, the attacker may convert lowercase to uppercase letters or convert a symbol such as "blank space" to its character code equivalent %20. The IDS must necessarily work from a canonical form of the data stream in order to recognize that %20 matches a pattern with a blank space. The attacker may insert malformed packets that the IDS will see, to intentionally cause a pattern mismatch; the protocol handler stack will discard the packets because of the malformation. Each of these variations could be detected by an IDS, but more signatures require additional work for the IDS, which reduces performance.

Of course, signature based IDSs cannot detect a new attack for which a signature is not yet installed in the database. Every attack type starts as a new pattern at some time, and the IDS is helpless to warn of its existence. Signature-based intrusion detection systems tend to use statistical analysis. This approach uses statistical tools both to obtain sample measurements of key indicators and to determine whether the collected measurements fit the predetermined attack signatures. Ideally, signatures should match every instance of an attack, match subtle variations of the attack, but not match traffic that is not part of an attack. However, this goal is grand but unreachable.

Heuristic Intrusion Detection:

Because signatures are limited to specific, known attack patterns, another form of intrusion detection becomes useful. Instead of looking for matches, heuristic intrusion detection looks for behaviour that is out of the ordinary. The inference engine of an intrusion detection system performs continuous analysis of the system, raising an alert when the system's dirtiness exceeds a threshold. Inference engines work in two ways. Some, called state-based intrusion detection systems, see the system going through changes of overall state or configuration. They try to detect when the system has veered into unsafe modes. Others try to map current activity onto a model of unacceptable activity and raise an alarm when the activity resembles the model. These are called model-based intrusion detection systems. This approach has been extended to networks. Alternatively, intrusion detection can work from a model of known bad activity. For example, except for a few utilities (login, change password, create user), any other attempt to access a password file is suspect. This form of intrusion detection is known as misuse intrusion detection.

All heuristic intrusion detection activity is classified in one of three categories: good/benign, suspicious, or unknown. Over time, specific kinds of actions can move from one of these categories to another, corresponding to the IDS's learning whether certain actions are acceptable or not. As with pattern-matching, heuristic intrusion detection is limited by the amount of information the system has seen (to classify actions into the right category) and how well the current actions fit into one of these categories.

Stealth Mode:

An IDS is a network device (or, in the case of a host-based IDS, a program running on a network device). Any network device is potentially vulnerable to network attacks. How useful would an IDS be if it itself were deluged with a denial-of-service attack? If an attacker succeeded in logging in to a system within the protected network, wouldn't trying to disable the IDS be the next step? To counter those problems, most IDSs run in stealth mode, whereby an IDS has two network interfaces: one for the network (or network segment) being monitored and the other to generate alerts and perhaps other administrative needs. The IDS use the monitored interface as input only; it *never* sends packets out through that interface. Often, the interface is configured so that the device has no published address through the monitored interface; that is, a router cannot route anything to that address directly, because the router does not know such a device exists. It is the perfect passive wiretap. If the IDS need to generate an alert, it uses only the alarm interface on a separate control network.

Other IDS Types:

Some security engineers consider other devices to be IDSs as well. For instance, to detect unacceptable code modification, programs can compare the active version of a software code with a saved version of a digest of that code. The *tripwire* program is the most well-known software (or static data) comparison program. You run *tripwire* on a new system, and it generates a hash value for each file; then you save these hash values in a secure place (offline, so that no intruder can modify them while modifying a system file). If you later suspect your system may have been compromised, you rerun *tripwire*, providing it the saved hash values. It recomputes the hash values and reports any mismatches, which would indicate files that were changed. System vulnerability scanners, such as *ISS Scanner* or *Nessus*, can be run against a network. They check for known vulnerabilities and report flaws found. As we have seen, a honeypot is a faux environment intended to lure an attacker. It can be considered an IDS, in the sense that the honeypot may record an intruder's actions and even attempt to trace who the attacker is from actions, packet data, or connections.

11.4.2 Goals for Intrusion Detection Systems:

The two styles of intrusion detection pattern matching, and heuristic represent different approaches, each of which has advantages and disadvantages. Actual IDS products often blend the two approaches. Ideally, an IDS should be fast, simple, and accurate, while at the same time being complete. It should detect all attacks with little performance penalty. An IDS could use some or all the following design approaches:

- Filter on packet headers
- Filter on packet content
- Maintain connection state
- Use complex, multipacket signatures
- Use minimal number of signatures with maximum effect
- Filter in real time, online
- Hide its presence
- Use optimal sliding time window size to match signatures

Responding to Alarms:

Whatever the type, an intrusion detection system raises an alarm when it finds a match. The alarm can range from something modest, such as writing a note in an audit log, to something significant, such as paging the system security administrator. Implementations allow the user to determine what action the system should take on what events. What are possible responses? The range is unlimited and can be anything the administrator can imagine (and program). In general, responses fall into three major categories (any or all of which can be used in a single response):

- Monitor, collect data, perhaps increase amount of data collected
- Protect, act to reduce exposure
- Call a human

Monitoring is appropriate for an attack of modest (initial) impact. Perhaps the real goal is to watch the intruder, to see what resources are being accessed or what attempted attacks are tried. Another monitoring possibility is to record all traffic from a given source for future analysis. This approach should be invisible to the attacker. Protecting can mean increasing access controls and even making a resource unavailable. In contrast to monitoring, protecting may be very visible to the attacker. Finally, calling a human allows individual discrimination. The IDS can take an initial defensive action immediately while also generating an alert to a human who may take seconds, minutes, or longer to respond.

False Results:

Intrusion detection systems are not perfect, and mistakes are their biggest problem. Although an IDS might detect an intruder correctly most of the time, it may stumble in two different ways: by raising an alarm for something that is not really an attack (called a false positive, or type I error in the statistical community) or not raising an alarm for a real attack (a false negative, or type II error). Too many false positives mean the administrator will be less confident of the IDS's warnings, perhaps leading to a real alarm's being ignored. But false negatives mean that real attacks are passing the IDS without action. We say that the degree of false positives and false negatives represents the sensitivity of the system. Most IDS implementations allow the administrator to tune the system's sensitivity, to strike an acceptable balance between false positives and negatives.

11.4.3 IDS Strengths and Limitations:

Intrusion detection systems are evolving products. Research began in the mid-1980s and products had appeared by the mid-1990s. However, this area continues to change as new research influences the design of products. On the upside, IDSs detect an ever-growing number of serious problems. And as we learn more about problems, we can add their signatures to the IDS model. Thus, over time, IDSs continue to improve. At the same time, they are becoming cheaper and easier to administer. On the downside, avoiding an IDS is a priority for successful attackers. An IDS that is not well defended is useless. Fortunately, stealth mode IDSs are difficult even to find on an internal network, let alone to compromise. IDSs look for known weaknesses, whether through patterns of known attacks or models of normal behaviour. Similar IDSs may have identical vulnerabilities, and their selection criteria may miss similar attacks. Knowing how to evade a model of IDS is an important piece of intelligence passed within the attacker community. Of course, once manufacturers become aware of a shortcoming in their products, they try to fix it. Fortunately, commercial IDSs are pretty good at identifying attacks. Another IDS limitation is its sensitivity, which is difficult to measure and adjust. IDSs will never be perfect, so finding the proper balance is critical. A final limitation is not of IDSs *per se* but is one of use. An IDS does not run itself; someone must monitor its track record and respond to its alarms. An administrator is foolish to buy and install an IDS and then ignore it.

In general, IDSs are excellent additions to a network's security. Firewalls block traffic to ports or addresses; they also constrain certain protocols to limit their impact. But, firewalls must allow some traffic to enter a protected area. Watching what that traffic does inside the protected area is an IDS's job, which it does quite well.

11.5 SECURE E-MAIL

The final control we consider in depth is secure e-mail. Think about how much you use e-mail and how much you rely on the accuracy of its contents. How would you react if you received a message from your instructor saying that because you had done so well in your course so far, you were excused from doing any further work in it? What if that message were a joke from a classmate? We rely on e-mail's confidentiality and integrity for sensitive and important communications, even though ordinary e-mail has almost no confidentiality or integrity. In this section we investigate how to add confidentiality and integrity protection to ordinary e-mail.

11.5.1 Security for E-mail:

E-mail is vital for today's commerce, as well a convenient medium for communications among ordinary users. But, as we noted earlier, e-mail is very public, exposed at every point from the sender's workstation to the recipient's screen. Just as you would not put sensitive or private thoughts on a postcard, you must also acknowledge that e-mail messages are exposed and available for others to read. Sometimes we would like e-mail to be more secure. To define and implement a more secure form, we begin by examining the exposures of ordinary e-mail.

Threats to E-mail

Consider threats to electronic mail:

- message interception (confidentiality)
- message interception (blocked delivery)
- message interception and subsequent replay
- message content modification
- message origin modification
- message content forgery by outsider
- message origin forgery by outsider
- message content forgery by recipient
- message origin forgery by recipient
- denial of message transmission

Confidentiality and content forgery are often handled by encryption. Encryption can also help in a defence against replay, although we would also have to use a protocol in which each message contains something unique that is encrypted. Symmetric encryption cannot protect against forgery by a recipient, since both

sender and recipient share a common key; however, public key schemes can let a recipient decrypt but not encrypt. Because of lack of control over the middle points of a network, senders or receivers generally cannot protect against blocked delivery.

11.5.2 Requirements and Solutions:

If we were to make a list of the requirements for secure e-mail, our wish list would include the following protections:

- *message confidentiality* (the message is not exposed en route to the receiver)
- *message integrity* (what the receiver sees is what was sent)
- *sender authenticity* (the receiver is confident who the sender was)
- *nonrepudiation* (the sender cannot deny having sent the message)

Not all these qualities are needed for every message, but an ideal secure e-mail package would allow these capabilities to be invoked selectively.

11.5.3 Designs:

The standard for encrypted e-mail was developed by the Internet Society, through its architecture board (IAB) and research (IRTF) and engineering (IETF) task forces. The encrypted e-mail protocols are documented as an Internet standard in documents 1421, 1422, 1423, and 1424. This standard is the third refinement of the original specification. One of the design goals for encrypted e-mail was allowing security-enhanced messages to travel as ordinary messages through the existing Internet e-mail system. This requirement ensures that the large existing e-mail network would not require change to accommodate security. Thus, all protection occurs within the body of a message.

Confidentiality:

Because the protection has several aspects, we begin our description of them by looking first at how to provide confidentiality enhancements. The sender chooses a (random) symmetric algorithm encryption key. Then, the sender encrypts a copy of the entire message to be transmitted, including FROM:, TO:, SUBJECT:, and DATE: headers. Next, the sender prepends plaintext headers. For key management, the sender encrypts the message key under the recipient's public key and attaches that to the message as well. Encryption can potentially yield any string as output. Many e-mail handlers expect that message traffic will not contain characters other than the normal printable characters. Network e-mail handlers use unprintable characters as control signals in the traffic stream. To

avoid problems in transmission, encrypted e-mail converts the entire ciphertext message to printable characters.

The encrypted e-mail standard works most easily, using both symmetric and asymmetric encryption. The encrypted e-mail standard supports multiple encryption algorithms, using popular algorithms such as DES, triple DES, and AES for message confidentiality, and RSA and Diffie-Hellman for key exchange.

Other Security Features:

In addition to confidentiality, we may want various forms of integrity for secure e-mail. Encrypted e-mail messages always carry a digital signature, so the authenticity and non-repudiability of the sender is assured. The integrity is also assured because of a hash function (called a message integrity check, or MIC) in the digital signature. Optionally, encrypted e-mail messages can be encrypted for confidentiality.

Encryption for Secure E-mail:

The major problem with encrypted e-mail is key management. The certificate scheme is excellent for exchanging keys and for associating an identity with a public encryption key. The difficulty with certificates is building the hierarchy. Many organizations have hierarchical structures. The encrypted e-mail dilemma is moving beyond the single organization to an interorganizational hierarchy. Precisely because of the problem of imposing a hierarchy on a non-hierarchical world, PGP was developed as a simpler form of encrypted e-mail. Encrypted e-mail provides strong end-to-end security for electronic mail. Triple DES, AES, and RSA cryptography are quite strong, especially if RSA is used with a long bit key. The vulnerabilities remaining with encrypted e-mail come from the points not covered: the endpoints. An attacker with access could subvert a sender's or receiver's machine, modifying the code that does the privacy enhancements or arranging to leak a cryptographic key.

11.5.4 Example Secure E-mail Systems:

Encrypted e-mail programs are available from many sources. Several universities (including Cambridge University in England and The University of Michigan in the United States) and companies (BBN, RSA-DSI, and Trusted Information Systems) have developed either prototype or commercial versions of encrypted e-mail.

PGP:

PGP stands for Pretty Good Privacy. It was invented by Phil Zimmerman in 1991. Originally a free package, it became a commercial product after being bought by Network Associates in 1996. A freeware version is still available. PGP is widely available, both in commercial versions and freeware, and it is heavily used by

individuals exchanging private e-mail. PGP addresses the key distribution problem with what is called a "ring of trust" or a user's "keyring." One user directly gives a public key to another, or the second user fetches the first's public key from a server. Some people include their PGP public keys at the bottom of e-mail messages. And one person can give a second person's key to a third (and a fourth, and so on). Thus, the key association problem becomes one of caveat emptor: "Let the buyer beware." If I am reasonably confident that an e-mail message really comes from you and has not been tampered with, I will use your attached public key. If I trust you, I may also trust the keys you give me for other people. The model breaks down intellectually when you give me all the keys you received from people, who in turn gave you all the keys they got from still other people, who gave them all their keys, and so forth. PGP does not mandate a policy for establishing trust. Rather, each user is free to decide how much to trust each key received.

The PGP processing performs some or all the following actions, depending on whether confidentiality, integrity, authenticity, or some combination of these is selected:

- Create a random session key for a symmetric algorithm.
- Encrypt the message, using the session key (for message confidentiality).
- Encrypt the session key under the recipient's public key.
- Generate a message digest or hash of the message; sign the hash by encrypting it with the sender's private key (for message integrity and authenticity).
- Attach the encrypted session key to the encrypted message and digest.
- Transmit the message to the recipient.

The recipient reverses these steps to retrieve and validate the message content.

S/MIME:

An Internet standard governs how e-mail is sent and received. The general MIME specification defines the format and handling of e-mail attachments. S/MIME (Secure Multipurpose Internet Mail Extensions) is the Internet standard for secure e-mail attachments. S/MIME is very much like PGP and its predecessors, PEM (Privacy-Enhanced Mail) and RIPEM. S/MIME has been adopted in commercial e-mail packages, such as Eudora and Microsoft Outlook. The principal difference between S/MIME and PGP is the method of key exchange. Basic PGP depends on each user's exchanging keys with all potential recipients and establishing a ring of trusted recipients; it also requires establishing a degree of trust in the

authenticity of the keys for those recipients. S/MIME uses hierarchically validated certificates, usually represented in X.509 format, for key exchange. Thus, with S/MIME, the sender and recipient do not need to have exchanged keys in advance if they have a common certifier they both trusts.

S/MIME works with a variety of cryptographic algorithms, such as DES, AES, and RC2 for symmetric encryption. S/MIME performs security transformations very similar to those for PGP. PGP was originally designed for plaintext messages, but S/MIME handles (secures) all sorts of attachments, such as data files (for example, spreadsheets, graphics, presentations, movies, and sound). Because it is integrated into many commercial e-mail packages, S/MIME is likely to dominate the secure e-mail market.

11.6 EXAMPLE PROTOCOLS

Much of the software currently used to protect the confidentiality of information are not true cryptosystems. Instead, they are applications to which cryptographic protocols have been added. This is perhaps particularly true of Internet protocols; some experts claim that the Internet and its corresponding protocols were designed without any consideration for security, which was added later as an afterthought. Whether or not this is true, the lack of threats in the environment in which it was launched allowed the Internet to grow rapidly. But as the number of threats grew, so did the need for additional security measures.

11.6.1 SSL:

Netscape developed the **Secure Sockets Layer (SSL)** protocol to use public key encryption to secure a channel over the Internet, thus enabling secure communications. Most popular browsers, including Internet Explorer, use SSL. In addition to providing data encryption, integrity, and server authentication, SSL can, when properly configured, provide client authentication.

The SSL protocol works as follows: during a normal client/server HTTP session, the client requests access to a portion of the Web site that requires secure communications, and the server sends a message to the client indicating that a secure connection must be established. The client sends its public key and security parameters. This handshaking phase is complete when the server finds a public key match and sends a digital certificate to the client in order to authenticate itself. Once the client verifies that the certificate is valid and trustworthy, the SSL session is established. Until the client or the server terminates the session, any amount of data can be transmitted securely.

SSL provides two protocol layers within the TCP framework: SSL Record Protocol and Standard HTTP. The **SSL Record Protocol** is responsible for the fragmentation, compression, encryption, and attachment of an SSL header to the plaintext prior to transmission. Received encrypted messages are decrypted and reassembled for presentation to the higher levels of the protocol. The SSL Record Protocol provides basic security and communication services to the top levels of the SSL protocol stack.

11.6.2 PEM:

Several cryptosystems have been adapted to work with the dominant e-mail protocols to incorporate some degree of security into this notoriously insecure communication medium. Some of the more popular adaptations included Secure Multipurpose Internet Mail Extensions, Privacy Enhanced Mail (PEM), and Pretty Good Privacy (PGP). Secure Multipurpose Internet Mail Extensions (S/MIME) builds on the encoding format of the Multipurpose Internet Mail Extensions (MIME) protocol and uses digital signatures based on public key cryptosystems to secure e-mail. Privacy Enhanced Mail (PEM) was proposed by the Internet Engineering Task Force (IETF) and is a standard that uses DES3 symmetric key encryption and RSA for key exchanges and digital signatures. Pretty Good Privacy (PGP) was developed by Phil Zimmermann and uses the IDEA cipher for message encoding. PGP also uses RSA for symmetric key exchange and digital signatures.

PEM employs a range of cryptographic techniques to allow for confidentiality, sender authentication, and message integrity. The message integrity aspects allow the user to ensure that a message hasn't been modified during transport from the sender. The sender authentication allows a user to verify that the PEM message that they have received is truly from the person who claims to have sent it. The confidentiality feature allows a message to be kept secret from people to whom the message was not addressed.

11.6.3 IPSec:

Internet Protocol Security (IPSec) is an open-source protocol framework for security development within the TCP/IP family of protocol standards. It is used to secure communications across IP-based networks such as LANs, WANs, and the Internet. The protocol is designed to protect data integrity, user confidentiality, and authenticity at the IP packet level. IPSec is the cryptographic authentication and encryption product of the IETF's IP Protocol Security Working Group. It is often described as the security system from IP version 6 (the future version of the TCP/IP protocol), retrofitted for use with IP version 4 (the current version). IPSec is

defined in Request for Comments (RFC) 1825, 1826, and 1827 and is widely used to create virtual private networks (VPNs). IPSec itself is an open framework. IPSec includes the IP Security protocol itself, which specifies the information to be added to an IP packet as well as how to encrypt packet data; and the Internet Key Exchange, which uses an asymmetric-based key exchange and negotiates the security associations. IPSec operates in two modes: transport and tunnel. In **transport mode** only the IP data are encrypted, not the IP headers. This allows intermediate nodes to read the source and destination addresses. In **tunnel mode** the entire IP packet is encrypted and is then placed into the content portion of another IP packet. This requires other systems at the beginning and end of the tunnel to act as proxies and to send and receive the encrypted packets. These systems then transmit the decrypted packets to their true destinations. IPSec uses several different cryptosystems:

- Diffie-Hellman key exchange for deriving key material between peers on a public network
- Public key cryptography for signing the Diffie-Hellman exchanges to guarantee the identity of the two parties
- Bulk encryption algorithms, such as DES, for encrypting the data
- Digital certificates signed by a certificate authority to act as digital ID cards

Within IPSec, IP layer security is achieved by means of an application header protocol or an encapsulating security payload protocol. The **application header (AH) protocol** provides system-to-system authentication and data integrity verification but does not provide secrecy for the content of a network communication. The **encapsulating security payload (ESP) protocol** provides secrecy for the contents of network communications as well as system-to-system authentication and data integrity verification. When two networked systems form an association that uses encryption and authentication keys, algorithms, and key lifetimes, they can implement either the AH or the ESP protocol, but not both.

The AH protocol is designed to provide data integrity and IP packet authentication. Although AH does not provide confidentiality protection, IP packets are protected from replay attacks and address spoofing as well as other types of cyberattacks against open networks.

The encapsulating security payload protocol provides confidentiality services for IP packets across insecure networks. ESP can also provide the authentication services of AH. ESP in tunnel

mode can be used to establish a virtual private network, assuring encryption and authentication between networks communicating via the Internet. In tunnel mode, the entire IP packet is encrypted with the attached ESP header. A new IP header is attached to the encrypted payload, providing the required routing information. Figure 10-1 shows the packet format of the IPsec AH and ESP protocol.

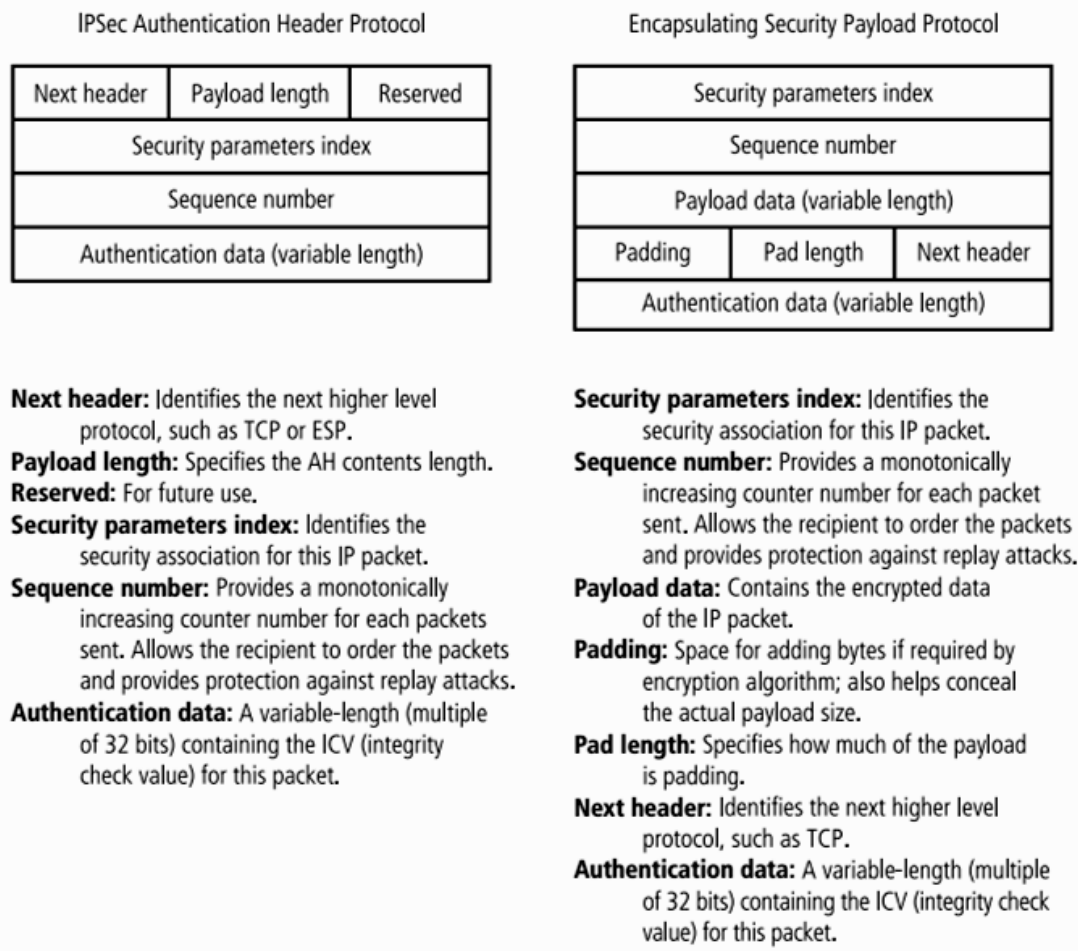


Figure 11-1 IPsec Headers

11.7 SUMMARY

Security in networks is the combination and culmination of everything we know about security, and certainly everything we have discussed in this book so far. A network's security depends on all the cryptographic tools at our disposal, good program development processes, operating system controls, trust and evaluation and assurance methods, and inference and aggregation controls.

Networks and their security remind us that good software engineering practices can go a long way toward making software difficult to attack. When a network and its components are structured,

designed, and architected well, the resulting system presents solid defences and avoids potential single points of failure. And a well-engineered network is easy to change as it evolves; because it is easier to understand, changes seldom introduce unintentional flaws.

Many of the controls useful for stand-alone systems are also useful in networks. But three controls are specific to networks: firewalls, intrusion detection systems, and secure e-mail. These controls have evolved from many years of research, both in security and in other computer science realms. They emphasize why we should know not only the history of security but also the relevance of other computing research. For example, firewalls are just an updated form of reference monitor. Similarly, intrusion detection profits from more fundamental research into pattern matching and expert systems. And secure e-mail is really a carefully designed application of cryptography. You might think that controls such as these are the result of strokes of genius. But in fact, they reflect the long-term nature of knowledge and engineering practice; new ways to provide security build on a growing base of understanding and experience.

Until now we have stressed technical controls, which can be very effective in protecting our computing assets. But many security losses come from trusted insiders either honest people making honest, human mistakes or dishonest insiders able to capitalize on their knowledge or privileges. In the next chapter we consider administrative controls, such as security policies, user awareness, and risk analysis, to address the insider threat.

11.8 REVIEW QUESTIONS

- a) What are the key architecture controls in network security?
- b) What is Link encryption and end-to-end encryption? Compare them.
- c) How can content integrity be maintained?
- d) Write a short note on PKI and certificates.
- e) Write a short note on Kerberos.
- f) What measures are required for Wireless security?
- g) Explain Honeypots.
- h) What is firewall? Explain the different types of firewalls.
- i) Compare the different types of firewalls.
- j) What can and cannot be blocked by firewalls?
- k) What are IDS? Explain the types of IDSs.
- l) What are the goals for IDSs?
- m) Write a short note on Secure E-mail.

n) Write a short note on IPSec protocol.

11.9 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- *Security in Computing* by C. P. Pfleeger, and S. L. Pfleeger, Pearson Education.
- *Computer Security: Art and Science* by Matt Bishop, Pearson Education.
- *Cryptography and Network Security: Principles and practice* by Stallings
- *Network Security* by Kaufman, Perlman, Speciner
- *Network Security : A Beginner's Guide* by Eric Maiwald, TMH
- *Java Network Security* by Macro Pistoia, Pearson Education
- *Principles of information security* by Whitman, Mattord, Thomson

SECURITY PLANNING AND RISK ANALYSIS

Unit Structure:

- 12.0 Objectives
- 12.1 Introduction
- 12.2 Security Planning
 - 12.2.1. Contents of a Security Plan
 - 12.2.2. Security Planning Team Members
 - 12.2.3. Assuring commitment to a Security Plan
 - 12.2.4. Business Continuity Plans
 - 12.2.5. Incident Response Plans
- 12.3 Risk Analysis
 - 12.3.1. The Nature of Risk
 - 12.3.2. Steps of a Risk Analysis
 - 12.3.3. Arguments for and against risk analysis
- 12.4 Summary
- 12.5 Review Questions
- 12.6 Bibliography, References and Further Reading

12.0 OBJECTIVES

In this chapter, we move above the technical knowledge and introduce the management aspects of security planning and risk analysis.

12.1 INTRODUCTION

Security is a combination of technical, administrative, and physical controls, as we first pointed out in the earlier chapters. So far, we have considered technical controls almost exclusively. But stop and think for a moment: What good is a firewall if there is no power to run it? How effective is a public key infrastructure if someone can walk off with the certificate server? And why have

elaborate access control mechanisms if your employee mails a sensitive document to a competitor? The administrative and physical controls may be less glamorous than the technical ones, but they are surely as important. In this and the next chapter we complete our study of security controls by considering administrative and physical aspects. We look at four related areas:

- *Planning*: What advance preparation and study lets us know that our implementation meets our security needs for today and tomorrow?
- *Risk analysis*: How do we weigh the benefits of controls against their costs, and how do we justify any controls?
- *Policy*: How do we establish a framework to see that our computer security needs continue to be met?
- *Physical control*: What aspects of the computing environment have an impact on security?

These four areas are just as important to achieving security as are the latest firewall or coding practice.

12.2 SECURITY PLANNING

Years ago, when most computing was done on mainframe computers, data processing centres were responsible for protection. Responsibility for security rested neither with the programmers nor the users but instead with the computing centres themselves. These centres developed expertise in security, and they implemented many protection activities in the background, without users having to be conscious of protection needs and practices. Since the early 1980s, the introduction of personal computers and the general ubiquity of computing have changed the way many of us work and interact with computers. A significant amount of the responsibility for security has shifted to the user and away from the computing center. But many users are unaware of (or choose to ignore) this responsibility, so they do not deal with the risks posed or do not implement simple measures to prevent or mitigate problems. Unfortunately, there are many common examples of this neglect. Moreover, it is exacerbated by the seemingly hidden nature of important data: Things we would protect if they were on paper are ignored when they are stored electronically. For example, a person who carefully locks up paper copies of company confidential records overnight may leave running a personal computer or terminal on an assistant's or manager's desk. In this situation, a curious or malicious person walking past can retrieve confidential memoranda and data. Similarly, the data on laptops and workstations are often more easily available than on older, more isolated systems. For instance, the large and cumbersome disk packs and tapes from a few years ago have been replaced by media such as diskettes, zip disks, and CDs, which hold

a similar volume of data but fit easily in a pocket or briefcase. Moreover, we all recognize that a box of CDs or diskettes may contain many times more data than a printed report. But since the report is an apparent, visible exposure and the CD or diskette is not, we leave the computer media in plain view, easy to borrow or steal. In all cases, whether the user initiates some computing action or simply interacts with an active application, every application has confidentiality, integrity, and availability requirements that relate to the data, programs, and computing machinery. In these situations, users suffer from lack of sensitivity: They often do not appreciate the security risks associated with using computers.

For these reasons, every organization using computers to create, and store valuable assets should perform thorough and effective security planning. A security plan is a document that describes how an organization will address its security needs. The plan is subject to periodic review and revision as the organization's security needs change. A good security plan is an official record of current security practices, plus a blueprint for orderly change to improve those practices. By following the plan, developers and users can measure the effect of proposed changes, leading eventually to further improvements. The impact of the security plan is important, too. A carefully written plan, supported by management, notifies employees that security is important to management. Thus, the security plan must have the appropriate content and produce the desired effects.

In this section we focus on three aspects of writing a security plan: what it should contain, who writes it, and how to obtain support for it. Then, we address two specific cases of security plans: business continuity plans, to ensure that an organization continues to function despite a computer security incident, and incident response plans, to organize activity to deal with the crisis of an incident.

12.2.1 Contents of a Security Plan:

A security plan identifies and organizes the security activities for a computing system. The plan is both a description of the current situation and a plan for improvement. Every security plan must address seven issues.

1. **policy**, indicating the goals of a computer security effort and the willingness of the people involved to work to achieve those goals
2. **current state**, describing the status of security at the time of the plan
3. **requirements**, recommending ways to meet the security goals
4. **recommended controls**, mapping controls to the vulnerabilities

identified in the policy and requirements

5. **accountability**, describing who is responsible for each security activity
6. **timetable**, identifying when different security functions are to be done
7. **continuing attention**, specifying a structure for periodically updating the security plan

There are many approaches to creating and updating a security plan. Some organizations have a formal, defined security planning process, much as they might have a defined and accepted development or maintenance process. Others look to security professionals for guidance on how to perform security planning. But every security plan contains the same basic material, no matter the format. The following sections expand on the seven parts of a security plan.

1. Policy:

A security plan must state the organization's policy on security. A security policy is a high-level statement of purpose and intent. Initially, you might think that all policies would be the same: to prevent security breaches. But in fact, the policy is one of the most difficult sections to write well. As we discuss later in this chapter, there are trade-offs among the strength of the security, the cost, the inconvenience to users, and more. For example, we must decide whether to implement very stringent and possibly unpopular controls that prevent all security problems or simply mitigate the effects of security breaches once they happen. For this reason, the policy statement must answer three essential questions:

- *Who* should be allowed access?
- To what system and organizational *resources* should access be allowed?
- What *types* of access should each user be allowed for each resource?

The policy statement should specify the following:

- The organization's *goals* on security. For example, should the system protect data from leakage to outsiders, protect against loss of data due to physical disaster, protect the data's integrity, or protect against loss of business when computing resources fail? What is the higher priority: serving customers or securing data?
- Where the *responsibility* for security lies. For example, should the responsibility rest with a small computer security group, with each employee, or with relevant managers?

- The organization's *commitment* to security. For example, who provides security support for staff, and where does security fit into the organization's structure?

2. Current Security Status:

To be able to plan for security, an organization must understand the vulnerabilities to which it may be exposed. The organization can determine the vulnerabilities by performing a risk analysis: a careful investigation of the system, its environment, and the things that might go wrong. The risk analysis forms the basis for describing the current status of security. The status can be expressed as a listing of organizational assets, the security threats to the assets, and the controls in place to protect the assets. The status portion of the plan also defines the limits of responsibility for security. It describes not only which assets are to be protected but also who is responsible for protecting them. The plan may note that some groups may be excluded from responsibility; for example, joint ventures with other organizations may designate one organization to provide security for all member organizations. The plan also defines the boundaries of responsibility, especially when networks are involved. For instance, the plan should clarify who provides the security for a network router or for a leased line to a remote site. Even though the security plan should be thorough, there will necessarily be vulnerabilities that are not considered. These vulnerabilities are not always the result of ignorance or naïveté; rather, they can arise from the addition of new equipment or data as the system evolves. They can also result from new situations, such as when a system is used in ways not anticipated by its designers. The security plan should detail the process to be followed when someone identifies a new vulnerability. Instructions should explain how to integrate controls for that vulnerability into the existing security procedures.

3. Requirements:

The heart of the security plan is its set of security requirements: functional or performance demands placed on a system to ensure a desired level of security. The requirements are usually derived from organizational needs. Sometimes these needs include the need to conform to specific security requirements imposed from outside, such as by a government agency or a commercial standard. A constraint is an aspect of the security policy that constrains, circumscribes, or directs the implementation of the requirements. A control is an action, device, procedure, or technique that removes or reduces a vulnerability. To see the difference between requirements, constraints, and controls, consider the six "requirements" of the U.S. Department of Defence's TCSEC, introduced earlier. These six items are listed below in Table 12-1.

Table 12-1. The Six "Requirements" of the TCSEC.

Security policy	There must be an explicit and well-defined security policy enforced by the system.
Identification	Every subject must be uniquely and convincingly identified. Identification is necessary so that subject/object access can be checked.
Marking	Every object must be associated with a label that indicates its security level. The association must be done so that the label is available for comparison each time an access to the object is requested.
Accountability	The system must maintain complete, secure records of actions that affect security. Such actions include introducing new users to the system, assigning or changing the security level of a subject or an object, and denying access attempts.
Assurance	The computing system must contain mechanisms that enforce security, and it must be possible to evaluate the effectiveness of these mechanisms.
Continuous protection	The mechanisms that implement security must be protected against unauthorized change.

Given our definitions of requirement, constraint, and control, it is easy to see that the first "requirement" of the TCSEC is really a constraint: the security policy. The second and third "requirements" describe mechanisms for enforcing security, not descriptions of required behaviours. That is, the second and third "requirements" describe explicit implementations, not a general characteristic or property that the system must have. However, the fourth, fifth, and sixth TCSEC "requirements" are indeed true requirements. They state that the system must have certain characteristics, but they do not enforce an implementation. These distinctions are important because the requirements explain *what* should be accomplished, not *how*. That is, the requirements should always leave the implementation details to the designers, whenever possible. For example, rather than writing a requirement that certain data records should require passwords for access (an implementation decision), a security planner should state only that access to the data records should be restricted (and note to whom the access should be restricted). This more flexible requirement allows the designers to decide among several other access controls (such as access control lists) and to balance the security requirements with other system requirements, such as performance and reliability. Figure 12-1 illustrates how the different aspects of system analysis support the security planning process.

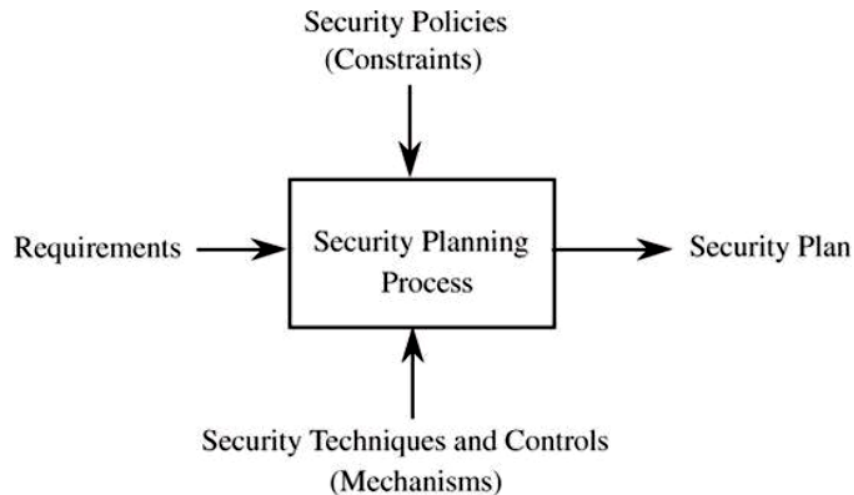


Figure 12-1. Inputs to the Security Plan.

As with the general software development process, the security planning process must allow customers or users to specify desired functions, independent of the implementation. The requirements should address all aspects of security: confidentiality, integrity, and availability. They should also be reviewed to make sure that they are of appropriate quality. We should make sure that the requirements have these characteristics:

- *Correctness*: Are the requirements understandable? Are they stated without error?
- *Consistency*: Are there any conflicting or ambiguous requirements?
- *Completeness*: Are all possible situations addressed by the requirements?
- *Realism*: Is it possible to implement what the requirements mandate?
- *Need*: Are the requirements unnecessarily restrictive?
- *Verifiability*: Can tests be written to demonstrate conclusively and objectively that the requirements have been met? Can the system or its functionality be measured in some way that will assess the degree to which the requirements are met?
- *Traceability*: Can each requirement be traced to the functions and data related to it so that changes in a requirement can lead to easy re-evaluation?

The requirements may then be constrained by budget, schedule, performance, policies, governmental regulations, and more. Given the requirements and constraints, the developers then choose appropriate controls.

4. Recommended Controls:

The security requirements lay out the system's needs in terms of what should be protected. The security plan must also recommend what controls should be incorporated into the system to meet those requirements. Throughout this book you have seen many examples of controls, so we need not review them here. As we see later in this chapter, we can use risk analysis to create a map from vulnerabilities to controls. The mapping tells us how the system will meet the security requirements. That is, the recommended controls address implementation issues: how the system will be designed and developed to meet stated security requirements.

5. Responsibility for Implementation:

A section of the security plan should identify which people are responsible for implementing the security requirements. This documentation assists those who must coordinate their individual responsibilities with those of other developers. At the same time, the plan makes explicit who is accountable should some requirement not be met, or some vulnerability not be addressed. That is, the plan notes who is responsible for implementing controls when a new vulnerability is discovered, or a new kind of asset is introduced. People building, using, and maintaining the system play many roles. Each role can take some responsibility for one or more aspects of security. Consider, for example, the groups listed here.

- *Personal computer users* may be responsible for the security of their own machines. Alternatively, the security plan may designate one person or group to be coordinator of personal computer security.
- *Project leaders* may be responsible for the security of data and computations.
- *Managers* may be responsible for seeing that the people they supervise implement security measures.
- *Database administrators* may be responsible for the access to and integrity of data in their databases.
- *Information officers* may be responsible for overseeing the creation and use of data; these officers may also be responsible for retention and proper disposal of data.
- *Personnel staff members* may be responsible for security involving employees, for example, screening potential employees for trustworthiness and arranging security training programs.

6. Timetable:

A comprehensive security plan cannot be executed instantly. The security plan includes a timetable that shows how and when the elements of the plan will be performed. These dates also give milestones so that management can track the progress of implementation. If the implementation is to be a phased

Unedited Version: Information Security

development, the plan should also describe how the security requirements will be implemented over time. Even when overall development is not phased, it may be desirable to implement the security aspects of the system over time. For example, if the controls are expensive or complicated, they may be acquired and implemented gradually. Similarly, procedural controls may require staff training to ensure that everyone understands and accepts the reason for the control. The plan should specify the order in which the controls are to be implemented so that the most serious exposures are covered as soon as possible. A timetable also gives milestones by which to judge the progress of the security program. Furthermore, the plan must be extensible. Conditions will change: New equipment will be acquired, new degrees and modes of connectivity will be requested, and new threats will be identified. The plan must include a procedure for change and growth, so that the security aspects of changes are considered as a part of preparing for the change, not for adding security after the change has been made. The plan should also contain a schedule for periodic review. Even though there may have been no obvious, major growth, most organizations experience modest change every day. At some point the cumulative impact of the change is enough to require the plan to be modified.

7. Continuing Attention:

Good intentions are not enough when it comes to security. We must not only take care in defining requirements and controls, but we must also find ways for evaluating a system's security to be sure that the system is as secure as we intend it to be. Thus, the security plan must call for reviewing the security situation periodically. As users, data, and equipment change, new exposures may develop. In addition, the current means of control may become obsolete or ineffective. The inventory of objects and the list of controls should periodically be scrutinized and updated, and risk analysis performed anew. The security plan should set times for these periodic reviews, based either on calendar time (such as, review the plan every nine months) or on the nature of system changes (such as, review the plan after every major system release).

12.2.2 Security Planning Team Members:

Who performs the security analysis, recommends a security program, and writes the security plan? As with any such comprehensive task, these activities are likely to be performed by a committee that represents all the interests involved. The size of the committee depends on the size and complexity of the computing organization and the degree of its commitment to security. Organizational behaviour studies suggest that the optimum size for a working committee is between five and nine members. Sometimes a larger committee may serve as an oversight body to review and comment on the products of a smaller working committee.

Alternatively, a large committee might designate subcommittees to address various sections of the plan. Security in operating systems and networks requires the cooperation of the systems administration staff. Program security measures can be understood and recommended by applications programmers. Physical security controls are implemented by those responsible for general physical security, both against human attacks and natural disasters. Finally, because controls affect system users, the plan should incorporate users' views, especially about usability and the general desirability of controls. Thus, no matter how it is organized, a security planning team should represent each of the following groups:

- computer hardware group
- system administrators
- systems programmers
- applications programmers
- data entry personnel
- physical security personnel
- representative users

In some cases, a group can be adequately represented by someone who is consulted at appropriate times, rather than a committee member from each possible constituency being enlisted.

12.2.3 Assuring Commitment to a Security Plan:

After the plan is written, it must be accepted, and its recommendations carried out. Acceptance by the organization is key; a plan that has no organizational commitment is simply a plan that collects dust on the shelf. Commitment to the plan means that security functions will be implemented, and security activities carried out. Three groups of people must contribute to making the plan a success.

- The planning team must be sensitive to the needs of each group affected by the plan.
- Those affected by the security recommendations must understand what the plan means for the way they will use the system and perform their business activities. They must see how what they do can affect other users and other systems.
- Management must be committed to using and enforcing the security aspects of the system.

Education and publicity can help people understand and accept a security plan. Acceptance involves not only the letter but also the spirit of the security controls. If people understand the need for recommended controls and accept them as sensible, they will use the controls properly and effectively. If people think the controls are bothersome, capricious, or counterproductive, they will work to avoid

or subvert them. Management commitment is obtained through understanding. But this understanding is not just a function of what makes sense technologically; it also involves knowing the cause and the potential effects of lack of security. Managers must also weigh trade-offs in terms of convenience and cost. The plan must present a picture of how cost effective the controls are, especially when compared to potential losses if security is breached without the controls. Thus, proper presentation of the plan is essential, in terms that relate to management as well as technical concerns. Remember that some managers are not computing specialists. Instead, the system supports a manager who is an expert in some other business function, such as banking, medical technology, or sports. In such cases, the security plan must present security risks in language that the managers understand. Sometimes outside experts can bridge the gap between the managers' business and security. Management is often reticent to allocate funds for controls until the value of those controls is explained. The results of a risk analysis can help communicate the financial trade-offs and benefits of implementing controls. By describing vulnerabilities in financial terms and in the context of ordinary business activities (such as leaking data to a competitor or an outsider), security planners can help managers understand the need for controls.

The plans we have just discussed are part of normal business. They address how a business handles computer security need. Similar plans might address how to increase sales or improve product quality, so these planning activities should be a natural part of management. Next, we turn to two kinds of business plans that address specific security problems: coping with and controlling activity during security incidents.

12.2.4 Business Continuity Plans:

Small companies working on a low profit margin can literally be put out of business by a computer incident. Large, financially sound businesses can weather a modest incident that interrupts their use of computers for a while, although it is painful to them. But even rich companies do not want to spend money unnecessarily. The analysis is sometimes as simple as *no computers means no customers means no sales means no profit*. Government agencies, educational institutions, and non-profit organizations also have limited budgets, which they want to use to further their needs. They may not have a direct profit motive but being able to meet the needs of their customers, the public, students, and constituents partially determines how well they will fare in the future. All kinds of organizations must plan for ways to cope with emergency situations.

A business continuity plan documents how a business will continue to function during a computer security incident. An ordinary

Unedited Version: Information Security

security plan covers computer security during normal times and deals with protecting against a wide range of vulnerabilities from the usual sources. A business continuity plan deals with situations having two characteristics:

- *catastrophic situations*, in which all or a major part of a computing capability is suddenly unavailable
- *long duration*, in which the outage is expected to last for so long that business will suffer

There are many situations in which a business continuity plan would be helpful. Here are some examples that typify what you might find in reading your daily newspaper:

- A fire destroys a company's entire network.
- A seemingly permanent failure of a critical software component renders the computing system unusable.
- A business must deal with the abrupt failure of its supplier of electricity, telecommunications, network access, or other critical service.
- A flood prevents the essential network support staff from getting to the operations centre.

As you can see, these examples are likely to recur, and each disables a vital function. You may also have noticed how often "the computer" is blamed for an inability to provide a service or product. For instance, the clerk in a shop is unable to use the cash register because "the computer is down." You may have a CD in your hand, plus exactly the cash to pay for it. But the clerk will not take your money and send you on your way. Often, computer service is restored shortly. But sometimes it is not. The key to coping with such disasters is advance planning and preparation, identifying activities that will keep a business viable when the computing technology is disabled. The steps in business continuity planning are these:

- Assess the business impact of a crisis.
- Develop a strategy to control impact.
- Develop and implement a plan for the strategy

Assess Business Impact:

To assess the impact of a failure on your business, you begin by asking two key questions:

- What are the *essential assets*? What are the things that will prevent the business from doing business? Answers are typically of the form "the network," "the customer reservations database," or "the system controlling traffic lights."
- What could *disrupt* use of these assets? The vulnerability is more

important than the threat agent. For example, whether destroyed by a fire or zapped in an electrical storm, the network is nevertheless down. Answers might be "failure," "corrupted," or "loss of power."

You probably will find only a handful of key assets when doing this analysis. Do not overlook people and the things they need for support, such as documentation and communications equipment. Another way to think about your assets is to ask yourself, "What is the minimum set of things or activities needed to keep business operational, at least to some degree?" If a manual system would compensate for a failed computer system, albeit inefficiently, you may want to consider building such a manual system as a potential critical asset.

Develop Strategy:

The continuity strategy investigates how the key assets can be safeguarded. In some cases, a backup copy of data or redundant hardware or an alternative manual process is good enough. Sometimes, the most reasonable answer is reduced capacity. Ideally, you would like to continue business with no loss. But with catastrophic failures, usually only a portion of the business function can be preserved. In this case, you must develop a strategy appropriate for your business and customers. For instance, you can decide whether it is better to preserve half of function A and half of B, or most of A and none of B. You also must consider the time frame in which business is done. Some catastrophes last longer than others. For example, rebuilding after a fire is a long process and implies a long time in disaster mode. Your strategy may have several steps, each dependent on how long the business is disabled. Thus, you may take one action in response to a one-hour outage, and another if the outage might last a day or longer. Because you are planning, you have the luxury of being able to think about possible circumstances and evaluate alternatives. The result of a strategy analysis is a selection of the best actions, organized by circumstances. The strategy can then be used as the basis for your business continuity plan.

Develop Plan:

The business continuity plan specifies several important things:

- who is in charge when an incident occurs?
- what to do
- who does it?

The plan justifies making advance arrangements, such as acquiring redundant equipment, arranging for data backups, and

stockpiling supplies, before the catastrophe. The plan also justifies advance training so that people know how they should react. In a catastrophe there will be confusion; you do not want to add confused people to the already severe problem. The person in charge declares the state of emergency and instructs people to follow the procedures documented in the plan. The person in charge also declares when the emergency is over, and conditions can revert to normal. Thus, the business continuity planning addresses how to maintain some degree of critical business activity despite a catastrophe. Its focus is on keeping the business viable. It is based on the asset survey, which focuses on only a few critical assets and serious vulnerabilities that could threaten operation for a long or undetermined period. The focus of the business continuity plan is to keep the business going while someone else addresses the crisis. That is, the business continuity plan does not include calling the fire department or evacuating the building, important though those steps are. The focus of a business continuity plan is the *business* and how to keep it functioning to the degree possible in the situation. Handling the emergency is someone else's problem.

12.2.5 Incident Response Plans:

An incident response plan tells the staff how to deal with a security incident. In contrast to the business continuity plan, the goal of incident response is handling the current security incident, without regard for the business issues. The security incident may at the same time be a business catastrophe, as addressed by the business continuity plan. But as a specific security event, it might be less than catastrophic but could be a serious breach of security, such as a hacker attack or a case of internal fraud. An incident could be a single event, a series of events, or an ongoing problem. An incident response plan should

- define what constitutes an *incident*
- identify who is responsible for *taking charge* of the situation
- describe the plan of *action*

The plan usually has three phases: advance planning, triage, and running the incident. A fourth phase, review, is useful after the situation abates so that this incident can lead to improvement for future incidents.

Advance Planning:

As with all planning functions, advance planning works best because people can think logically, unhurried, and without pressure. What constitutes an incident may be vague. We cannot know the details of an incident in advance. Typical characteristics include harm or risk of harm to computer systems, data, or processing; initial uncertainty as to the extent of damage; and similar uncertainty as to the source or method of the incident. For example, you can see that the file is missing, or the home page has been defaced, but you do

not know how or by whom or what other damage there may be. In organizations that have not done incident planning, chaos may develop at this point. Someone calls the network manager. Someone sends e-mail to the help desk. Someone calls the FBI, the CERT, the newspapers, or the fire department. People start to investigate on their own, without coordinating with the relevant staff in other departments, agencies, or businesses. And there is a lot of conversation, rumour, and misinformation: more heat than light. With an incident response plan in place, everybody is trained in advance to call the designated leader. There is an established list of people to call, in order, in case the first person is unavailable. The leader decides what to do next, and he or she begins by determining if this is a real incident or a false alarm. Indeed, natural events sometimes look like incidents, and the facts of the situation should be established first. If the leader decides this may be a real incident, he or she invokes the response team.

Response Team:

The response team is the set of people charged with responding to the incident. The response team may include:

- *director*: person in charge of the incident, who decides what actions to take and when to terminate the response. The director is typically a management employee.
- *lead technician*: person who directs and coordinates the response. The lead technician decides where to focus attention, analyzes situation data, documents the incident and how it was handled, and calls for other technical people to assist with the analysis.
- *advisor(s)*: legal, human resources, or public relations staff members as appropriate.

In a small incident a single person can handle more than one of these roles. Nevertheless, it is important that a single person be in charge, a single person who directs the response work, a single point of contact for "insiders" (employees, users), and a single point of contact for "the public." To develop policy and identify a response team, you need to consider certain matters:

- *Legal issues*: An incident has legal ramifications. In some countries, computer intrusions are illegal, so law enforcement officials must be involved in the investigation. In other places, you have discretion in deciding whether to ask law enforcement to participate. In addition to criminal action, you may be able to bring a civil case. Both kinds of legal action have serious implications for the response. For example, evidence must be gathered and maintained in specific ways to be usable in court. Similarly, laws may limit what you can do against the alleged attacker: Cutting off a connection is probably acceptable but launching a retaliatory

denial-of-service attack may not be.

- *Preserving evidence:* The most common reaction in an incident is to assume the cause was internal or accidental. For instance, you may surmise that the hardware has failed or that the software isn't working correctly. The staff may be directed to change the configuration, reload the software, reboot the system, or similarly attempt to resolve the problem by adjusting the software. Unfortunately, each of these acts can irreparably distort or destroy evidence. When dealing with a possible incident, do as little as possible before "dusting for fingerprints."
- *Records:* It may be difficult to remember what you have already done: Have you already reloaded a file? What steps got you to the prompt asking for the new DNS server's address? If you call in an outside forensic investigator or the police, you will need to tell exactly what you have already done.
- *Public relations:* In handling an incident your organization should speak with one voice. You risk sending confusing messages if too many people speak. It is especially important that only one person speak publicly if legal action may be taken. An unguarded comment may tip off the attacker or have a negative effect on the case. You can simply say that an incident occurred, tell briefly and generally what it was, and state that the incident is now under control and normal operation is resuming.

After the Incident Is Resolved:

Eventually, the incident response team closes the case. At this point it will hold a review after the incident to consider two things:

- *Is any security control action to be taken?* Did an intruder compromise a system because security patches were not up to date; if so, should there be a procedure to ensure that patches are applied when they become available? Was access obtained because of a poorly chosen password; if so, should there be a campaign to educate users on how to strong passwords? If there were control failures, what should be done to prevent similar attacks in the future?
- *Did the incident response plan work?* Did everyone know whom to notify? Did the team have needed resources? Was the response fast enough? What should be done differently next time? The incident response plan ensures that incidents are handled promptly, efficiently, and with minimal harm.

12.3 RISK ANALYSIS

Good, effective security planning includes a careful risk analysis. A risk is a potential problem that the system or its users may experience. We distinguish a risk from other project events by looking for three things:

- *A loss associated with an event.* The event must generate a negative effect: compromised security, lost time, diminished quality, lost money, lost control, lost understanding, and so on. This loss is called the risk impact.
- *The likelihood that the event will occur.* The probability of occurrence associated with each risk is measured from 0 (impossible) to 1 (certain). When the risk probability is 1, we say we have a problem.
- *The degree to which we can change the outcome.* We must determine what, if anything, we can do to avoid the impact or at least reduce its effects. Risk control involves a set of actions to reduce or eliminate the risk.

We usually want to weigh the pros and cons of different actions we can take to address each risk. To that end, we can quantify the effects of a risk by multiplying the risk impact by the risk probability, yielding the risk exposure. For example, if the likelihood of virus attack is 0.3 and the cost to clean up the affected files is \$10,000, then the risk exposure is \$3,000. So, we can use a calculation like this one to decide that a virus checker is worth an investment of \$100, since it will prevent a much larger potential loss. Clearly, risk probabilities can change over time, so it is important to track them and plan for events accordingly.

Risk is inevitable in life: Crossing the street is risky but that does not keep us from doing it. We can identify, limit, avoid, or transfer risk but we can seldom eliminate it. In general, we have three strategies for dealing with risk:

- *avoiding* the risk, by changing requirements for security or other system characteristics
- *transferring* the risk, by allocating the risk to other systems, people, organizations, or assets; or by buying insurance to cover any financial loss should the risk become a reality
- *assuming* the risk, by accepting it, controlling it with available resources, and preparing to deal with the loss if it occurs

Thus, costs are associated not only with the risk's potential impact but also with reducing it. Risk leverage is the difference in risk exposure divided by the cost of reducing the risk. In other words, risk leverage is

$$\frac{(\text{risk exposure before reduction}) - (\text{risk exposure after reduction})}{(\text{cost of risk reduction})}$$

If the leverage value of a proposed action is not high enough, then we look for alternative but less costly actions or more effective reduction techniques. Risk analysis is the process of examining a system and its operational context to determine possible exposures and the potential harm they can cause. Thus, the first step in a risk analysis is to identify and list all exposures in the computing system of interest. Then, for each exposure, we identify possible controls and their costs. The last step is a cost benefit analysis: Does it cost less to implement a control or to accept the expected cost of the loss? In the remainder of this section, we describe risk analysis, present examples of risk analysis methods, and discuss some of the drawbacks to performing risk analysis.

12.3.1 The Nature of Risk:

In our everyday lives, we take risks. In crossing the road, eating oysters, or playing the lottery, we take the chance that our actions may result in some negative result such as being injured, getting sick, or losing money. Consciously or unconsciously, we weigh the benefits of taking the action with the possible losses that might result. Just because there is a risk to a certain act, we do not necessarily avoid it; we may look both ways before crossing the street, but we do cross it. In building and using computing systems, we must take a more organized and careful approach to assessing our risks. Many of the systems we build, and use can have a dramatic impact on life and health if they fail. For this reason, risk analysis is an essential part of security planning. We cannot guarantee that our systems will be risk free; that is why our security plans must address actions needed should an unexpected risk become a problem. And some risks are simply part of doing business; for example, as we have seen, we must plan for disaster recovery, even though we take many steps to avoid disasters in the first place. When we acknowledge that a significant problem cannot be prevented, we can use controls to reduce the seriousness of a threat. For example, you can back up files on your computer as a defence against the possible failure of a file storage device. But as our computing systems become more complex and more distributed, complete risk analysis becomes more difficult and time consuming and more essential.

12.3.2 Steps of a Risk Analysis:

Risk analysis is performed in many different contexts; for example, environmental and health risks are analyzed for activities such as building dams, disposing of nuclear waste, or changing a manufacturing process. Risk analysis for security is adapted from

more general management practices, placing special emphasis on the kinds of problems likely to arise from security issues. By following well-defined steps, we can analyze the security risks in a computing system. The basic steps of risk analysis are listed below:

- Identify assets.
- Determine vulnerabilities.
- Estimate likelihood of exploitation.
- Compute expected annual loss.
- Survey applicable controls and their costs.
- Project annual savings of control.

These steps are described in detail in the following sections:

Step 1: Identify Assets

Before we can identify vulnerabilities, we must first decide what we need to protect. Thus, the first step of a risk analysis is to identify the assets of the computing system. The assets can be considered in categories, as listed below:

- *hardware*: processors, boards, keyboards, monitors, terminals, microcomputers, workstations, tape drives, printers, disks, disk drives, cables, connections, communications controllers, and communications media
- *software*: source programs, object programs, purchased programs, in-house programs, utility programs, operating systems, systems programs (such as compilers), and maintenance diagnostic programs
- *data*: data used during execution, stored data on various media, printed data, archival data, update logs, and audit records
- *people*: skills needed to run the computing system or specific programs
- *documentation*: on programs, hardware, systems, administrative procedures, and the entire system
- *supplies*: paper, forms, laser cartridges, magnetic media, and printer fluid

It is essential to tailor this list to your own situation. No two organizations will have the same assets to protect, and something that is valuable in one organization may not be as valuable to another. For example, RAND Corporation's Vulnerability Assessment and Mitigation (VAM) methodology includes additional assets, such as:

- the enabling infrastructure
- the building or vehicle in which the system will reside
- the power, water, air, and other environmental conditions necessary for proper functioning
- human and social assets, such as policies, procedures, and

training

The VAM methodology is a process supported by a tool to help people identify assets, vulnerabilities, and countermeasures. In a sense, the list of assets is an inventory of the system, including intangibles and human resource items. For security purposes, this inventory is more comprehensive than the traditional inventory of hardware and software often performed for configuration management or accounting purposes. The point is to identify all assets necessary for the system to be usable.

Step 2: Determine Vulnerabilities:

The next step in risk analysis is to determine the vulnerabilities of these assets. This step requires imagination; we want to predict what damage might occur to the assets and from what sources. We can enhance our imaginative skills by developing a clear idea of the nature of vulnerabilities. This nature derives from the need to ensure the three basic goals of computer security: confidentiality, integrity, and availability. Thus, a vulnerability is any situation that could cause loss of confidentiality, integrity, and availability.

We want to use an organized approach to considering situations that could cause these losses for an object. Software engineering offers us several techniques for investigating possible problems. Hazard analysis explores failures that may occur and faults that may cause them. These techniques have been used successfully in analyzing safety-critical systems. However, additional techniques are tailored specifically to security concerns.

To organize the way, we consider threats and assets, we can use a matrix such as the one shown in Table 12-2. One vulnerability can affect more than one asset or cause more than one type of loss. The table is a guide to stimulate thinking, but its format is not rigid.

Table 12-2. Assets and Security Properties.

Asset	Confidentiality	Integrity	Availability
Hardware			
Software			
Data			
People			
Documentation			
Supplies			

In thinking about the contents of each matrix entry, we can ask the following questions:

- What are the effects of unintentional errors? Consider typing the wrong command, entering the wrong data, using the wrong data item, discarding the wrong listing, and disposing of output insecurely.
- What are the effects of wilfully malicious insiders? Consider disgruntled employees, bribery, and curious browsers.
- What are the effects of outsiders? Consider network access, dial-in access, hackers, people walking through the building, and people sifting through the trash.
- What are the effects of natural and physical disasters? Consider fires, storms, floods, power outages, and component failures.

Table 12-3 is a version of the previous table with some of the entries filled in. It shows that certain general problems can affect the assets of a computing system. In each installation, it is necessary to determine what can happen to specific hardware, software, data items, and other assets.

Table 12-3. Assets and Attacks.

Asset	Secrecy	Integrity	Availability
Hardware		overloaded destroyed tampered with	failed stolen destroyed unavailable
Software	stolen copied pirated	impaired by Trojan horse modified tampered with	deleted misplaced usage expired
Data	disclosed accessed by outsider inferred	damaged – software error – hardware error – user error	deleted misplaced destroyed
People			quit retired terminated on vacation
Documentation			lost stolen destroyed
Supplies			lost stolen damaged

Alas, there is no simple checklist or easy procedure to list all vulnerabilities. Tools can help us conceive of vulnerabilities by providing a structured way to think. For example, RAND's VAM methodology suggests that assets have certain properties that make them vulnerable. The properties exist in three categories: aspects of the design or architecture, aspects of behaviour, and general attributes. Table 12-4 lists these properties in more detail. Notice that the properties apply to many kinds of systems and at various places within a given system. These attributes can be used to build a matrix, each of whose entries may suggest one or more vulnerabilities. Using that matrix for example, the design attribute *limits, finiteness* applied to a *cyber object*, a *software program* could lead you to suspect buffer overflow vulnerabilities, or *uniqueness* for a *hardware object* could signal a single point of failure. To use this methodology, you would work through the matrix, thinking of each contributing attribute on each asset class to derive the set of vulnerabilities.

Table 12-4. Attributes Contributing to Vulnerabilities.

Design/Architecture	Behavioral	General
<ul style="list-style-type: none"> • Singularity <ul style="list-style-type: none"> – Uniqueness – Centrality – Homogeneity • Separability • Logic/implementation errors; fallibility • Design sensitivity, fragility, limits, finiteness • Unrecoverability 	<ul style="list-style-type: none"> • Behavioral sensitivity/fragility • Malevolence • Rigidity • Malleability • Gullibility, deceivability, naïveté • Complacency • Corruptibility, controllability 	<ul style="list-style-type: none"> • Accessible, detectable, identifiable, transparent, interceptable • Hard to manage or control • Self-unawareness and unpredictability • Predictability

Step 3: Estimate Likelihood of Exploitation:

The third step in conducting a risk analysis is determining how often each exposure is likely to be exploited. Likelihood of occurrence relates to the stringency of the existing controls and the likelihood that someone or something will evade the existing controls. There are several approaches to computing the probability that an event will occur: classical, frequency, and subjective. Each approach has its advantages and disadvantages, and we must choose the approach that best suits the situation (and its available information). In security, it is often not possible to directly evaluate an event's probability by using classical techniques. However, we can try to apply frequency probability by using observed data for a specific system. Local failure rates are easy to record, and we can identify which failures resulted in security breaches or created new vulnerabilities. Operating systems can track data on hardware failures, failed login attempts, numbers of accesses, and changes in the sizes of data files. Another alternative is to estimate the number of occurrences in each time period. We can ask an analyst familiar with the system to approximate the number of times a described event occurred in the last year, for example. Although the count is not exact, the analyst's knowledge of the system and its usage may yield reasonable estimates. Of course, the two methods described depend on the fact that a system is already built and has been in use for some period. In many cases, and especially for proposed systems, the usage data are not available. In this case, we may ask an analyst to estimate likelihood by reviewing a table based on a similar system; this approach is incorporated in several formal security risk processes.

Step 4: Compute Expected Loss:

By this time, we have gained an understanding of the assets we value, their possible vulnerabilities, and the likelihood that the vulnerabilities will be exploited. Next, we must determine the likely loss if the exploitation does indeed occur. As with likelihood of occurrence, this value is difficult to determine. Some costs, such as the cost to replace a hardware item, are easy to obtain. The cost to

replace a piece of software can be approximated reasonably well from the initial cost to buy it (or specify, design, and write it). However, we must take care to include hidden costs in our calculations. For instance, there is a cost to others of not having a piece of hardware or software. Similarly, there are costs in restoring a system to its previous state, reinstalling software, or deriving a piece of information. These costs are substantially harder to measure. In addition, there may be hidden costs that involve legal fees if certain events take place. For example, some data require protection for legal reasons. Personal data, such as police records, tax information, census data, and medical information, are so sensitive that there are criminal penalties for releasing the data to unauthorized people. Other data are company confidential; their release may give competitors an edge on new products or on likely changes to the stock price. Some financial data, especially when they reflect an adverse event, could seriously affect public confidence in a bank, an insurance company, or a stock brokerage. It is difficult to determine the cost of releasing these data. If a computing system, a piece of software, or a key person is unavailable, causing a computing task to be delayed, there may be serious consequences. If a program that prints pay checks is delayed, employees' confidence in the company may be shaken, or some employees may face penalties from not being able to pay their own bills. If customers cannot make transactions because the computer is down, they may choose to take their business to a competitor. For some time-critical services involving human lives, such as a hospital's life-support systems or a space station's guidance systems, the costs of failure are infinitely high. Thus, we must analyze the ramifications of a computer security failure. The following questions can prompt us to think about issues of explicit and hidden cost related to security. The answers may not produce precise cost figures, but they will help identify the sources of various types of costs.

- What are the legal obligations for preserving the confidentiality or integrity of a given data item?
- What business requirements and agreements cover the situation? Does the organization have to pay a penalty if it cannot provide a service?
- Could release of a data item cause harm to a person or organization? Would there be the possibility of legal action if harm were done?
- Could unauthorized access to a data item cause the loss of future business opportunity? Might it give a competitor an unfair advantage? What would be the estimated loss in revenue?
- What is the psychological effect of lack of computer service? Embarrassment? Loss of credibility? Loss of business? How many customers would be affected? What is their value as customers?

- What is the value of access to data or programs? Could this computation be deferred? Could this computation be performed elsewhere? How much would it cost to have a third party do the computing elsewhere?
- What is the value to someone else of having access to data or programs? How much would a competitor be willing to pay for access?
- What other problems would arise from loss of data? Could the data be replaced or reconstructed? With what amount of work?

These are not easy costs to evaluate. Nevertheless, they are needed to develop a thorough understanding of the risks. Furthermore, the vulnerabilities in computer security are often considerably higher than managers expect. Realistic estimates of potential harm can raise concern and suggest places in which attention to security is especially needed.

Step 5: Survey and Select New Controls:

By this point in our risk analysis, we understand the system's vulnerabilities and the likelihood of exploitation. We turn next to an analysis of the controls to see which ones address the risks we have identified. We want to match each vulnerability with at least one appropriate security technique. Once we do that, we can use our expected loss estimates to help us decide which controls, alone or in concert, are the most cost effective for a given situation.

Choosing Controls:

In this analysis controls can overlap, as for example, when a human guard and a locked door both protect against unauthorized access. Neither of these is redundant, because the human guard can handle exceptional situations (for example, when a legitimate user loses a key), but the lock prevents access if the guard is distracted. Also, one control may cover multiple vulnerabilities, so encrypting a set of data may protect both confidentiality and integrity. Controls have positive and negative effects: Encryption, for example, protects confidentiality, but it also takes time and introduces key management issues. Thus, when selecting controls, you must consider the full impact. Controls are not perfect. They can fail: Guards can be bribed or fall asleep, encryption can be broken, and access control devices can malfunction. Some controls are stronger than others. For example, a physical device is generally stronger than a written policy (policies are nevertheless useful).

Which Controls Are Best?

Typically, there is no single best set of controls. One control is stronger, another is more usable, another prevents harm instead of detecting it afterwards, and still another protects against several types of vulnerabilities. As you have inferred, risk analysis involves

building a multidimensional array: assets, vulnerabilities, likelihoods, controls. Mapping controls to vulnerabilities may involve using graph theory to select a minimal set of controls that address all vulnerabilities. The advantage of careful, systematic documentation of all these data is that each choice can be analyzed, and the side effects of changes are apparent. If this process sounds difficult, it is, but it need not be overwhelming. Listing all assets is less important than listing the top few, probably five to ten. Postulating all vulnerabilities is less important than recognizing several classes of harm and representative causes. With a manageable number of assets and vulnerabilities, determining controls (some of which may already be in place) need not be extensive, if some control covers each major vulnerability.

Step 6: Project Savings:

By this point in our risk analysis, we have identified controls that address each vulnerability in our list. The next step is to determine whether the costs outweigh the benefits of preventing or mitigating the risks. Recall that we multiply the risk probability by the risk impact to determine the risk exposure. The risk impact is the loss that we might experience if the risk were to turn into a real problem. There are techniques to help us determine the risk exposure. The effective cost of a given control is the actual cost of the control (such as purchase price, installation costs, and training costs) minus any expected loss from using the control (such as administrative or maintenance costs). Thus, the true cost of a control may be positive if the control is expensive to administer or introduces new risk in another area of the system. Or the cost can even be negative if the reduction in risk is greater than the cost of the control. For example, suppose a department has determined that some users have gained unauthorized access to the computing system. It is feared that the intruders might intercept or even modify sensitive data on the system. One approach to addressing this problem is to install a more secure data access control program. Even though the cost of the access control software is high, its cost is easily justified when compared to its value. Because the entire cost of the package is charged in the first year, even greater benefits are expected for subsequent years.

12.3.3 Arguments for and against Risk Analysis:

Risk analysis is a well-known planning tool, used often by auditors, accountants, and managers. In many situations, such as obtaining approval for new drugs, new power plants, and new medical devices, a risk analysis is required by law in many countries. There are many good reasons to perform a risk analysis in preparation for creating a security plan.

- **Improve awareness:** Discussing issues of security can raise the

general level of interest and concern among developers and users. Especially when the user population has little expertise in computing, the risk analysis can educate users about the role security plays in protecting functions and data that are essential to user operations and products.

- **Relate security mission to management objectives:** Security is often perceived as a financial drain for no gain. Management does not always see that security helps balance harm and control costs.
- **Identify assets, vulnerabilities, and controls:** Some organizations are unaware of their computing assets, their value to the organization, and the vulnerabilities associated with those assets. A systematic analysis produces a comprehensive list of assets, valuations, and risks.
- **Improve basis for decisions:** A security manager can present an argument such as "I think we need a firewall here" or "I think we should use token-based authentication instead of passwords." Risk analysis augments the manager's judgment as a basis for the decision.
- **Justify expenditures for security:** Some security mechanisms appear to be very expensive and without obvious benefit. A risk analysis can help identify instances where it is worth the expense to implement a major security mechanism. Justification is often derived from examining the much larger risks of *not* spending for security.

However, despite the advantages of risk analysis, there are several arguments against using it to support decision making.

- **False sense of precision and confidence:** The heart of risk analysis is the use of empirical data to generate estimates of risk impact, risk probability, and risk exposure. The danger is that these numbers will give us a false sense of precision, thereby giving rise to an undeserved confidence in the numbers. However, in many cases the numbers themselves are much less important than their relative sizes. Whether an expected loss is \$100,000 or \$150,000 is relatively unimportant. It is much more significant that the expected loss is far above the \$10,000 or \$20,000 budget allocated for implementing a control. Moreover, anytime a risk analysis generates a large potential loss, the system deserves further scrutiny to see if the root cause of the risk can be addressed.
- **Hard to perform:** Enumerating assets, vulnerabilities, and controls requires creative thinking. Assessing loss frequencies and impact can be difficult and subjective. A large risk analysis will have many things to consider. Risk analysis can be restricted to certain assets or vulnerabilities, however.

- **Immutability:** It is typical on many software projects to view processes like risk analysis as an irritating fact of life a step to be taken in a hurry so that the developers can get on with the more interesting jobs related to designing, building, and testing the system. For this reason, risk analyses, like contingency plans and five-year plans, tend to be filed and promptly forgotten. But if an organization takes security seriously, it will view the risk analysis as a living document, updating it at least annually or in conjunction with major system upgrades.
- **Lack of accuracy:** Risk analysis is not always accurate, for many reasons. First, we may not be able to calculate the risk probability with any accuracy, especially when we have no history of similar situations. Second, even if we know the likelihood, we cannot always estimate the risk impact very well. The risk management literature is replete with papers about describing the scenario, showing that presenting the same situation in two different ways to two equivalent groups of people can yield two radically different estimates of impact. And third, we may not be able to anticipate all the possible risks. For example, bridge builders did not know about the risks introduced by torque from high winds until the Tacoma Narrows Bridge twisted in the wind and collapsed. After studying the colossal failure of this bridge and discovering the cause, engineers made mandatory the inclusion of torque in their simulation parameters. Similarly, we may not know enough about software, security, or the context in which the system is to be used, so there may be gaps in our risk analysis that cause it to be inaccurate.

This lack of accuracy is often cited as a deficiency of risk analysis. But this lack is a red herring. Risk analysis is useful as a planning tool, to compare options. We may not be able to predict events accurately, but we can use risk analysis to weigh the tradeoffs between one action and another. When risk analysis is used in security planning, it highlights which security expenditures are likely to be most cost effective. This investigative basis is important for choosing among controls when money available for security is limited. And our risk analysis should improve as we build more systems, evaluate their security, and have a larger experience base from which to draw our estimates.

A risk analysis has many advantages as part of security plan or as a tool for less formal security decision making. It ranges from very subjective and imprecise to highly quantitative. It is useful for generating and documenting thoughts about likely threats and possible countermeasures. Finally, it supports rational decision making about security controls.

12.4 SUMMARY

- The administration of security draws on skills slightly different from the technical skills. The security administrator must understand not just security assets, threats, vulnerabilities, and controls, but management and implementation. In this chapter we examined how security is administered.
- First, security planning is a process that drives the rest of security administration. A security plan is a structure that allows things to happen in a studied, organized manner. General security plans explain how the organization will match threats to controls and to assets. Business continuity plans focus on the single issue of maintaining some ability to do business. Incident response plans cover how to keep a security event, such as a breach or attack, from running out of control. All plans offer the advantage that you can think about a situation in advance, with a clear mind, when you can weigh options easily.
- Risk assessment is a technique supporting security planning. In a risk assessment, you list vulnerabilities and controls, and then balance the cost of each control against the potential harm it can block. Risk assessments let you calculate the savings of security measures, instead of their costs, as is more frequently the case. Not all risk can be blocked. With a thorough risk assessment, you can know what risks you choose to accept.

12.5 REVIEW QUESTIONS

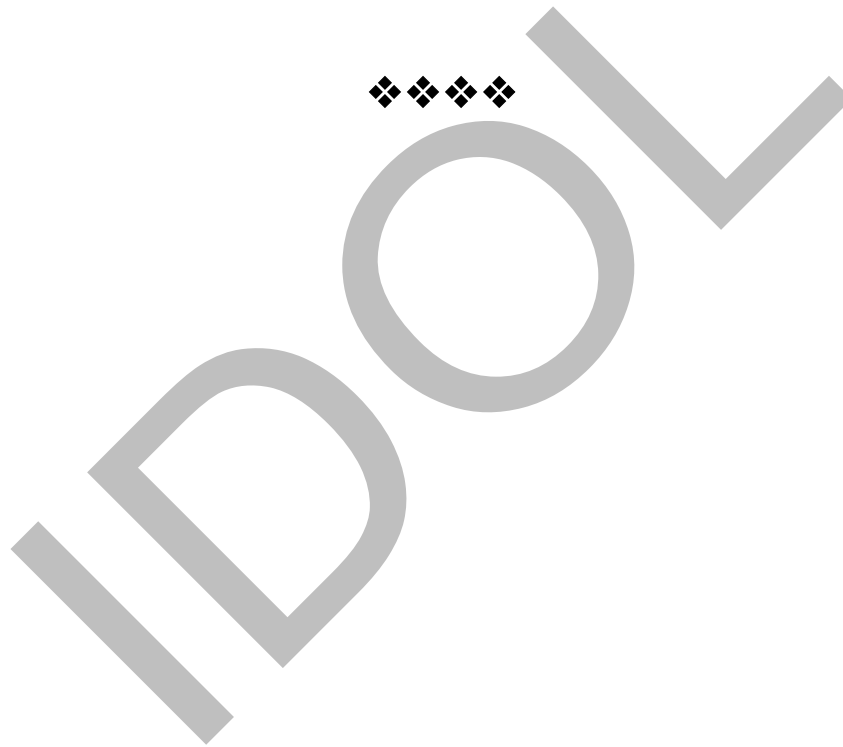
- a) Explain the contents of a security plan.
- b) Explain the six requirements of TCSEC.
- c) Explain the characteristics for the requirements of security plan.
- d) Write a short note on Business Continuity Plan.
- e) Write a short note on Incident Response Plans.
- f) What is Risk Analysis? Explain its nature.
- g) Explain the steps in risk analysis.
- h) Why should you perform risk analysis?
- i) What are the disadvantages of performing risk analysis?

12.6 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- *Security in Computing* by C. P. Pfleeger, and S. L. Pfleeger,
Unedited Version: Information Security

Pearson Education.

- *Computer Security: Art and Science* by Matt Bishop, Pearson Education.
- *Cryptography and Network Security: Principles and practice* by Stallings
- *Network Security* by Kaufman, Perlman, Speciner
- *Network Security : A Beginner's Guide* by Eric Maiwald, TMH
- *Java Network Security* by Macro Pistoia, Pearson Education
- *Principles of information security* by Whitman, Mattord, Thomson



SECURITY POLICIES AND PHYSICAL SECURITY

Unit Structure:

- 13.0 Objectives
- 13.1 Introduction
- 13.2 Organizational Security Policies
 - 13.2.1. Purpose
 - 13.2.2. Audience
 - 13.2.3. Contents
 - 13.2.4. Characteristics of a Good Security Plan
 - 13.2.5. Examples
- 13.3 Physical Security
 - 13.3.1. Natural Disasters
 - 13.3.2. Power Loss
 - 13.3.3. Surge Suppressor
 - 13.3.4. Human Vandals
 - 13.3.5. Interception of Sensitive Information
 - 13.3.6. Contingency Planning
- 13.4 Summary
- 13.5 Review Questions
- 13.6 Bibliography, References and Further Reading

13.0 OBJECTIVES

In this chapter, we understand how to establish a framework for security needs and understand the impact that computing environment has on the physical security.

13.1 INTRODUCTION

Security is a combination of technical, administrative, and physical controls, as we first pointed out in the earlier chapters. So far, we have considered technical controls almost exclusively. But stop and think for a moment: What good is a firewall if there is no power to run it? How effective is a public key infrastructure if someone can walk off with the certificate server? And why have

[Type here]

[Type here]

[Type here]

elaborate access control mechanisms if your employee mails a sensitive document to a competitor? The administrative and physical controls may be less glamorous than the technical ones, but they are surely as important. In this and the next chapter we complete our study of security controls by considering administrative and physical aspects. We look at four related areas:

- *Planning*: What advance preparation and study lets us know that our implementation meets our security needs for today and tomorrow?
- *Risk analysis*: How do we weigh the benefits of controls against their costs, and how do we justify any controls?
- *Policy*: How do we establish a framework to see that our computer security needs continue to be met?
- *Physical control*: What aspects of the computing environment have an impact on security?

These four areas are just as important to achieving security as are the latest firewall or coding practice.

13.2 ORGANIZATIONAL SECURITY POLICIES

A key element of any organization's security planning is an effective security policy. A security policy must answer three questions: *who* can access *which resources* in *what manner*? A security policy is a high-level management document to inform all users of the goals of and constraints on using a system. A policy document is written in broad enough terms that it does not change frequently. The information security policy is the foundation upon which all protection efforts are built. It should be a visible representation of priorities of the entire organization, definitively stating underlying assumptions that drive security activities. The policy should articulate senior management's decisions regarding security as well as asserting management's commitment to security. To be effective, the policy must be understood by everyone as the product of a directive from an authoritative and influential person at the top of the organization. People sometimes issue other documents, called procedures or guidelines, to define how the policy translates into specific actions and controls. In this section, we examine how to write a useful and effective security policy.

13.2.1 Purpose:

Security policies are used for several purposes, including the following:

- recognizing sensitive information assets
- clarifying security responsibilities
- promoting awareness for existing employees
- guiding new employees

13.2.2 Audience:

A security policy addresses several different audiences with different expectations. That is, each group – users, owners, and beneficiaries use the security policy in important but different ways.

Users:

Users legitimately expect a certain degree of confidentiality, integrity, and continuous availability in the computing resources provided to them. Although the degree varies with the situation, a security policy should reaffirm a commitment to this requirement for service. Users also need to know and appreciate what is considered acceptable use of their computers, data, and programs. For users, a security policy should define acceptable use.

Owners:

Each piece of computing equipment is owned by someone, and the owner may not be a system user. An owner provides the equipment to users for a purpose, such as to further education, support commerce, or enhance productivity. A security policy should also reflect the expectations and needs of owners.

Beneficiaries:

A business has paying customers or clients; they are beneficiaries of the products and services offered by that business. At the same time, the general public may benefit in several ways: as a source of employment or by provision of infrastructure. For example, the government has customers: the citizens of its country, and "guests" who have visas enabling entry for various purposes and times. A university's customers include its students and faculty; other beneficiaries include the immediate community (which can take advantage of lectures and concerts on campus) and often the world population (enriched by the results of research and service). To varying degrees, these beneficiaries depend, directly or indirectly, on the existence of or access to computers, their data and programs, and their computational power. For this set of beneficiaries, continuity and integrity of computing are very important. In addition, beneficiaries value confidentiality and correctness of the data involved. Thus, the interests of beneficiaries of a system must be reflected in the system's security policy.

Balance Among All Parties:

A security policy must relate to the needs of users, owners, and beneficiaries. Unfortunately, the needs of these groups may conflict. A beneficiary might require immediate access to data, but owners or users might not want to bear the expense or inconvenience of providing access at all hours. Continuous availability may be a goal for users, but that goal is inconsistent with a need to perform preventive or emergency maintenance. Thus, the

[Type here] [Type here] [Type here]
security policy must balance the priorities of all affected communities.

13.2.3 Contents:

A security policy must identify its audiences: the beneficiaries, users, and owners. The policy should describe the nature of each audience and their security goals. Several other sections are required, including the purpose of the computing system, the resources needing protection, and the nature of the protection to be supplied. We discuss each one in turn.

Purpose:

The policy should state the purpose of the organization's security functions, reflecting the requirements of beneficiaries, users, and owners. For example, the policy may state that the system will "protect customers' confidentiality or preserve a trust relationship," "ensure continual usability," or "maintain profitability." There are typically three to five goals, such as:

- Promote efficient business operation.
- Facilitate sharing of information throughout the organization.
- Safeguard business and personal information.
- Ensure that accurate information is available to support business processes.
- Ensure a safe and productive place to work.
- Comply with applicable laws and regulations.

The security goals should be related to the overall goal or nature of the organization. It is important that the system's purpose be stated clearly and completely because subsequent sections of the policy will relate back to these goals, making the policy a goal-driven product.

Protected Resources:

A risk analysis will have identified the assets that are to be protected. These assets should be listed in the policy, in the sense that the policy lays out which items it addresses. For example, will the policy apply to all computers or only to those on the network? Will it apply to all data or only to client or management data? Will security be provided to all programs or only the ones that interact with customers? If the degree of protection varies from one service, product, or data type to another, the policy should state the differences. For example, data that uniquely identify clients may be protected more carefully than the names of cities in which clients reside.

Nature of the Protection:

The asset list tells us *what* should be protected. The policy should also indicate *who* should have access to the protected items. It may also indicate *how* that access will be ensured and *how* unauthorized people will be denied access. All the mechanisms described in this book are at your disposal in deciding which controls should protect which objects. In particular, the security policy should state what degree of protection should be provided to which kinds of resources.

13.2.4 Characteristics of a Good Security Plan:

If a security policy is written poorly, it cannot guide the developers and users in providing appropriate security mechanisms to protect important assets. Certain characteristics make a security policy a good one.

Coverage:

A security policy must be comprehensive: It must either apply to or explicitly exclude all possible situations. Furthermore, a security policy may not be updated as each new situation arises, so it must be general enough to apply naturally to new cases that occur as the system is used in unusual or unexpected ways.

Durability:

A security policy must grow and adapt well. In large measure, it will survive the system's growth and expansion without change. If written in a flexible way, the existing policy will be applicable to new situations. However, there are times when the policy must change (such as when government regulations mandate new security constraints), so the policy must be changeable when it needs to be. An important key to durability is keeping the policy free from ties to specific data or protection mechanisms that almost certainly will change. It is preferable to describe assets needing protection in terms of their function and characteristics, rather than in terms of specific implementation. Better still, we can separate the elements of the policy, having one policy statement for student grades and another for customers' proprietary data. Similarly, we may want to define one policy that applies to preserving the confidentiality of relationships, and another protecting the use of the system through strong authentication.

Realism:

The policy must be realistic. That is, it must be possible to implement the stated security requirements with existing technology. Moreover, the implementation must be beneficial in terms of time, cost, and convenience; the policy should not recommend a control that works but prevents the system or its users from performing their

[Type here] [Type here] [Type here]
activities and functions. It is important to make economically worthwhile investments in security, just as for any other careful business investment.

Usefulness:

An obscure or incomplete security policy will not be implemented properly, if at all. The policy must be written in language that can be read, understood, and followed by anyone who must implement it or is affected by it. For this reason, the policy should be succinct, clear, and direct.

13.2.5 Examples:

To understand the nature of security policies, we study a few examples to illustrate some of the points just presented.

Data Sensitivity Policy:

Our first example is from an organization that decided to classify all its data resources into four levels, based on how severe the effect might be if a resource were damaged. These levels are sensitive, personal or protected, company confidential and open. Then, the required protection was based on the resource's level. Finally, the organization analyzed its threats, their possible severities, and countermeasures, and their effectiveness, within each of the four levels. Although the phrases describing the degree of damage are open to interpretation, the intent of these levels is clear: All information assets are to be classified as sensitive, personal, confidential, or open, and protection requirements for these four types are detailed in the remainder of the organization's policy document.

Government Agency IT Security Policy:

The U.S. Department of Energy (DOE), like many government units, has established its own security policy. The following excerpt is from the policy on protecting classified material, although the form is appropriate for many unclassified uses as well. It is the policy of DOE that classified information and classified ADP [automatic data processing] systems shall be protected from unauthorized access (including the enforcement of need-to-know protections), alteration, disclosure, destruction, penetration, denial of service, subversion of security measures, or improper use as a result of espionage, criminal, fraudulent, negligent, abusive, or other improper actions. The DOE shall use all reasonable measures to protect ADP systems that process, store, transfer, or provide access to classified information, to include but not limited to the following: physical security, personnel security, telecommunications security, administrative security, and hardware and software security

[Type here] [Type here] [Type here]
measures. This order establishes this policy and defines responsibilities for the development, implementation, and periodic evaluation of the DOE program. The policy then continues for several more pages to list specific responsibilities for specific people. The cited paragraph is comprehensive, covering practically every possible source (espionage, crime, fraud, etc.) of practically every possible harm (unauthorized access, alteration, destruction, etc.), and practically every possible kind of control (physical, personnel, etc.).

The generality of the header paragraph is complemented by subsequent paragraphs giving specific responsibilities:

- "Each data owner shall determine and declare the required protection level of information . . ."
- "Each security officer shall . . . perform a risk assessment to identify and document specific . . . assets, . . . threats, . . . and vulnerability . . ."
- "Each manager shall...establish procedures to ensure that systems are continuously monitored...to detect security infractions . . ." and so on.

Internet Security Policy:

The Internet does not have a governing security policy per se, because it is a federation of users. Nevertheless, the Internet Society drafted a security policy for its members. The policy contains the following interesting portions.

- Users are individually responsible for understanding and respecting the security policies of the systems (computers and networks) they are using. Users are individually accountable for their own behaviour.
- Users have a responsibility to employ available security mechanisms and procedures for protecting their own data. They also have a responsibility for assisting in the protection of the systems they use.
- Computer and network service providers are responsible for maintaining the security of the systems they operate. They are further responsible for notifying users of their security policies and any changes to these policies.
- Vendors and system developers are responsible for providing systems which are sound, and which embody adequate security controls.
- Users, service providers, and hardware and software vendors are responsible for cooperating to provide security.
- Technical improvements in Internet security protocols should be sought on a continuing basis. At the same time, personnel

[Type here] [Type here] [Type here]
developing new protocols, hardware or software for the Internet
are expected to include security considerations as part of the
design and development process.

These statements clearly state to whom they apply and for what each
party is responsible.

13.3 PHYSICAL SECURITY

Much of this book has focused on technical issues in security and their technical solutions: firewalls, encryption techniques, and more. But many threats to security involve human or natural disasters, events that should also be addressed in the security plan. For this reason, in this section we consider how to cope with the nontechnical things that can go wrong. There are two pieces to the process of dealing with nontechnical problems: preventing things that can be prevented and recovering from the things that cannot be prevented. Physical security is the term used to describe protection needed outside the computer system. Typical physical security controls include guards, locks, and fences to deter direct attacks. In addition, there are other kinds of protection against less direct disasters, such as floods and power outages; these, too, are part of physical security. As we will see, many physical security measures can be provided simply by good common sense, a characteristic that Mark Twain noted "is a most uncommon virtue."

13.3.1 Natural Disasters:

Computers are subject to the same natural disasters that can occur to homes, stores, and automobiles. They can be flooded, burned, melted, hit by falling objects, and destroyed by earthquakes, storms, and tornadoes. Additionally, computers are sensitive to their operating environment, so excessive heat or inadequate power is also a threat. It is impossible to prevent natural disasters, but through careful planning it is possible to reduce the damage they inflict. Some measures can be taken to reduce their impact. Because many of these perils cannot be prevented or predicted, controls focus on limiting possible damage and recovering quickly from a disaster. Issues to be considered include the need for offsite backups, the cost of replacing equipment, the speed with which equipment can be replaced, the need for available computing power, and the cost or difficulty of replacing data and programs.

Flood:

Water from a natural flood comes from ground level, rising gradually, and bringing with it mud and debris. Often, there is time for an orderly shutdown of the computing system; at worst, the organization loses some of the processing in progress. At other times, such as when a dam breaks, a water pipe bursts, or the roof collapses in a storm, a sudden flood can overwhelm the system and

[Type here]

[Type here]

[Type here]

its users before anything can be saved. Water can come from above, below, or the side. The machinery may be destroyed or damaged by mud and water, but most computing systems are insured and replaceable by the manufacturer. Managers of unique or irreplaceable equipment who recognize the added risk sometimes purchase or lease duplicate redundant hardware systems to ensure against disruption of service. Even when the hardware can be replaced, we must be concerned about the stored data and programs. The system administrator may choose to label storage media in a way that makes it easy to identify the most important data. For example, green, yellow, and red labels may show which disks are the most sensitive, so that all red disks are moved from the data centre during a storm. Similarly, large plastic bags and waterproof tape can be kept near important equipment and media; they are used to protect the hardware and storage media in case of a burst pipe or other sudden flood. The real issue is protecting data and preserving the ability to compute. The only way to ensure the safety of data is to store backup copies in one or more safe locations.

Fire:

Fire is more serious than water; often there is not as much time to react, and human lives are more likely to be in immediate danger. To ensure that system personnel can react quickly, every user and manager should have a plan for shutting down the system in an orderly manner. Such a process takes only a few minutes but can make recovery much easier. This plan should include individual responsibilities for all people: some to halt the system, others to protect crucial media, others to close doors on media cabinets. Provision should be made for secondary responsibilities, so that onsite staff can perform duties for those who are not in the office. Water is traditionally used to put out fires, but it is not a good idea for use in computer rooms. In fact, more destruction can be the result of sprinklers than of the fires themselves. A fire sensor usually activates many sprinklers, dousing an entire room, even when the fire is merely some ignited paper in a wastebasket and of no threat to the computing system. Many computing centres use carbon dioxide extinguishers or an automatic system that sprays a gas such as Halon to smother a fire but leave no residue. Unfortunately, these gas systems work by displacing the oxygen in the room, choking the fire but leaving humans unable to breathe. Consequently, when these protection devices are activated, humans must leave, disabling efforts to protect media. The best defines for situations like these is careful placement of the computing facility. A windowless location with fire-resistant access doors and non-flammable full-height walls can prevent some fires from spreading from adjacent areas to the computing room. With a fire and smoke-resistant facility, personnel merely shut down the system and leave, perhaps carrying out the most important media. Fire prevention is quite effective, especially because most computer goods are not especially flammable. Advance planning, reinforced with simulation drills, can help make

[Type here]

[Type here]

[Type here]

good use of the small amount of time available before evacuation is necessary.

Other Natural Disasters:

Computers are subject to storms, earthquakes, volcanoes, and similar events. Although not natural disasters, building collapse, explosion, and damage from falling objects can be considered in the same category. These kinds of catastrophes are difficult to predict or estimate. But we know these catastrophes will occur. Security managers cope with them in several ways:

- developing contingency plans so that people know how to react in emergencies and business can continue
- ensuring physical assets, computers, buildings, devices, supplies against harm
- preserving sensitive data by maintaining copies in physically separated locations

13.3.2 Power Loss:

Computers need their food, electricity and they require a constant, pure supply of it. With a direct power loss, all computation ceases immediately. Because of possible damage to media by sudden loss of power, many disk drives monitor the power level and quickly retract the recording head if power fails. For certain time-critical applications, loss of service from the system is intolerable; in these cases, alternative complete power supplies must be instantly available.

Uninterruptible Power Supply:

One protection against power loss is an uninterruptible power supply. This device stores energy during normal operation so that it can return the backup energy if power fails. One form of uninterruptible power supply uses batteries that are continually charged when the power is on, but which then provide power when electricity fails. However, size, heat, flammability, and low output can be problems with batteries. Some uninterruptible power supplies use massive wheels that are kept in continuous motion when electricity is available. When the power fails, the inertia in the wheels operates generators to produce more power. Size and limited duration of energy output are problems with this variety of power supply. Both forms of power supplies are intended to provide power for a limited time, just long enough to allow the current state of the computation to be saved so that no computation is lost.

13.3.3 Surge Suppressor:

Another problem with power is its "cleanness." Although most people are unaware of it, a variation of 10 percent from the stated voltage of a line is considered acceptable, and some power lines vary

[Type here]

[Type here]

[Type here]

even more. A power line may always be 10 percent high or low. In many places, lights dim momentarily when a large appliance, such as an air conditioner, begins operation. When a large motor starts, it draws an exceptionally large amount of current, which reduces the flow to other devices on the line. When a motor stops, the sudden termination of draw can send a temporary surge along the line. Similarly, lightning strikes may send a momentary large pulse. Thus, instead of being constant, the power delivered along any electric line shows many brief fluctuations, called drops, spikes, and surges. A drop is a momentary reduction in voltage, and a spike or surge is a rise. For computing equipment, a drop is less serious than a surge. Most electrical equipment is tolerant of rather large fluctuations of current. These variations can be destructive to sensitive electronic equipment, however. Simple devices called "surge suppressors" filter spikes from an electric line, blocking fluctuations that would affect computers. These devices cost from \$20 to \$100; they should be installed on every computer, printer, or other connected component. More sensitive models are typically used on larger systems. As mentioned previously, a lightning strike can send a surge through a power line. To increase protection, personal computer users usually unplug their machines when they are not in use, as well as during electrical storms. Another possible source of destruction is lightning striking a telephone line. Because the power surge can travel along the phone line and into the computer or peripherals, the phone line should be disconnected from the modem during storms. These simple measures may save much work as well as valuable equipment.

13.3.4 Human Vandals:

Because computers and their media are sensitive to a variety of disruptions, a vandal can destroy hardware, software, and data. Human attackers may be disgruntled employees, bored operators, saboteurs, people seeking excitement, or unwitting bumbler. If physical access is easy to obtain, crude attacks using axes or bricks can be very effective. Physical attacks by unskilled vandals are often easy to prevent; a guard can stop someone approaching a computer installation with a threatening or dangerous object. When physical access is difficult, more subtle attacks can be tried, resulting in quite serious damage. People with only some sophisticated knowledge of a system can short-circuit a computer with a car key or disable a disk drive with a paper clip. These items are not likely to attract attention until the attack is completed.

Unauthorized Access and Use:

Films and newspaper reports exaggerate the ease of gaining access to a computing system. Still, as distributed computing systems become more prevalent, protecting the system from outside access becomes more difficult and more important. Interception is a form of unauthorized access; the attacker intercepts data and either

[Type here]

[Type here]

[Type here]

breaks confidentiality or prevents the data from being read or used by others. In this context, interception is a passive attack. But we must also be concerned about active interception, in the sense that the attacker can change or insert data before allowing it to continue to its destination.

Theft:

It is hard to steal a large mainframe computer. Not only is carrying it away difficult but finding a willing buyer and arranging installation and maintenance also require special assistance. However, printed reports, tapes, or disks can be carried easily. If done well, the loss may not be detected for some time. Personal computers, laptops, and personal digital assistants (PDAs, such as Palms or Blackberries) are designed to be small and portable. Diskettes and tape backup cartridges are easily carried in a shirt pocket or briefcase. Computers and media that are easy to carry are also easy to conceal. We can take one of three approaches to preventing theft: preventing access, preventing portability, or detecting exit.

Preventing Access:

The surest way to prevent theft is to keep the thief away from the equipment. However, thieves can be either insiders or outsiders. Therefore, access control devices are needed both to prevent access by unauthorized individuals and to record access by those authorized. A record of accesses can help identify who committed a theft. The oldest access control is a guard, not in the database management system sense but rather in the sense of a human being stationed at the door to control access to a room or to equipment. Guards offer traditional protection; their role is well understood, and the protection they offer is adequate in many situations. However, guards must be on duty continuously to be effective; providing breaks implies at least four guards for a 24-hour operation, with extras for vacation and illness. A guard must personally recognize someone or recognize an access token, such as a badge. People can lose or forget badges; terminated employees and forged badges are also problems. Unless the guard makes a record of everyone who has entered a facility, there is no way to know who (employee or visitor) has had access in case a problem is discovered.

The second oldest access control is a lock. This device is even easier, cheaper, and simpler to manage than a guard. However, it too provides no record of who has had access, and difficulties arise when keys are lost or duplicated. At computer facilities, it is inconvenient to fumble for a key when your hands are filled with tapes or disks, which might be ruined if dropped. There is also the possibility of piggybacking: a person walks through the door that someone else has just unlocked. Still, guards and locks provide simple, effective security for access to facilities such as computer rooms.

[Type here]

[Type here]

[Type here]

More exotic access control devices employ cards with radio transmitters, magnetic stripe cards (like 24-hour bank cards), and smart cards with chips containing electronic circuitry that makes them difficult to duplicate. Because each of these devices' interfaces with a computer, it is easy for the computer to capture identity information, generating a list of who entered and left the facility, when, and by which routes. Some of these devices operate by proximity, so that a person can carry the device in a pocket or clipped to a collar; the person obtains easy access even when hands are full. Because these devices are computer controlled, it is easy to invalidate an access authority when someone quits or reports the access token lost or stolen. The nature of the application or service determines how strict the access control needs to be. Working in concert with computer-based authentication techniques, the access controls can be part of defines in depth using multiple mechanisms to provide security.

Preventing Portability:

Portability is a mixed blessing. We can now carry around in our pockets devices that provide as much computing power as mainframes did twenty years ago. Portability is in fact a necessity in devices such as PDAs and mobile phones. And we do not want to permanently affix our personal computers to our desks, in case they need to be removed for repair or replacement. Thus, we need to find ways to enable portability without promoting theft. One antitheft device is a pad connected to cable, like those used to secure bicycles. The pad is glued to the desktop with extremely strong adhesive. The cables loop around the equipment and are locked in place. Releasing the lock permits the equipment to be moved. An alternative is to couple the base of the equipment to a secure pad, in much the same way that televisions are locked in place in hotel rooms. Yet a third possibility is a large, lockable cabinet in which the personal computer and its peripherals are kept when they are not in use. Some people argue that cables, pads, and cabinets are unsightly and, worse, they make the equipment inconvenient to use. Another alternative is to use movement-activated alarm devices when the equipment is not in use. Small alarms are available that can be locked to a laptop or PDA. When movement is detected, a loud, annoying whine or whistle warns that the equipment has been disturbed. Such an alarm is especially useful when laptops must be left in meeting or presentation rooms overnight or during a break. Used in concert with guards, the alarms can offer reasonable protection at reasonable cost.

Detecting Theft:

For some devices, protection is more important than detection. We want to keep someone from stealing certain systems or information at all costs. But for other devices, it may be enough to detect that an attempt has been made to access or steal hardware or software. For example, chaining down a disk makes it unusable.

[Type here]

[Type here]

[Type here]

Instead, we try to detect when someone tries to leave a protected area with the disk or other protected object. In these cases, the protection mechanism should be small and unobtrusive. One such mechanism is like the protection used by many libraries, bookstores, or department stores. Each sensitive object is marked with a special label. Although the label looks like a normal pressure-sensitive one, its presence can be detected by a machine at the exit door if the label has not been disabled by an authorized party, such as a librarian or salesclerk. Similar security code tags are available for vehicles, people, machinery, and documents. Some tags are enabled by radio transmitters. When the detector sounds an alarm, someone must apprehend the person trying to leave with the marked object.

13.3.5 Interception of Sensitive Information:

When disposing of a draft copy of a confidential report containing its sales strategies for the next five years, a company wants to be especially sure that the report is not reconstructable by one of its competitors. When the report exists only as hard copy, destroying the report is straightforward, usually accomplished by shredding or burning. But when the report exists digitally, destruction is more problematic. There may be many copies of the report in digital and paper form and in many locations (including on the computer and on storage media). There may also be copies in backups and archived in e-mail files. In this section, we look at several ways to dispose of sensitive information.

Shredding:

Shredders have existed for a long time, as devices used by banks, government agencies, and other organizations to dispose of large amounts of confidential data. Although most of the shredded data is on paper, shredders can also be used for destroying printer ribbons and some types of disks and tapes. Shredders work by converting their input to thin strips or pulp, with enough volume to make it infeasible for most people to try to reconstruct the original from its many pieces. When data are extremely sensitive, some organizations burn the shredded output for added protection.

Overwriting Magnetic Data:

Magnetic media present a special problem for those trying to protect the contents. When data are stored on magnetic disks, the ERASE or DELETE functions often simply change a directory pointer to free up space on the disk. As a result, the sensitive data are still recorded on the medium, and they can be recovered by analysis of the directory. A more secure way to destroy data on magnetic devices is to overwrite the data several times, using a different pattern each time. This process removes enough magnetic residue to prevent most people from reconstructing the original file. However, "cleaning" a disk in this fashion takes time. Moreover, a person using highly specialized equipment might be able to identify each separate

[Type here]

[Type here]

[Type here]

message, much like the process of peeling off layers of wallpaper to reveal the wall beneath.

Degaussing:

Degaussers destroy magnetic fields. Passing a disk or other magnetic medium through a degausser generates a magnetic flux so forceful that all magnetic charges are instantly realigned, thereby fusing all the separate layers. A degausser is a fast way to cleanse a magnetic medium, although there is still question as to whether it is adequate for use in the most sensitive of applications. For most users, a degausser is a fast way to neutralize a disk or tape, permitting it to be reused by others.

Protecting Against Emanation: Tempest:

Computer screens emit signals that can be detected from a distance. In fact, any components, including printers, disk drives, and processors, can emit information. Tempest is a U.S. government program under which computer equipment is certified as emission-free (that is, no detectable emissions). There are two approaches for preparing a device for Tempest certification: enclosing the device and modifying the emanations. The obvious solution to preventing emanations is to trap the signals before they can be picked up. Enclosing a device in a conductive case, such as copper, diffuses all the waves by conducting them throughout the case. Copper is a good conductor, and the waves travel much better through copper than through the air outside the case, so the emissions are rendered harmless. This solution works very well with cable, which is then enclosed in a solid, emanation-proof shield. Typically, the shielded cable is left exposed so that it is easy to inspect visually for any signs of tapping or other tampering. The shielding must be complete. That is, it does little good to shield a length of cable but not also shield the junction box at which that cable is connected to a component. The line to the component and the component itself must be shielded, too. The shield must enclose the device completely. If top, bottom, and three sides are shielded, emanations are prevented only in those directions. However, a solid copper shield is useless in front of a computer screen. Covering the screen with a fine copper mesh in an intricate pattern carries the emanation safely away. This approach solves the emanation problem while still maintaining the screen's usability.

Entire computer rooms or even whole buildings can be shielded in copper so that large computers inside do not leak sensitive emanations. Although it seems appealing to shield the room or building instead of each component, the scheme has significant drawbacks.

A shielded room is inconvenient because it is impossible to expand the room easily as needs change. The shielding must be done carefully, because any puncture is a possible point of emanation. Furthermore, continuous metal pathways, such as water

[Type here]

[Type here]

[Type here]

pipes or heating ducts, act as antennas to convey the emanations away from their source. Emanations can also be designed in such a way that they cannot be retrieved. This process is like generating noise to jam or block a radio signal. With this approach, the emanations of a piece of equipment must be modified by addition of spurious signals. Additional processors are added to Tempest equipment specifically to generate signals that fool an interceptor. The exact Tempest modification methods are classified.

As might be expected, Tempest-enclosed components are larger and heavier than their unprotected counterparts. Tempest testing is a rigorous program of the U.S. Department of Defence. Once a product has been approved, even a minor design modification, such as changing from one manufacturer's power supply to an equivalent one from another manufacturer, invalidates the Tempest approval. Therefore, these components are costly, ranging in price from 10 percent to 300 percent more than similar non-Tempest products. They are most appropriate in situations in which the data to be confined are of great value, such as top-level government information. Other groups with less dramatic needs can use other less rigorous shielding.

13.3.6 Contingency Planning:

The key to successful recovery is adequate preparation. Seldom does a crisis destroy irreplaceable equipment; most computing systems, personal computers to mainframes are standard, off-the-shelf systems that can be easily replaced. Data and locally developed programs are more vulnerable because they cannot be quickly substituted from another source. Let us look more closely at what to do after a crisis occurs.

Backup:

In many computing systems, some data items change frequently, whereas others seldom change. For example, a database of bank account balances changes daily, but a file of depositors' names and addresses changes much less often. Also, the number of changes in each period is different for these two files. These variations in number and extent of change relate to the amount of data necessary to reconstruct these files in the event of a loss. A backup is a copy of all or a part of a file to assist in re-establishing a lost file. In professional computing systems, periodic backups are usually performed automatically, often at night when system usage is low. Everything on the system is copied, including system files, user files, scratch files, and directories, so that the system can be regenerated after a crisis. This type of backup is called a complete backup. Complete backups are done at regular intervals, usually weekly or daily, depending on the criticality of the information or service provided by the system.

[Type here]

[Type here]

[Type here]

Major installations may perform revolving backups, in which the last several backups are kept. Each time a backup is done, the oldest backup is replaced with the newest one. There are two reasons to perform revolving backups: to avoid problems with corrupted media and to allow users or developers to retrieve old versions of a file. Another form of backup is a selective backup, in which only files that have been changed (or created) since the last backup are saved. In this case, fewer files must be saved, so the backup can be done more quickly. A selective backup combined with an earlier complete backup gives the effect of a complete backup in the time needed for only a selective backup. For each type of backup, we need the means to move from the backup forward to the point of failure. That is, we need a way to restore the system in the event of failure. In critical transaction systems, we address this need by keeping a complete record of changes since the last backup.

Sometimes, the system state is captured by a combination of computer- and paper-based recording media. For example, if a system handles bank teller operations, the individual tellers duplicate their processing on paper records the deposit and withdrawal slips that accompany your bank transactions; if the system fails, the staff restores the latest backup version and reapplies all changes from the collected paper copies. Or the banking system creates a paper journal, which is a log of transactions printed just as each transaction completes.

Personal computer users often do not appreciate the need for regular backups. Even minor crises, such as a failed piece of hardware, can seriously affect personal computer users. With a backup, users can simply change to a similar machine and continue work.

Offsite Backup:

A backup copy is useless if it is destroyed in the crisis, too. Many major computing installations rent warehouse space some distance from the computing system, far enough away that a crisis is not likely to affect the offsite location at the same time. As a backup is completed, it is transported to the backup site. Keeping a backup version separate from the actual system reduces the risk of its loss. Similarly, the paper trail is also stored somewhere other than at the main computing facility. Personal computer users concerned with integrity can take home a copy of important disks as protection or send a copy to a friend in another city. If both secrecy and integrity are important, a bank vault, or even a secure storage place in another part of the same building can be used. The worst place to store a backup copy is where it usually is stored: right next to the machine.

Networked Storage:

With today's extensive use of networking, using the network to implement backups is a good idea. Storage providers sell space in which you can store data; think of these services as big network-attached disk drives. You rent space just as you would consume electricity: You pay for what you use. The storage provider needs to provide only enough total space to cover everyone's needs, and it is easy to monitor usage patterns and increase capacity as combined needs rise. Networked storage is perfect for backups of critical data because you can choose a storage provider whose physical storage is not close to your processing. In this way, physical harm to your system will not affect your backup. You do not need to manage tapes or other media and physically transport them offsite.

Cold Site:

Depending on the nature of the computation, it may be important to be able to recover from a crisis and resume computation quickly. A bank, for example, might be able to tolerate a four-hour loss of computing facilities during a fire, but it could not tolerate a ten-month period to rebuild a destroyed facility, acquire new equipment, and resume operation. Most computer manufacturers have several spare machines of most models that can be delivered to any location within 24 hours in the event of a real crisis. Sometimes the machine will come straight from assembly; other times the system will have been in use at a local office. Machinery is seldom the hard part of the problem. Rather, the hard part is deciding where to put the equipment in order to begin a temporary operation. A cold site or shell is a facility with power and cooling available, in which a computing system can be installed to begin immediate operation. Some companies maintain their own cold sites, and other cold sites can be leased from disaster recovery companies. These sites usually come with cabling, fire prevention equipment, separate office space, telephone access, and other features. Typically, a computing center can have equipment installed and resume operation from a cold site within a week of a disaster.

Hot Site:

If the application is more critical or if the equipment needs are more specialized, a hot site may be more appropriate. A hot site is a computer facility with an installed and ready-to run computing system. The system has peripherals, telecommunications lines, power supply, and even personnel ready to operate on short notice. Some companies maintain their own; other companies subscribe to a service that has available one or more locations with installed and running computers. To activate a hot site, it is necessary only to load software and data from offsite backup copies.

Numerous services offer hot sites equipped with every

[Type here] [Type here] [Type here]
popular brand and model of system. They provide diagnostic and system technicians, connected communications lines, and an operations staff. The hot site staff also assists with relocation by arranging transportation and housing, obtaining needed blank forms, and acquiring office space. Because these hot sites serve as backups for many customers, most of whom will not need the service, the annual cost to any one customer is low. The cost structure is like insurance: The likelihood of an auto accident is low, so the premium is reasonable, even for a policy that covers the complete replacement cost of an expensive car. Notice, however, that the first step in being able to use a service of this type is a complete and timely backup.

13.4 SUMMARY

- The administration of security draws on skills slightly different from the technical skills. The security administrator must understand not just security assets, threats, vulnerabilities, and controls, but management and implementation. In this chapter we examined how security is administered.
- An organizational security policy is a document that specifies the organization's goals regarding security. It lists policy elements that are statements of actions that must or must not be taken to preserve those goals. Policy documents often lead to implementational procedures. Also, user education and awareness activities ensure that users are aware of policy restrictions.
- Physical security concerns the physical aspects of computing: the devices themselves and harm that can come to them because of the buildings in which they are contained. Physical security addresses two branches of threats: natural threats to buildings and the infrastructure, and human threats. Redundancy and physical controls address physical security threats.
- The administration of security has a strong human component, from the writing of plans and policies, to the mental work in performing a risk analysis, to the human guards that implement or reinforce many physical controls.
- By no means have we covered all physical security in this brief introduction. Professionals become experts at individual aspects, such as fire control or power provision. We must protect the facility against many sorts of disasters, from weather to chemical spills and vehicle crashes to explosions. It is impossible to predict what will occur or when.
- The physical security manager must consider all assets and a wide range of harm. Malicious humans seeking physical access are a different category of threat agent. With them, you can

[Type here]

[Type here]

[Type here]

consider motive or objective: is it theft of equipment, disruption of processing, interception of data, or access to service? Fences, guards, solid walls, and locks will deter or prevent most human attacks. But you always need to ask where weaknesses remain; a solid wall has a weakness in every door and window.

- The primary physical controls are strength and duplication. Strength means overlapping controls implementing a defines-in-depth approach so that if one control fails, the next one will protect. People who built ancient castles practiced this philosophy with moats, walls, drawbridges, and arrow slits. Duplication means eliminating single points of failure. Redundant copies of data protect against harm to one copy from any cause. Spare hardware components protect against failures.

13.5 REVIEW QUESTIONS

- a) What are the various audiences and how is balance managed among all the audiences?
- b) Explain the contents of a good security plan.
- c) What are the characteristics of a good security plan?
- d) What natural disasters are computers prone to and how can they be saved?
- e) What are the ways in which human vandals can cause problems in physical security?
- f) How can sensitive information be intercepted?
- g) Write a short note on contingency planning.

13.6 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- *Security in Computing* by C. P. Pfleeger, and S. L. Pfleeger, Pearson Education.
- *Computer Security: Art and Science* by Matt Bishop, Pearson Education.
- *Cryptography and Network Security: Principles and practice* by Stallings
- *Network Security* by Kaufman, Perlman, Speciner
- *Network Security : A Beginner's Guide* by Eric Maiwald, TMH
- *Java Network Security* by Macro Pistoia, Pearson Education

[Type here]

[Type here]

[Type here]

- *Principles of information security* by Whitman, Mattord, Thomson



100%

SECURITY POLICIES AND PHYSICAL SECURITY

Unit Structure:

- 13.0 Objectives
- 13.1 Introduction
- 13.2 Organizational Security Policies
 - 13.2.1. Purpose
 - 13.2.2. Audience
 - 13.2.3. Contents
 - 13.2.4. Characteristics of a Good Security Plan
 - 13.2.5. Examples
- 13.3 Physical Security
 - 13.3.1. Natural Disasters
 - 13.3.2. Power Loss
 - 13.3.3. Surge Suppressor
 - 13.3.4. Human Vandals
 - 13.3.5. Interception of Sensitive Information
 - 13.3.6. Contingency Planning
- 13.4 Summary
- 13.5 Review Questions
- 13.6 Bibliography, References and Further Reading

13.0 OBJECTIVES

In this chapter, we understand how to establish a framework for security needs and understand the impact that computing environment has on the physical security.

13.1 INTRODUCTION

Security is a combination of technical, administrative, and physical controls, as we first pointed out in the earlier chapters. So far, we have considered technical controls almost exclusively. But stop and think for a moment: What good is a firewall if there is no power to run it? How effective is a public key infrastructure if someone can walk off with the certificate server? And why have

[Type here]

[Type here]

[Type here]

elaborate access control mechanisms if your employee mails a sensitive document to a competitor? The administrative and physical controls may be less glamorous than the technical ones, but they are surely as important. In this and the next chapter we complete our study of security controls by considering administrative and physical aspects. We look at four related areas:

- *Planning*: What advance preparation and study lets us know that our implementation meets our security needs for today and tomorrow?
- *Risk analysis*: How do we weigh the benefits of controls against their costs, and how do we justify any controls?
- *Policy*: How do we establish a framework to see that our computer security needs continue to be met?
- *Physical control*: What aspects of the computing environment have an impact on security?

These four areas are just as important to achieving security as are the latest firewall or coding practice.

13.2 ORGANIZATIONAL SECURITY POLICIES

A key element of any organization's security planning is an effective security policy. A security policy must answer three questions: *who* can access *which resources* in *what manner*? A security policy is a high-level management document to inform all users of the goals of and constraints on using a system. A policy document is written in broad enough terms that it does not change frequently. The information security policy is the foundation upon which all protection efforts are built. It should be a visible representation of priorities of the entire organization, definitively stating underlying assumptions that drive security activities. The policy should articulate senior management's decisions regarding security as well as asserting management's commitment to security. To be effective, the policy must be understood by everyone as the product of a directive from an authoritative and influential person at the top of the organization. People sometimes issue other documents, called procedures or guidelines, to define how the policy translates into specific actions and controls. In this section, we examine how to write a useful and effective security policy.

13.2.1 Purpose:

Security policies are used for several purposes, including the following:

- recognizing sensitive information assets
- clarifying security responsibilities
- promoting awareness for existing employees
- guiding new employees

13.2.2 Audience:

A security policy addresses several different audiences with different expectations. That is, each group – users, owners, and beneficiaries use the security policy in important but different ways.

Users:

Users legitimately expect a certain degree of confidentiality, integrity, and continuous availability in the computing resources provided to them. Although the degree varies with the situation, a security policy should reaffirm a commitment to this requirement for service. Users also need to know and appreciate what is considered acceptable use of their computers, data, and programs. For users, a security policy should define acceptable use.

Owners:

Each piece of computing equipment is owned by someone, and the owner may not be a system user. An owner provides the equipment to users for a purpose, such as to further education, support commerce, or enhance productivity. A security policy should also reflect the expectations and needs of owners.

Beneficiaries:

A business has paying customers or clients; they are beneficiaries of the products and services offered by that business. At the same time, the general public may benefit in several ways: as a source of employment or by provision of infrastructure. For example, the government has customers: the citizens of its country, and "guests" who have visas enabling entry for various purposes and times. A university's customers include its students and faculty; other beneficiaries include the immediate community (which can take advantage of lectures and concerts on campus) and often the world population (enriched by the results of research and service). To varying degrees, these beneficiaries depend, directly or indirectly, on the existence of or access to computers, their data and programs, and their computational power. For this set of beneficiaries, continuity and integrity of computing are very important. In addition, beneficiaries value confidentiality and correctness of the data involved. Thus, the interests of beneficiaries of a system must be reflected in the system's security policy.

Balance Among All Parties:

A security policy must relate to the needs of users, owners, and beneficiaries. Unfortunately, the needs of these groups may conflict. A beneficiary might require immediate access to data, but owners or users might not want to bear the expense or inconvenience of providing access at all hours. Continuous availability may be a goal for users, but that goal is inconsistent with a need to perform preventive or emergency maintenance. Thus, the

[Type here] [Type here] [Type here]
security policy must balance the priorities of all affected communities.

13.2.3 Contents:

A security policy must identify its audiences: the beneficiaries, users, and owners. The policy should describe the nature of each audience and their security goals. Several other sections are required, including the purpose of the computing system, the resources needing protection, and the nature of the protection to be supplied. We discuss each one in turn.

Purpose:

The policy should state the purpose of the organization's security functions, reflecting the requirements of beneficiaries, users, and owners. For example, the policy may state that the system will "protect customers' confidentiality or preserve a trust relationship," "ensure continual usability," or "maintain profitability." There are typically three to five goals, such as:

- Promote efficient business operation.
- Facilitate sharing of information throughout the organization.
- Safeguard business and personal information.
- Ensure that accurate information is available to support business processes.
- Ensure a safe and productive place to work.
- Comply with applicable laws and regulations.

The security goals should be related to the overall goal or nature of the organization. It is important that the system's purpose be stated clearly and completely because subsequent sections of the policy will relate back to these goals, making the policy a goal-driven product.

Protected Resources:

A risk analysis will have identified the assets that are to be protected. These assets should be listed in the policy, in the sense that the policy lays out which items it addresses. For example, will the policy apply to all computers or only to those on the network? Will it apply to all data or only to client or management data? Will security be provided to all programs or only the ones that interact with customers? If the degree of protection varies from one service, product, or data type to another, the policy should state the differences. For example, data that uniquely identify clients may be protected more carefully than the names of cities in which clients reside.

Nature of the Protection:

The asset list tells us *what* should be protected. The policy should also indicate *who* should have access to the protected items. It may also indicate *how* that access will be ensured and *how* unauthorized people will be denied access. All the mechanisms described in this book are at your disposal in deciding which controls should protect which objects. In particular, the security policy should state what degree of protection should be provided to which kinds of resources.

13.2.4 Characteristics of a Good Security Plan:

If a security policy is written poorly, it cannot guide the developers and users in providing appropriate security mechanisms to protect important assets. Certain characteristics make a security policy a good one.

Coverage:

A security policy must be comprehensive: It must either apply to or explicitly exclude all possible situations. Furthermore, a security policy may not be updated as each new situation arises, so it must be general enough to apply naturally to new cases that occur as the system is used in unusual or unexpected ways.

Durability:

A security policy must grow and adapt well. In large measure, it will survive the system's growth and expansion without change. If written in a flexible way, the existing policy will be applicable to new situations. However, there are times when the policy must change (such as when government regulations mandate new security constraints), so the policy must be changeable when it needs to be. An important key to durability is keeping the policy free from ties to specific data or protection mechanisms that almost certainly will change. It is preferable to describe assets needing protection in terms of their function and characteristics, rather than in terms of specific implementation. Better still, we can separate the elements of the policy, having one policy statement for student grades and another for customers' proprietary data. Similarly, we may want to define one policy that applies to preserving the confidentiality of relationships, and another protecting the use of the system through strong authentication.

Realism:

The policy must be realistic. That is, it must be possible to implement the stated security requirements with existing technology. Moreover, the implementation must be beneficial in terms of time, cost, and convenience; the policy should not recommend a control that works but prevents the system or its users from performing their

[Type here] [Type here] [Type here]
activities and functions. It is important to make economically worthwhile investments in security, just as for any other careful business investment.

Usefulness:

An obscure or incomplete security policy will not be implemented properly, if at all. The policy must be written in language that can be read, understood, and followed by anyone who must implement it or is affected by it. For this reason, the policy should be succinct, clear, and direct.

13.2.5 Examples:

To understand the nature of security policies, we study a few examples to illustrate some of the points just presented.

Data Sensitivity Policy:

Our first example is from an organization that decided to classify all its data resources into four levels, based on how severe the effect might be if a resource were damaged. These levels are sensitive, personal or protected, company confidential and open. Then, the required protection was based on the resource's level. Finally, the organization analyzed its threats, their possible severities, and countermeasures, and their effectiveness, within each of the four levels. Although the phrases describing the degree of damage are open to interpretation, the intent of these levels is clear: All information assets are to be classified as sensitive, personal, confidential, or open, and protection requirements for these four types are detailed in the remainder of the organization's policy document.

Government Agency IT Security Policy:

The U.S. Department of Energy (DOE), like many government units, has established its own security policy. The following excerpt is from the policy on protecting classified material, although the form is appropriate for many unclassified uses as well. It is the policy of DOE that classified information and classified ADP [automatic data processing] systems shall be protected from unauthorized access (including the enforcement of need-to-know protections), alteration, disclosure, destruction, penetration, denial of service, subversion of security measures, or improper use as a result of espionage, criminal, fraudulent, negligent, abusive, or other improper actions. The DOE shall use all reasonable measures to protect ADP systems that process, store, transfer, or provide access to classified information, to include but not limited to the following: physical security, personnel security, telecommunications security, administrative security, and hardware and software security

[Type here] [Type here] [Type here]
measures. This order establishes this policy and defines responsibilities for the development, implementation, and periodic evaluation of the DOE program. The policy then continues for several more pages to list specific responsibilities for specific people. The cited paragraph is comprehensive, covering practically every possible source (espionage, crime, fraud, etc.) of practically every possible harm (unauthorized access, alteration, destruction, etc.), and practically every possible kind of control (physical, personnel, etc.).

The generality of the header paragraph is complemented by subsequent paragraphs giving specific responsibilities:

- "Each data owner shall determine and declare the required protection level of information . . ."
- "Each security officer shall . . . perform a risk assessment to identify and document specific . . . assets, . . . threats, . . . and vulnerability . . ."
- "Each manager shall...establish procedures to ensure that systems are continuously monitored...to detect security infractions . . ." and so on.

Internet Security Policy:

The Internet does not have a governing security policy per se, because it is a federation of users. Nevertheless, the Internet Society drafted a security policy for its members. The policy contains the following interesting portions.

- Users are individually responsible for understanding and respecting the security policies of the systems (computers and networks) they are using. Users are individually accountable for their own behaviour.
- Users have a responsibility to employ available security mechanisms and procedures for protecting their own data. They also have a responsibility for assisting in the protection of the systems they use.
- Computer and network service providers are responsible for maintaining the security of the systems they operate. They are further responsible for notifying users of their security policies and any changes to these policies.
- Vendors and system developers are responsible for providing systems which are sound, and which embody adequate security controls.
- Users, service providers, and hardware and software vendors are responsible for cooperating to provide security.
- Technical improvements in Internet security protocols should be sought on a continuing basis. At the same time, personnel

[Type here] [Type here] [Type here]
developing new protocols, hardware or software for the Internet
are expected to include security considerations as part of the
design and development process.

These statements clearly state to whom they apply and for what each
party is responsible.

13.3 PHYSICAL SECURITY

Much of this book has focused on technical issues in security and their technical solutions: firewalls, encryption techniques, and more. But many threats to security involve human or natural disasters, events that should also be addressed in the security plan. For this reason, in this section we consider how to cope with the nontechnical things that can go wrong. There are two pieces to the process of dealing with nontechnical problems: preventing things that can be prevented and recovering from the things that cannot be prevented. Physical security is the term used to describe protection needed outside the computer system. Typical physical security controls include guards, locks, and fences to deter direct attacks. In addition, there are other kinds of protection against less direct disasters, such as floods and power outages; these, too, are part of physical security. As we will see, many physical security measures can be provided simply by good common sense, a characteristic that Mark Twain noted "is a most uncommon virtue."

13.3.1 Natural Disasters:

Computers are subject to the same natural disasters that can occur to homes, stores, and automobiles. They can be flooded, burned, melted, hit by falling objects, and destroyed by earthquakes, storms, and tornadoes. Additionally, computers are sensitive to their operating environment, so excessive heat or inadequate power is also a threat. It is impossible to prevent natural disasters, but through careful planning it is possible to reduce the damage they inflict. Some measures can be taken to reduce their impact. Because many of these perils cannot be prevented or predicted, controls focus on limiting possible damage and recovering quickly from a disaster. Issues to be considered include the need for offsite backups, the cost of replacing equipment, the speed with which equipment can be replaced, the need for available computing power, and the cost or difficulty of replacing data and programs.

Flood:

Water from a natural flood comes from ground level, rising gradually, and bringing with it mud and debris. Often, there is time for an orderly shutdown of the computing system; at worst, the organization loses some of the processing in progress. At other times, such as when a dam breaks, a water pipe bursts, or the roof collapses in a storm, a sudden flood can overwhelm the system and

[Type here]

[Type here]

[Type here]

its users before anything can be saved. Water can come from above, below, or the side. The machinery may be destroyed or damaged by mud and water, but most computing systems are insured and replaceable by the manufacturer. Managers of unique or irreplaceable equipment who recognize the added risk sometimes purchase or lease duplicate redundant hardware systems to ensure against disruption of service. Even when the hardware can be replaced, we must be concerned about the stored data and programs. The system administrator may choose to label storage media in a way that makes it easy to identify the most important data. For example, green, yellow, and red labels may show which disks are the most sensitive, so that all red disks are moved from the data centre during a storm. Similarly, large plastic bags and waterproof tape can be kept near important equipment and media; they are used to protect the hardware and storage media in case of a burst pipe or other sudden flood. The real issue is protecting data and preserving the ability to compute. The only way to ensure the safety of data is to store backup copies in one or more safe locations.

Fire:

Fire is more serious than water; often there is not as much time to react, and human lives are more likely to be in immediate danger. To ensure that system personnel can react quickly, every user and manager should have a plan for shutting down the system in an orderly manner. Such a process takes only a few minutes but can make recovery much easier. This plan should include individual responsibilities for all people: some to halt the system, others to protect crucial media, others to close doors on media cabinets. Provision should be made for secondary responsibilities, so that onsite staff can perform duties for those who are not in the office. Water is traditionally used to put out fires, but it is not a good idea for use in computer rooms. In fact, more destruction can be the result of sprinklers than of the fires themselves. A fire sensor usually activates many sprinklers, dousing an entire room, even when the fire is merely some ignited paper in a wastebasket and of no threat to the computing system. Many computing centres use carbon dioxide extinguishers or an automatic system that sprays a gas such as Halon to smother a fire but leave no residue. Unfortunately, these gas systems work by displacing the oxygen in the room, choking the fire but leaving humans unable to breathe. Consequently, when these protection devices are activated, humans must leave, disabling efforts to protect media. The best defines for situations like these is careful placement of the computing facility. A windowless location with fire-resistant access doors and non-flammable full-height walls can prevent some fires from spreading from adjacent areas to the computing room. With a fire and smoke-resistant facility, personnel merely shut down the system and leave, perhaps carrying out the most important media. Fire prevention is quite effective, especially because most computer goods are not especially flammable. Advance planning, reinforced with simulation drills, can help make

[Type here]

[Type here]

[Type here]

good use of the small amount of time available before evacuation is necessary.

Other Natural Disasters:

Computers are subject to storms, earthquakes, volcanoes, and similar events. Although not natural disasters, building collapse, explosion, and damage from falling objects can be considered in the same category. These kinds of catastrophes are difficult to predict or estimate. But we know these catastrophes will occur. Security managers cope with them in several ways:

- developing contingency plans so that people know how to react in emergencies and business can continue
- ensuring physical assets, computers, buildings, devices, supplies against harm
- preserving sensitive data by maintaining copies in physically separated locations

13.3.2 Power Loss:

Computers need their food, electricity and they require a constant, pure supply of it. With a direct power loss, all computation ceases immediately. Because of possible damage to media by sudden loss of power, many disk drives monitor the power level and quickly retract the recording head if power fails. For certain time-critical applications, loss of service from the system is intolerable; in these cases, alternative complete power supplies must be instantly available.

Uninterruptible Power Supply:

One protection against power loss is an uninterruptible power supply. This device stores energy during normal operation so that it can return the backup energy if power fails. One form of uninterruptible power supply uses batteries that are continually charged when the power is on, but which then provide power when electricity fails. However, size, heat, flammability, and low output can be problems with batteries. Some uninterruptible power supplies use massive wheels that are kept in continuous motion when electricity is available. When the power fails, the inertia in the wheels operates generators to produce more power. Size and limited duration of energy output are problems with this variety of power supply. Both forms of power supplies are intended to provide power for a limited time, just long enough to allow the current state of the computation to be saved so that no computation is lost.

13.3.3 Surge Suppressor:

Another problem with power is its "cleanness." Although most people are unaware of it, a variation of 10 percent from the stated voltage of a line is considered acceptable, and some power lines vary

[Type here]

[Type here]

[Type here]

even more. A power line may always be 10 percent high or low. In many places, lights dim momentarily when a large appliance, such as an air conditioner, begins operation. When a large motor starts, it draws an exceptionally large amount of current, which reduces the flow to other devices on the line. When a motor stops, the sudden termination of draw can send a temporary surge along the line. Similarly, lightning strikes may send a momentary large pulse. Thus, instead of being constant, the power delivered along any electric line shows many brief fluctuations, called drops, spikes, and surges. A drop is a momentary reduction in voltage, and a spike or surge is a rise. For computing equipment, a drop is less serious than a surge. Most electrical equipment is tolerant of rather large fluctuations of current. These variations can be destructive to sensitive electronic equipment, however. Simple devices called "surge suppressors" filter spikes from an electric line, blocking fluctuations that would affect computers. These devices cost from \$20 to \$100; they should be installed on every computer, printer, or other connected component. More sensitive models are typically used on larger systems. As mentioned previously, a lightning strike can send a surge through a power line. To increase protection, personal computer users usually unplug their machines when they are not in use, as well as during electrical storms. Another possible source of destruction is lightning striking a telephone line. Because the power surge can travel along the phone line and into the computer or peripherals, the phone line should be disconnected from the modem during storms. These simple measures may save much work as well as valuable equipment.

13.3.4 Human Vandals:

Because computers and their media are sensitive to a variety of disruptions, a vandal can destroy hardware, software, and data. Human attackers may be disgruntled employees, bored operators, saboteurs, people seeking excitement, or unwitting bumbler. If physical access is easy to obtain, crude attacks using axes or bricks can be very effective. Physical attacks by unskilled vandals are often easy to prevent; a guard can stop someone approaching a computer installation with a threatening or dangerous object. When physical access is difficult, more subtle attacks can be tried, resulting in quite serious damage. People with only some sophisticated knowledge of a system can short-circuit a computer with a car key or disable a disk drive with a paper clip. These items are not likely to attract attention until the attack is completed.

Unauthorized Access and Use:

Films and newspaper reports exaggerate the ease of gaining access to a computing system. Still, as distributed computing systems become more prevalent, protecting the system from outside access becomes more difficult and more important. Interception is a form of unauthorized access; the attacker intercepts data and either

[Type here]

[Type here]

[Type here]

breaks confidentiality or prevents the data from being read or used by others. In this context, interception is a passive attack. But we must also be concerned about active interception, in the sense that the attacker can change or insert data before allowing it to continue to its destination.

Theft:

It is hard to steal a large mainframe computer. Not only is carrying it away difficult but finding a willing buyer and arranging installation and maintenance also require special assistance. However, printed reports, tapes, or disks can be carried easily. If done well, the loss may not be detected for some time. Personal computers, laptops, and personal digital assistants (PDAs, such as Palms or Blackberries) are designed to be small and portable. Diskettes and tape backup cartridges are easily carried in a shirt pocket or briefcase. Computers and media that are easy to carry are also easy to conceal. We can take one of three approaches to preventing theft: preventing access, preventing portability, or detecting exit.

Preventing Access:

The surest way to prevent theft is to keep the thief away from the equipment. However, thieves can be either insiders or outsiders. Therefore, access control devices are needed both to prevent access by unauthorized individuals and to record access by those authorized. A record of accesses can help identify who committed a theft. The oldest access control is a guard, not in the database management system sense but rather in the sense of a human being stationed at the door to control access to a room or to equipment. Guards offer traditional protection; their role is well understood, and the protection they offer is adequate in many situations. However, guards must be on duty continuously to be effective; providing breaks implies at least four guards for a 24-hour operation, with extras for vacation and illness. A guard must personally recognize someone or recognize an access token, such as a badge. People can lose or forget badges; terminated employees and forged badges are also problems. Unless the guard makes a record of everyone who has entered a facility, there is no way to know who (employee or visitor) has had access in case a problem is discovered.

The second oldest access control is a lock. This device is even easier, cheaper, and simpler to manage than a guard. However, it too provides no record of who has had access, and difficulties arise when keys are lost or duplicated. At computer facilities, it is inconvenient to fumble for a key when your hands are filled with tapes or disks, which might be ruined if dropped. There is also the possibility of piggybacking: a person walks through the door that someone else has just unlocked. Still, guards and locks provide simple, effective security for access to facilities such as computer rooms.

[Type here]

[Type here]

[Type here]

More exotic access control devices employ cards with radio transmitters, magnetic stripe cards (like 24-hour bank cards), and smart cards with chips containing electronic circuitry that makes them difficult to duplicate. Because each of these devices' interfaces with a computer, it is easy for the computer to capture identity information, generating a list of who entered and left the facility, when, and by which routes. Some of these devices operate by proximity, so that a person can carry the device in a pocket or clipped to a collar; the person obtains easy access even when hands are full. Because these devices are computer controlled, it is easy to invalidate an access authority when someone quits or reports the access token lost or stolen. The nature of the application or service determines how strict the access control needs to be. Working in concert with computer-based authentication techniques, the access controls can be part of defines in depth using multiple mechanisms to provide security.

Preventing Portability:

Portability is a mixed blessing. We can now carry around in our pockets devices that provide as much computing power as mainframes did twenty years ago. Portability is in fact a necessity in devices such as PDAs and mobile phones. And we do not want to permanently affix our personal computers to our desks, in case they need to be removed for repair or replacement. Thus, we need to find ways to enable portability without promoting theft. One antitheft device is a pad connected to cable, like those used to secure bicycles. The pad is glued to the desktop with extremely strong adhesive. The cables loop around the equipment and are locked in place. Releasing the lock permits the equipment to be moved. An alternative is to couple the base of the equipment to a secure pad, in much the same way that televisions are locked in place in hotel rooms. Yet a third possibility is a large, lockable cabinet in which the personal computer and its peripherals are kept when they are not in use. Some people argue that cables, pads, and cabinets are unsightly and, worse, they make the equipment inconvenient to use. Another alternative is to use movement-activated alarm devices when the equipment is not in use. Small alarms are available that can be locked to a laptop or PDA. When movement is detected, a loud, annoying whine or whistle warns that the equipment has been disturbed. Such an alarm is especially useful when laptops must be left in meeting or presentation rooms overnight or during a break. Used in concert with guards, the alarms can offer reasonable protection at reasonable cost.

Detecting Theft:

For some devices, protection is more important than detection. We want to keep someone from stealing certain systems or information at all costs. But for other devices, it may be enough to detect that an attempt has been made to access or steal hardware or software. For example, chaining down a disk makes it unusable.

[Type here]

[Type here]

[Type here]

Instead, we try to detect when someone tries to leave a protected area with the disk or other protected object. In these cases, the protection mechanism should be small and unobtrusive. One such mechanism is like the protection used by many libraries, bookstores, or department stores. Each sensitive object is marked with a special label. Although the label looks like a normal pressure-sensitive one, its presence can be detected by a machine at the exit door if the label has not been disabled by an authorized party, such as a librarian or salesclerk. Similar security code tags are available for vehicles, people, machinery, and documents. Some tags are enabled by radio transmitters. When the detector sounds an alarm, someone must apprehend the person trying to leave with the marked object.

13.3.5 Interception of Sensitive Information:

When disposing of a draft copy of a confidential report containing its sales strategies for the next five years, a company wants to be especially sure that the report is not reconstructable by one of its competitors. When the report exists only as hard copy, destroying the report is straightforward, usually accomplished by shredding or burning. But when the report exists digitally, destruction is more problematic. There may be many copies of the report in digital and paper form and in many locations (including on the computer and on storage media). There may also be copies in backups and archived in e-mail files. In this section, we look at several ways to dispose of sensitive information.

Shredding:

Shredders have existed for a long time, as devices used by banks, government agencies, and other organizations to dispose of large amounts of confidential data. Although most of the shredded data is on paper, shredders can also be used for destroying printer ribbons and some types of disks and tapes. Shredders work by converting their input to thin strips or pulp, with enough volume to make it infeasible for most people to try to reconstruct the original from its many pieces. When data are extremely sensitive, some organizations burn the shredded output for added protection.

Overwriting Magnetic Data:

Magnetic media present a special problem for those trying to protect the contents. When data are stored on magnetic disks, the ERASE or DELETE functions often simply change a directory pointer to free up space on the disk. As a result, the sensitive data are still recorded on the medium, and they can be recovered by analysis of the directory. A more secure way to destroy data on magnetic devices is to overwrite the data several times, using a different pattern each time. This process removes enough magnetic residue to prevent most people from reconstructing the original file. However, "cleaning" a disk in this fashion takes time. Moreover, a person using highly specialized equipment might be able to identify each separate

[Type here]

[Type here]

[Type here]

message, much like the process of peeling off layers of wallpaper to reveal the wall beneath.

Degaussing:

Degaussers destroy magnetic fields. Passing a disk or other magnetic medium through a degausser generates a magnetic flux so forceful that all magnetic charges are instantly realigned, thereby fusing all the separate layers. A degausser is a fast way to cleanse a magnetic medium, although there is still question as to whether it is adequate for use in the most sensitive of applications. For most users, a degausser is a fast way to neutralize a disk or tape, permitting it to be reused by others.

Protecting Against Emanation: Tempest:

Computer screens emit signals that can be detected from a distance. In fact, any components, including printers, disk drives, and processors, can emit information. Tempest is a U.S. government program under which computer equipment is certified as emission-free (that is, no detectable emissions). There are two approaches for preparing a device for Tempest certification: enclosing the device and modifying the emanations. The obvious solution to preventing emanations is to trap the signals before they can be picked up. Enclosing a device in a conductive case, such as copper, diffuses all the waves by conducting them throughout the case. Copper is a good conductor, and the waves travel much better through copper than through the air outside the case, so the emissions are rendered harmless. This solution works very well with cable, which is then enclosed in a solid, emanation-proof shield. Typically, the shielded cable is left exposed so that it is easy to inspect visually for any signs of tapping or other tampering. The shielding must be complete. That is, it does little good to shield a length of cable but not also shield the junction box at which that cable is connected to a component. The line to the component and the component itself must be shielded, too. The shield must enclose the device completely. If top, bottom, and three sides are shielded, emanations are prevented only in those directions. However, a solid copper shield is useless in front of a computer screen. Covering the screen with a fine copper mesh in an intricate pattern carries the emanation safely away. This approach solves the emanation problem while still maintaining the screen's usability.

Entire computer rooms or even whole buildings can be shielded in copper so that large computers inside do not leak sensitive emanations. Although it seems appealing to shield the room or building instead of each component, the scheme has significant drawbacks.

A shielded room is inconvenient because it is impossible to expand the room easily as needs change. The shielding must be done carefully, because any puncture is a possible point of emanation. Furthermore, continuous metal pathways, such as water

[Type here]

[Type here]

[Type here]

pipes or heating ducts, act as antennas to convey the emanations away from their source. Emanations can also be designed in such a way that they cannot be retrieved. This process is like generating noise to jam or block a radio signal. With this approach, the emanations of a piece of equipment must be modified by addition of spurious signals. Additional processors are added to Tempest equipment specifically to generate signals that fool an interceptor. The exact Tempest modification methods are classified.

As might be expected, Tempest-enclosed components are larger and heavier than their unprotected counterparts. Tempest testing is a rigorous program of the U.S. Department of Defence. Once a product has been approved, even a minor design modification, such as changing from one manufacturer's power supply to an equivalent one from another manufacturer, invalidates the Tempest approval. Therefore, these components are costly, ranging in price from 10 percent to 300 percent more than similar non-Tempest products. They are most appropriate in situations in which the data to be confined are of great value, such as top-level government information. Other groups with less dramatic needs can use other less rigorous shielding.

13.3.6 Contingency Planning:

The key to successful recovery is adequate preparation. Seldom does a crisis destroy irreplaceable equipment; most computing systems, personal computers to mainframes are standard, off-the-shelf systems that can be easily replaced. Data and locally developed programs are more vulnerable because they cannot be quickly substituted from another source. Let us look more closely at what to do after a crisis occurs.

Backup:

In many computing systems, some data items change frequently, whereas others seldom change. For example, a database of bank account balances changes daily, but a file of depositors' names and addresses changes much less often. Also, the number of changes in each period is different for these two files. These variations in number and extent of change relate to the amount of data necessary to reconstruct these files in the event of a loss. A backup is a copy of all or a part of a file to assist in re-establishing a lost file. In professional computing systems, periodic backups are usually performed automatically, often at night when system usage is low. Everything on the system is copied, including system files, user files, scratch files, and directories, so that the system can be regenerated after a crisis. This type of backup is called a complete backup. Complete backups are done at regular intervals, usually weekly or daily, depending on the criticality of the information or service provided by the system.

[Type here]

[Type here]

[Type here]

Major installations may perform revolving backups, in which the last several backups are kept. Each time a backup is done, the oldest backup is replaced with the newest one. There are two reasons to perform revolving backups: to avoid problems with corrupted media and to allow users or developers to retrieve old versions of a file. Another form of backup is a selective backup, in which only files that have been changed (or created) since the last backup are saved. In this case, fewer files must be saved, so the backup can be done more quickly. A selective backup combined with an earlier complete backup gives the effect of a complete backup in the time needed for only a selective backup. For each type of backup, we need the means to move from the backup forward to the point of failure. That is, we need a way to restore the system in the event of failure. In critical transaction systems, we address this need by keeping a complete record of changes since the last backup.

Sometimes, the system state is captured by a combination of computer- and paper-based recording media. For example, if a system handles bank teller operations, the individual tellers duplicate their processing on paper records the deposit and withdrawal slips that accompany your bank transactions; if the system fails, the staff restores the latest backup version and reapplies all changes from the collected paper copies. Or the banking system creates a paper journal, which is a log of transactions printed just as each transaction completes.

Personal computer users often do not appreciate the need for regular backups. Even minor crises, such as a failed piece of hardware, can seriously affect personal computer users. With a backup, users can simply change to a similar machine and continue work.

Offsite Backup:

A backup copy is useless if it is destroyed in the crisis, too. Many major computing installations rent warehouse space some distance from the computing system, far enough away that a crisis is not likely to affect the offsite location at the same time. As a backup is completed, it is transported to the backup site. Keeping a backup version separate from the actual system reduces the risk of its loss. Similarly, the paper trail is also stored somewhere other than at the main computing facility. Personal computer users concerned with integrity can take home a copy of important disks as protection or send a copy to a friend in another city. If both secrecy and integrity are important, a bank vault, or even a secure storage place in another part of the same building can be used. The worst place to store a backup copy is where it usually is stored: right next to the machine.

Networked Storage:

With today's extensive use of networking, using the network to implement backups is a good idea. Storage providers sell space in which you can store data; think of these services as big network-attached disk drives. You rent space just as you would consume electricity: You pay for what you use. The storage provider needs to provide only enough total space to cover everyone's needs, and it is easy to monitor usage patterns and increase capacity as combined needs rise. Networked storage is perfect for backups of critical data because you can choose a storage provider whose physical storage is not close to your processing. In this way, physical harm to your system will not affect your backup. You do not need to manage tapes or other media and physically transport them offsite.

Cold Site:

Depending on the nature of the computation, it may be important to be able to recover from a crisis and resume computation quickly. A bank, for example, might be able to tolerate a four-hour loss of computing facilities during a fire, but it could not tolerate a ten-month period to rebuild a destroyed facility, acquire new equipment, and resume operation. Most computer manufacturers have several spare machines of most models that can be delivered to any location within 24 hours in the event of a real crisis. Sometimes the machine will come straight from assembly; other times the system will have been in use at a local office. Machinery is seldom the hard part of the problem. Rather, the hard part is deciding where to put the equipment in order to begin a temporary operation. A cold site or shell is a facility with power and cooling available, in which a computing system can be installed to begin immediate operation. Some companies maintain their own cold sites, and other cold sites can be leased from disaster recovery companies. These sites usually come with cabling, fire prevention equipment, separate office space, telephone access, and other features. Typically, a computing center can have equipment installed and resume operation from a cold site within a week of a disaster.

Hot Site:

If the application is more critical or if the equipment needs are more specialized, a hot site may be more appropriate. A hot site is a computer facility with an installed and ready-to run computing system. The system has peripherals, telecommunications lines, power supply, and even personnel ready to operate on short notice. Some companies maintain their own; other companies subscribe to a service that has available one or more locations with installed and running computers. To activate a hot site, it is necessary only to load software and data from offsite backup copies.

Numerous services offer hot sites equipped with every

[Type here] [Type here] [Type here]
popular brand and model of system. They provide diagnostic and system technicians, connected communications lines, and an operations staff. The hot site staff also assists with relocation by arranging transportation and housing, obtaining needed blank forms, and acquiring office space. Because these hot sites serve as backups for many customers, most of whom will not need the service, the annual cost to any one customer is low. The cost structure is like insurance: The likelihood of an auto accident is low, so the premium is reasonable, even for a policy that covers the complete replacement cost of an expensive car. Notice, however, that the first step in being able to use a service of this type is a complete and timely backup.

13.4 SUMMARY

- The administration of security draws on skills slightly different from the technical skills. The security administrator must understand not just security assets, threats, vulnerabilities, and controls, but management and implementation. In this chapter we examined how security is administered.
- An organizational security policy is a document that specifies the organization's goals regarding security. It lists policy elements that are statements of actions that must or must not be taken to preserve those goals. Policy documents often lead to implementational procedures. Also, user education and awareness activities ensure that users are aware of policy restrictions.
- Physical security concerns the physical aspects of computing: the devices themselves and harm that can come to them because of the buildings in which they are contained. Physical security addresses two branches of threats: natural threats to buildings and the infrastructure, and human threats. Redundancy and physical controls address physical security threats.
- The administration of security has a strong human component, from the writing of plans and policies, to the mental work in performing a risk analysis, to the human guards that implement or reinforce many physical controls.
- By no means have we covered all physical security in this brief introduction. Professionals become experts at individual aspects, such as fire control or power provision. We must protect the facility against many sorts of disasters, from weather to chemical spills and vehicle crashes to explosions. It is impossible to predict what will occur or when.
- The physical security manager must consider all assets and a wide range of harm. Malicious humans seeking physical access are a different category of threat agent. With them, you can

[Type here]

[Type here]

[Type here]

consider motive or objective: is it theft of equipment, disruption of processing, interception of data, or access to service? Fences, guards, solid walls, and locks will deter or prevent most human attacks. But you always need to ask where weaknesses remain; a solid wall has a weakness in every door and window.

- The primary physical controls are strength and duplication. Strength means overlapping controls implementing a defines-in-depth approach so that if one control fails, the next one will protect. People who built ancient castles practiced this philosophy with moats, walls, drawbridges, and arrow slits. Duplication means eliminating single points of failure. Redundant copies of data protect against harm to one copy from any cause. Spare hardware components protect against failures.

13.5 REVIEW QUESTIONS

- a) What are the various audiences and how is balance managed among all the audiences?
- b) Explain the contents of a good security plan.
- c) What are the characteristics of a good security plan?
- d) What natural disasters are computers prone to and how can they be saved?
- e) What are the ways in which human vandals can cause problems in physical security?
- f) How can sensitive information be intercepted?
- g) Write a short note on contingency planning.

13.6 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- *Security in Computing* by C. P. Pfleeger, and S. L. Pfleeger, Pearson Education.
- *Computer Security: Art and Science* by Matt Bishop, Pearson Education.
- *Cryptography and Network Security: Principles and practice* by Stallings
- *Network Security* by Kaufman, Perlman, Speciner
- *Network Security : A Beginner's Guide* by Eric Maiwald, TMH
- *Java Network Security* by Macro Pistoia, Pearson Education

[Type here]

[Type here]

[Type here]

- *Principles of information security* by Whitman, Mattord, Thomson



100%

PRIVACY

Unit Structure:

- 14.0 Objectives
- 14.1 Introduction
- 14.2 Privacy Concepts
 - 14.2.1. Aspects of Information Privacy
 - 14.2.2. Computer Related Privacy Problems
- 14.3 Privacy Principles and Policies
 - 14.3.1. Fair Information Policies
 - 14.3.2. U.S. Privacy Laws
 - 14.3.3. Controls on Commercial Web Sites
 - 14.3.4. Non – U.S. Privacy Principles
 - 14.3.5. Anonymity, Multiple Identities
 - 14.3.6. Government and Privacy
 - 14.3.7. Identity Theft
- 14.4 Authentication and Privacy
 - 14.4.1. What authentication means
- 14.5 Summary
- 14.6 Review Questions
- 14.7 Bibliography, References and Further Reading

14.0 OBJECTIVES

After reading this chapter, the reader will be able to understand

- the privacy aspect of security and
- authentication effects on privacy.

14.1 INTRODUCTION

Computers did not invent or even cause privacy issues; we had those long before computers and probably even before written language. But computers' high-speed processing and data storage

and transmission capabilities made possible data collection and correlation that affect privacy. Because privacy is part of confidentiality, it is an aspect of computer security.

Privacy is a human right, although people can legitimately disagree over when or to what extent privacy is deserved; this disagreement may have cultural, historical, or personal roots. Laws and ethics can set the baseline for and enforce expectations of privacy. But inherently, the right to privacy depends on the situation and the affected parties. And just as confidentiality, integrity, and availability can conflict, so too can privacy and other aspects of security. We won't take a position on when a right to privacy should be enforceable because that is outside the scope of this book. You might characterize the presentation of this chapter as "assuming a particular right to privacy exists, what are its implications in computing and information technology?" We as citizens help decide the contours of privacy rights; we as computer security experts implement those decisions in computer systems.

Privacy is also a broad topic, affected by computing but not just a security topic. We don't want to try to survey all possible privacy issues in this chapter, just those inextricably linked to computer security. In this chapter we look at the meaning of information privacy. We examine identification and authentication, two familiar aspects of computing that have significant privacy implications.

14.2 PRIVACY CONCEPTS

In this section we examine privacy, first from its general or common usage and then as it applies in technological situations.

14.2.1. Aspects of Information Privacy:

Information privacy has three aspects: sensitive data, affected parties, and controlled disclosure. In fact, these aspects are like the three elements of access control from earlier chapters: subject, object, and access rights. We examine these three in turn.

Controlled Disclosure:

What is privacy? A good working definition is that privacy is the right to control who knows certain aspects about you, your communications, and your activities. In other words, you voluntarily choose who can know things about you and what those things are. People ask you for your telephone number: your auto mechanic, a clerk in a store, your tax authority, a new business contact, or a cute person in a bar. You consider why the person wants the number and decide whether to give it out. But the key point is *you* decide. So, privacy is something over which you have considerable influence.

You do not have complete control, however. Once you give your number to someone else, your control is diminished because it depends in part on what someone else does. As soon as you give out your number, you transfer authority and control to someone else. You may say “don't give my number to anyone else”, “use discretion”, or “I am sensitive about my privacy”, but you do not control the other person. You must trust the other person to comply with your wishes, whether you state them explicitly or not. This problem is like the propagation problem of computer security: Anyone who has access to an object can copy, transfer, or propagate that object or its content to others without restriction.

Sensitive Data:

Someone asks you for your shoe size; you might answer, “I'm a very private person and cannot imagine why you would want to know such an intimate detail” or you could say “10C”; some people find that data more sensitive than others. We know things people usually consider sensitive, such as financial status, certain health data, unsavoury events in their past, and the like, so if you learn something you consider sensitive about someone, you will keep it quiet. But most of us are not too sensitive about our shoe size, so we don't normally protect that if we learn it about someone else. Of course, if a friend told me not to pass that along, I wouldn't. It is not up to me to question why someone else considers something private. Here are examples (in no order) of data many people consider private.

- identity, the ownership of private data and the ability to control its disclosure
- finances, credit, bank details
- legal matters
- medical conditions, drug use, DNA, genetic predisposition to illnesses
- voting, opinions, membership in advocacy organizations
- preferences: religion, sexuality
- biometrics, physical characteristics, polygraph results, fingerprints
- diaries, poems, correspondence, recorded thoughts
- privileged communications with professionals such as lawyers, accountants, doctors, counsellors, and clergy
- performance: school records, employment ratings
- activities: reading habits, web browsing, music, art, videos
- air travel data, general travel data, a person's location (present and past)
- communications: mail, e-mail, telephone calls, spam
- history: "youthful indiscretions," past events

- illegal activities, criminal records

Privacy is also affected by who you are. When you are in a room of people you don't know, perhaps at a reception, someone may come up to you and say, "So you are the man who baked that beautiful cake over there; I really appreciate your skills as a pastry chef". It feels kind of nice to get that kind of recognition. Conversely, a friend was frequently on local television; she far preferred having dinner at home instead of going to a restaurant because she had grown tired of people rushing up to her saying "you're [Olga], I see you all the time on TV". Public personalities cherish the aspects of privacy they retain. World champion athletes cannot avoid having their results made public, whereas you might not want everyone to know how poorly you finished in the last event. Culture also influences what people consider sensitive. In general, a person's privacy expectations depend on context: who is affected and what the prevailing norm of privacy is.

Affected Subject:

This brings us to another point about privacy: Individuals, groups, companies, organizations, and governments all have data they consider sensitive. So far, we have described privacy from the standpoint of a person. Companies may have data they consider private or sensitive: product plans, key customers, profit margins, and newly discovered technologies. For organizations such as companies, privacy usually relates to gaining and maintaining an edge over the competition.

Other organizations, for example, schools, hospitals, or charities, may need to protect personal data on their students, patients, or donors, or they may want to control negative news, and so forth. Governments consider military and diplomatic matters sensitive, but they also recognize a responsibility to keep confidential data they collect from citizens, such as tax information. We may use terms like subject or owner to cover privacy issues affecting people, groups, and the like.

Privacy is an aspect of confidentiality. As we have learned throughout this book, the three security goals of confidentiality, integrity, and availability conflict, and confidentiality frequently conflicts with availability. If you choose not to have your telephone number published in a directory, that also means some people will not be able to reach you by telephone.

14.2.2. Computer Related Privacy Problems:

You may notice that many of the kinds of sensitive data and
[Unedited Version: Information Security](#)

many of the points about privacy have nothing to do with computers. These sensitivities and issues predate computers. Computers and networks have only affected the feasibility of some unwanted disclosures. Public records offices have long been open for people to study the data held there, but the storage capacity and speed of computers have given us the ability to amass, search, and correlate. Search engines have given us the ability to find one data item out of billions, the equivalent of finding one sheet of paper out of a warehouse full of boxes of papers. Furthermore, the openness of networks and the portability of technology (such as laptops, PDAs, cell phones, and memory devices) have greatly increased the risk of disclosures affecting privacy. Eight dimensions of privacy (specifically as it relates to the web, although the definitions carry over naturally to other types of computing) are as follows:

- *Information collection*: Data are collected only with knowledge and explicit consent.
- *Information usage*: Data are used only for certain specified purposes.
- *Information retention*: Data are retained for only a set period.
- *Information disclosure*: Data are disclosed to only an authorized set of people.
- *Information security*: Appropriate mechanisms are used to ensure the protection of the data.
- *Access control*: All modes of access to all forms of collected data are controlled.
- *Monitoring*: Logs are maintained showing all accesses to data.
- *Policy changes*: Less restrictive policies are never applied after-the-fact to already obtained data.

Here are the privacy issues that have come about through use of computers.

Data Collection:

As we have previously said, advances in computer storage make it possible to hold and manipulate huge numbers of records. Disks on ordinary consumer PCs are measured in gigabytes (10^9 bytes), and commercial storage capacities often measure in terabytes (10^{12} bytes). In 2006, EMC Corporation announced a storage product whose capacity exceeds one petabyte (10^{15} bytes). Indiana University plans to acquire a supercomputer with one petabyte of storage, and the San Diego Supercomputer Center has online storage of one petabyte and offline archives of seven petabytes. Estimates of Google's stored data are also in the petabyte range. We have both devices to store massive amounts of data and the data to fill those devices. Whereas physical space limited storing

(and locating) massive amounts of printed data, electronic data take relatively little space. We never throw away data; we just move it to slower secondary media or buy more storage.

No Informed Consent:

Where do all these bytes come from? Although some are from public and commercial sources (newspapers, web pages, digital audio, and video recordings) and others are from intentional data transfers (tax returns, a statement to the police after an accident, readers' survey forms, school papers), still others are collected without announcement. Telephone companies record the date, time, duration, source, and destination of each telephone call. ISPs track sites visited. Some sites keep the IP address of each visitor to the site. The user is not necessarily aware of this third category of data collection and thus cannot be said to have given informed consent.

Loss of Control:

We realize that others may keep data we give them. When you order merchandise online, you know you have just released your name, probably some address and payment data, and the items you purchased. Or when you use a customer appreciation card at a store, you know the store can associate your identity with the things you buy. Having acquired your data, a merchant can redistribute it to anyone. The fact that you booked one brand of hotel room through a travel agent could be sold to other hotels. If you frequently telephone someone in one city and have taken several plane trips to that city, local stores, restaurants, or tourist attractions in that city might want your name. You have little control over dissemination (or re-dissemination) of your data. We do not always appreciate the ramifications of lost control. Suppose in a moment of anger you dash off a strong note to someone. Although 100 years ago you would have written the note on paper and 50 years ago you would have voiced the comment by telephone, now you post the message to a blog.

Next suppose you have a change of heart and you want to retract your angry note. Let us consider how you would deal with these three forms of the communication. For the written note, you write a letter of apology, your recipient tears up your first note, and no trace remains. In the second case you telephone to apologize and all that remains is a memory. As for the blog, you delete your posting. However, several other people might have seen your original posting and copied it to blogs or other web sites that you do not control. Search engines might have found the original or copies. And other people might have picked up your words and circulated them in e-mail. Thus, with letters and phone calls, we can usually obliterate something we want to retract. But once something is out of your control on the web, it may never be deleted. This example concerned something you wrote.

A similar situation concerns something written about you. Someone else has posted something on the web that is personal about you and you want it removed. Even if the poster agrees, you may not be able to remove all its traces. Finally, some people are finding they reveal more than they should on sites like myspace.com. Prospective employees are being turned down for jobs because of things they have written. The web is a great historical archive, but because of archives, caches, and mirror sites, things posted on the web may never go away.

A second issue of loss of control concerns data exposure. Suppose a company holds data about you and that company's records are exposed in a computer attack. The company may not be responsible for preventing harm to you, compensating you if you are harmed, or even informing you of the event.

Ownership of the Data:

In the cases just described, customer details are being marketed. Information about you is being sold and you have no control; nor do you get to share in the profit. Even before computers customer data were valuable. Mailing lists and customer lists were company assets that were safeguarded against access by the competition. Sometimes companies rented their mailing lists when there was not a conflict with a competitor. But in those cases, the subject of the data, the name on the list, did not own the right to be on the list or not. With computers the volume and sources of data have increased significantly, but the subject still has no rights.

These issues, loss of control, no informed consent, no ownership of data have significant privacy implications. The way we address these kinds of issues is with policies, written statements of practice that inform all affected parties of their rights.

14.3 PRIVACY PRINCIPLES AND POLICIES

In the United States, interest in privacy and computer databases dates back at least to the early 1970s. (It is worth noting that the U.S. Watergate burglary occurred in 1972. Shortly after, reports surfaced that Nixon maintained an enemies list and had used IRS records as a means of combating adversaries. Thus, people in the United States were sensitive about privacy at that time. Public concern for privacy has varied over the years.) In the early 1970s, a committee developed privacy principles that have affected U.S. laws and regulations and that also set the path for privacy legislation in other countries.

14.3.1. Fair Information Policies:

Unedited Version: Information Security

In 1973 Willis Ware of the RAND Corporation chaired a committee to advise the Secretary of the U.S. Department of Human Services on privacy issues. The report proposes a set of principles of fair information practice:

- *Collection limitation:* Data should be obtained lawfully and fairly.
- *Data quality:* Data should be relevant to their purposes, accurate, complete, and up to date.
- *Purpose specification:* The purposes for which data will be used should be identified and the data destroyed if no longer necessary to serve that purpose.
- *Use limitation:* Use for purposes other than those specified is authorized only with consent of the data subject or by authority of law.
- *Security safeguards:* Procedures to guard against loss, corruption, destruction, or misuse of data should be established.
- *Openness:* It should be possible to acquire information about the collection, storage, and use of personal data systems.
- *Individual participation:* The data subject normally has a right to access and to challenge data relating to her.
- *Accountability:* A data controller should be designated and accountable for complying with the measures to give effect to the principles.

These principles describe the rights of individuals, not requirements on collectors; that is, the principles do not require protection of the data collected. Ware raises the problem of linking data in multiple files and of overusing keys, such as social security numbers, that were never intended to be used to link records. And although he saw that society was moving toward a universal identity number, he feared that movement would be without plan (and hence without control). He was right, even though he could not have foreseen the amount of data exchanged 30 years later. Turn and Ware consider protecting the data themselves, recognizing that collections of data will be attractive targets for unauthorized access attacks. They suggest four ways to protect stored data:

- Reduce exposure by limiting the amount of data maintained, asking for only what is necessary and using random samples instead of complete surveys.
- Reduce data sensitivity by interchanging data items or adding subtle errors to the data (and warning recipients that the data

have been altered).

- Anonymize the data by removing or modifying identifying data items.
- Encrypt the data.

14.3.2. U.S. Privacy Laws:

Ware and his committee expected these principles to apply to all collections of personal data on individuals. Unfortunately, that is not the way the legislation developed. The Ware committee report led to the 1974 Privacy Act (5 USC 552a), which embodies most of these principles, although that law applies only to data maintained by the U.S. government. The Privacy Act is a broad law, covering all data collected by the government. It is the strongest U.S. privacy law because of its breadth: It applies to all personal data held anywhere in the government.

The United States subsequently passed laws protecting data collected and held by other organizations, but these laws apply piecemeal, by individual data type. Consumer credit is addressed in the Fair Credit Reporting Act, healthcare information in the Health Insurance Portability and Accountability Act (HIPAA), financial service organizations in the GrammLeachBliley Act (GLBA), children's web access in the Children's Online Privacy Protection Act (COPPA), and student records in the Federal Educational Rights and Privacy Act. Not surprisingly these separate laws are inconsistent in protecting privacy. Laws and regulations do help in some aspects of privacy protection. Antón et al. investigated the impact of the HIPAA law by analyzing companies' posted privacy policies before and after the privacy provisions of the law became effective. They found the following in policies posted after HIPAA:

- Statements on data transfer (to other organizations) were more explicit than before HIPAA.
- Consumers still had little control over the disclosure or dissemination of their data.
- Statements were longer and more complex, making them harder for consumers to understand.
- Even within the same industry branch (such as drug companies), statements varied substantially, making it hard for consumers to compare policies.
- Statements were unique to specific web pages, meaning they covered more precisely the content and function of a page.

A problem with many laws is that the target areas of the laws still overlap: Which law (if any) would require privacy protection of a university student's health center bills paid by credit card? The laws

have different protection and handling requirements, so it is important to determine which law applies to a single piece of data. Also, gaps between laws are not covered. As new technologies (such as computers, the Internet, or cell phones) are developed, either existing privacy laws must be reinterpreted by the courts to apply to the new technologies or new laws must be passed, which takes time. Sometimes the privacy provisions of a law are a second purpose, somewhat disguised by the first purpose of the law. As an example, the primary purpose of HIPAA was to ensure that people who left or were terminated from one job had health insurance to cover them until they got another job; the privacy aspects were far less prominent as the law was being developed.

14.3.3. Controls on Commercial Web Sites:

The e-Government Act places strong controls on government data collection through web sites. As we described, privacy outside the government is protected by law in some areas, such as credit, banking, education, and healthcare. But there is no counterpart to the e-Government Act for private companies.

No Deceptive Practices:

The Federal Trade Commission has the authority to prosecute companies that engage in deceptive trade or unfair business practices. If a company advertises in a false or misleading way, the FTC can sue. The FTC has used that approach on web privacy: If a company advertises a false privacy protection that is, if the company says it will protect privacy in some way but does not do so the FTC considers that false advertising and can take legal action. Because of the FTC, privacy notices at the bottom of web sites do have meaning. This practice leads to a bizarre situation, however. A company can collect personal information and pass it in any form to anyone, if the company's privacy policy said it would do so, or at least the policy did not say it would not do so. Vowing to maintain privacy and intentionally not doing so is an illegal deceptive practice. Stating an intention to share data with marketing firms or "other third parties" makes such sharing acceptable, even though the third parties could be anyone.

Examples of Deceptive Practices:

The FTC settled a prosecution in 2005 against CartManager International, a firm that runs familiar web shopping cart software to collect items of an order, obtain the purchaser's name and address, and determine shipping and payment details. This software runs as an application under other well-known retail merchants' web sites to handle order processing. Some of these other retailers had privacy statements on their web sites saying, in effect, that they would not sell or distribute customers' data, but CartManager did sell the data it collected. The FTC held that the relationship to CartManager was

invisible to users, and so the policy from the online merchants applied also to CartManager.

In another case, Antón analyzed the privacy policy posted on the web site of Jet Blue airlines and found it misleading. Jet Blue stated that it would not disclose passenger data to third parties. It then released passenger data, "in response to a special request from the Department of Défense" to Torch Concepts, which in turn passed it to the Defense Department to use to test passenger screening algorithms for airline security. The data in question involved credit card information: Clearly the only reason for Jet Blue to have collected those data from passengers was to process charges for airline tickets. The analysis by Antón is interesting for two reasons: First, Jet Blue violated its own policy. Second, the Department of Defense may have circumvented the e-Government Act by acquiring from a private company data it would not have been able to collect as a government entity. The purpose for which the data were originally collected was ordinary business and accounting activities of Jet Blue. Using those same records to screen for terrorists was outside the scope of the original data collection. Commercial sites have no standard of content comparable to the FTC recommendation from the e-Government Act. Some companies display solid and detailed privacy statements that they must obey. On the other hand, you may find no statement at all, which gives the company the greatest flexibility because it is impossible to lie when saying nothing. Cranor makes some recommendations for useful web privacy policies.

14.3.4. Non – U.S. Privacy Principles:

In 1981, the Council of Europe (an international body of 46 European countries, founded in 1949) adopted Convention 108 for the protection of individuals with regard to the automatic processing of personal data, and in 1995, the European Union (E.U.) adopted Directive 95/46/EC on the processing of personal data. Directive 95/46/EC, often called the European Privacy Directive, requires that rights of privacy of individuals be maintained and that data about them be

- processed fairly and lawfully
- collected for specified, explicit and legitimate purposes and not further processed in a way incompatible with those purposes
- adequate, relevant, and not excessive in relation to the purposes for which they are collected and/or further processed
- accurate and, where necessary, kept up to date; every reasonable step must be taken to ensure that inaccurate or incomplete data having regard for the purposes for which they were collected or for which they are further processed, are erased or rectified

- kept in a form that permits identification of data subjects for no longer than is necessary for the purposes for which the data were collected or for which they are further processed. In addition, individuals have the right to access data collected about them, to correct inaccurate or incomplete data, and to have those corrections sent to those who have received the data. The report adds three more principles to the Fair Information Policies.
- *Special protection for sensitive data:* There should be greater restrictions on data collection and processing that involves "sensitive data." Under the E.U. data protection directive, information is sensitive if it involves "racial or ethnic origin, political opinions, religious beliefs, philosophical or ethical persuasion . . . [or] health or sexual life."
- *Data transfer:* This principle explicitly restricts authorized users of personal information from transferring that information to third parties without the permission of the data subject.
- *Independent oversight:* Entities that process personal data should not only be accountable but should also be subject to independent oversight. In the case of the government, this requires oversight by an office or department that is separate and independent from the unit engaged in the data processing. Under the data protection directive, the independent overseer must have the authority to audit data processing systems, investigate complaints brought by individuals, and enforce sanctions for noncompliance.

These requirements apply to governments, businesses, and other organizations that collect personal data. Since the 1995 directive, the European Union has extended coverage to telecommunications systems and made other changes to adapt to advances in technology. In addition to European countries and the United States, other countries, such as Japan, Australia, and Canada, have passed laws protecting the privacy of personal data about individuals. Different laws in different jurisdictions will inevitably clash. Relations between the European Union and the United States have been strained over privacy because the E.U. law forbids sharing data with companies or governments in countries whose privacy laws are not as strong as those of the E.U.

14.3.5. Anonymity, Multiple Identities:

One way to preserve privacy is to guard our identity. Not every context requires us to reveal our identity, so some people wear a form of electronic mask.

Anonymity:

A person may want to do some things anonymously. For

example, a rock star buying a beach house might want to avoid unwanted attention from neighbours, or someone posting to a dating list might want to view replies before making a date. Some people like the anonymity of the web because it reduces fears of discrimination. Fairness in housing, employment, and association are easier to ensure when the basis for potential discrimination is hidden. Also, people researching what they consider a private matter, such as a health issue or sexual orientation, are more likely to seek first information from what they consider an anonymous source, turning to a human when they have found out more about their situation.

Anonymity creates problems, too. How does an anonymous person pay for something? A trusted third party (for example, a real estate agent or a lawyer) can complete the sale and preserve anonymity. But then you need a third party and the third party knows who you are. Chaum studied this problem and devised a set of protocols by which such payments could occur without revealing the buyer to the seller.

Multiple Identities Linked or Not:

Most people already have multiple identities. To your bank you might be the holder of account 123456, to your motor vehicles bureau you might be the holder of driver's license number 234567, and to your credit card company you might be the holder of card 345678. For their purposes, these numbers are your identity; the fact that each may (or may not) be held in your name is irrelevant. The name does become important if it is used to link these records. How many people share your name? Can (or should) it serve as a key value to link these separate databases? We ignore the complication of misspellings and multiple valid forms (with and without middle initials, with full middle name, with one of two middle names if you have them, and so forth). Suppose you changed your name legally but never changed the name on your credit card; then your name could not be used as a key on which to link. Another possible link field is address. However, trying to use an address on which to link presents another risk: Perhaps a criminal lived in your house before you bought it. You should not have to defend your reputation because of a previous occupant. Now we need to match on date, too, so we connect only people who lived in a house at the same time. Then we need to address the problem of group houses or roommates of convenience, and so forth. As computer scientists, we know we can program all these possibilities, but that requires careful and time-consuming consideration of the potential problems before designing the solution. We can also see the potential for misuse and inaccuracy.

Linking identities correctly to create dossiers and break anonymity creates privacy risks but linking them incorrectly creates

much more serious risks for the use of the data and the privacy of affected people. If we think carefully, we can determine many of the ways such a system would fail, but that approach is potentially expensive and time consuming. The temptation to act quickly but inaccurately will also affect privacy.

Pseudonymity:

Sometimes, full anonymity is not wanted. A person may want to order flower bulbs but not be placed on a dozen mailing lists for gardening supplies. But the person does want to be able to place similar orders again, asking for the same colour tulips as before. This situation calls for pseudonyms, unique identifiers that can be used to link records in a server's database but that cannot be used to trace back to a real identity. Multiple identities can also be convenient, for example, having a professional e-mail account and a social one. Similarly, disposable identities can be convenient. When you sign up for something and you know your e-mail address will be sold many times, you might get a new e-mail address to use until the spam and other unsolicited e-mail are oppressive, and then you discard the address. These uses are called pseudonymity. These ways protect our privacy because we do not have to divulge what we consider sensitive data. But they also show we need a form of privacy protection that is unavailable.

The Swiss bank account was a classic example of a pseudonym. Each customer had only a number to access the account. Presumably anyone with that number could perform any transaction on the account. While such accounts were in use, Swiss banks had an outstanding reputation for maintaining the anonymity of the depositors. Some people register pseudonyms with e-mail providers so that they have anonymous drop boxes for e-mail. Others use pseudonyms in chat rooms or with online dating services.

14.3.6. Government and Privacy:

The government gathers and stores data on citizens, residents, and visitors. Government facilitates and regulates commerce and other kinds of personal activities such as healthcare, employment, education, and banking. In those roles the government is both an enabler or regulator of privacy and a user of private data. Government use of private data should be controlled. In this section we consider some of the implications of government access to private data.

Authentication:

Government plays a complex role in personal authentication. Many government agencies (such as the motor vehicles bureau) use identifiers to perform their work. Authentication documents (such as passports and insurance cards) often come from the government.

The government may also regulate the businesses that use identification and authentication keys. And sometimes the government obtains data based on those keys from others (for example, the U.S. government planned to buy credit reports from private companies to help with screening airline passenger lists for terrorists). In these multiple roles, the government may misuse data and violate privacy rights.

Data Access Risks:

Recognizing that there were risks in government access to personal data, the Secretary of Defense appointed a committee to investigate private data collection. The Technology and Privacy Advisory Committee, chaired by Newton Minow, former chair of the Federal Communications Commission, produced its report in 2004, they recognized risks when the government started to acquire data from other parties:

- *data errors*: ranging from transcription errors to incorrect analysis
- *inaccurate linking*: two or more correct data items but incorrectly linked on a presumed common element
- *difference of form and content*: precision, accuracy, format, and semantic errors
- *purposely wrong*: collected from a source that intentionally gives incorrect data, such as a forged identity card or a false address given to mislead
- *false positive*: an incorrect or out-of-date conclusion that the government does not have data to verify or reject, for example, delinquency in paying state taxes
- *mission creep*: data acquired for one purpose leading to a broader use because the data will support that mission
- *poorly protected*: data of questionable integrity because of the way it has been managed and handled

These risks apply to all branches of government, and most of them apply to private collection and use of data.

Steps to Protect Against Privacy Loss:

The committee recommended several steps the government can take to help safeguard private data.

- *Data minimization*: Obtain the least data necessary for the task. For example, if the goal is to study the spread of a disease, only the condition, date, and vague location (city or county) may suffice; the name or contact information of the patient may be unnecessary.
- *Data anonymization*: Where possible, replace identifying

information with untraceable codes (such as a record number); but make sure those codes cannot be linked to another database that reveals sensitive data.

- *Audit trail:* Record who has accessed data and when, both to help identify responsible parties in the event of a breach and to document the extent of damage.
- *Security and controlled access:* Adequately protect and control access to sensitive data.
- *Training:* Ensure people accessing data understand what to protect and how to do so.
- *Quality:* Consider the purpose for which data were collected, how they were stored, their age, and similar factors to determine the usefulness of the data.
- *Restricted usage:* Different from controlling access, review all proposed uses of the data to determine if those uses are consistent with the purpose for which the data were collected and the way they were handled (validated, stored, controlled).
- *Data left in place:* If possible, leave data in place with the original owner. This step helps guard against possible misuses of the data from expanded mission just because the data are available.
- *Policy:* Establish a clear policy for data privacy. Do not encourage violation of privacy policies.

These steps would help significantly to ensure protection of privacy.

14.3.7. Identity Theft:

As the name implies, identity theft is taking another person's identity. Use of another person's credit card is fraud; taking out a new credit card in that person's name is identity theft. Identity theft has risen as a problem from a relatively rare issue in the 1970s. In 2005, the U.S. Federal Trade Commission received over 250,000 complaints of identity theft. Most cases of identity theft become apparent in a month or two when fraudulent bills start coming in. By that time the thief has made a profit and has dropped this identity, moving on to a new victim. Having relatively few unique keys facilitates identity theft: A thief who gets one key can use that to get a second, and those two to get a third. Each key gives access to more data and resources. Few companies or agencies are set up to ask truly discriminating authentication questions (such as the grocery store at which you frequently shop or the city to which you recently bought an airplane ticket or third digit on line four of your last tax return). Because there are few authentication keys, we are often asked to give the same key (such as mother's maiden name) out too many people, some of whom might be part-time accomplices in identity theft.

14.4 AUTHENTICATION AND PRIVACY

In an earlier chapter we studied authentication, which we described as a means of proving or verifying a previously given identity. We also discussed various authentication technologies, which are subject to false accept (false positive) and false reject (false negative) limitations.

A social problem occurs when we confuse authentication with identification. We know that passwords are a poor discriminator. You would not expect all users of a system to have chosen different passwords. All we need is for the ID, password pair to be unique. On the other end of the spectrum, fingerprints and the blood vessel pattern in the retina of the eye are unique: given a fingerprint or retina pattern we expect to get but one identity that corresponds or to find no match in the database. That assumes we work with a good image. If the fingerprint is blurred or incomplete (not a complete contact or on a partly unsuitable surface), we might get several possible matches. If the possible matches are A, B, and C and the question is whether the print belongs to B, it is probably acceptable to allow the access on the grounds that the identity was among a small set of probable matches. Other authenticators are less sophisticated still. Hand geometry or the appearance of a face does not discriminate so well. Face recognition is highly dependent on the quality of the facial image: Evaluating a photograph of one person staring directly into a camera is very different from trying to work with one face in the picture of a crowd. Two different purposes are at work here, although the two are sometimes confused. For authentication we have an identity and some authentication data, and we ask if the authentication data match the pattern for the given identity. For identification, we have only the authentication data and we ask which identity corresponds to the authenticator. The second is a much harder question to answer than the first. For the first, we can say the pattern matches some percentage of the characteristics of our stored template, and based on the percentage, we declare a match or no match. For the second question, we do not know if the subject is even in the database. So even if we find several potential matches at various percentages, we do not know if there might be an even better match with a template not in our database.

14.4.1. What authentication means:

We use the term authentication to mean three different things: We authenticate an individual, identity, or attribute. An individual is a unique person. Authenticating an individual is what we do when we allow a person to enter a controlled room: We want only that human being to be allowed to enter. An identity is a character string or similar

descriptor, but it does not necessarily correspond to a single person, nor does each person have only one name. We authenticate an *identity* when we acknowledge that whoever (or whatever) is trying to log in as *admin* has presented an authenticator valid for that account. Similarly, authenticating an identity in a chat room as SuzyQ does not say anything about the person using that identifier: It might be a 16-year-old girl or a pair of middle-aged male police detectives, who at other times use the identity FrereJacques. Finally, we authenticate an *attribute* if we verify that a person has that attribute. An attribute is a characteristic. Here's an example of authenticating an attribute. Some places require one to be 21 or older in order to drink alcohol. A club's doorkeeper verifies a person's age and stamps the person's hand to show that the patron is over 21. Note that to decide, the doorkeeper may have looked at an identity card listing the person's birth date, so the doorkeeper knew the person's exact age to be 24 years, 6 months, 3 days, or the doorkeeper might be authorized to look at someone's face and decide if the person looks so far beyond 21 that there is no need to verify. The stamp authenticator signifies only that the person possesses the attribute of being 21 or over.

In computing applications, we frequently authenticate individuals, identities, and attributes. Privacy issues arise when we confuse these different authentications and what they mean. For example, the U.S. social security number was never intended to be an identifier, but now it often serves as an identifier, an authenticator, a database key, or all of these. When one data value serves two or more uses, a person acquiring it for one purpose can use it for another.

Relating an identity to a person is tricky. In an earlier chapter, we tell the story of rootkits, malicious software by which an unauthorized person can acquire supervisory control of a computer. Suppose the police arrest Ionut for chewing gum in public and seize his computer. By examining the computer, the police find evidence connecting that computer to an espionage case. The police show incriminating e-mail messages from Ionut on Ionut's computer and charge him. In his defense, Ionut points to a rootkit on his computer. He acknowledges that his computer may have been used in the espionage, but he denies that he was personally involved. The police have, he says, drawn an unjustifiable connection between Ionut's identity in the e-mail and Ionut the person. The rootkit is a plausible explanation for how some other person acted under the identity of Ionut. This example shows why we must carefully distinguish individual, identity, and attribute authentication.

Individual Authentication:

There are relatively few ways of identifying an individual. When we are born, for most of us our birth is registered at a

government records office, and we (probably our parents) receive a birth certificate. A few years later our parents enrol us in school, and they must present the birth certificate, which then may lead to receiving a school identity card. We submit the birth certificate and a photo to get a passport or a national identity card. We receive many other authentication numbers and cards throughout life. The whole process starts with a birth certificate issued to (the parents of) a baby, whose physical description (height, weight, even hair colour) will change significantly in just months. Birth certificates may contain the baby's fingerprints but matching a poorly taken fingerprint of a newborn baby to that of an adult is challenging at best. Fortunately, in most settings it is acceptable to settle for weak authentication for individuals: A friend who has known you since childhood, a schoolteacher, neighbours, and co-workers can support a claim of identity.

Identity Authentication:

We all use many different identities. When you buy something with a credit card, you do so under the identity of the credit card holder. You check into a hotel and get a magnetic stripe card instead of a key, and the door to your room authenticates you as a valid resident for the next three nights. If you think about your day, you will probably find 10 to 20 different ways some identity of you has been authenticated.

From a privacy standpoint, there may or may not be ways to connect all these different identities. A credit card links to the name and address of the card payer, who maybe you, your spouse, or anyone else willing to pay your expenses. Your auto toll device links to the name and perhaps address of whoever is paying the tolls: you, the car's owner, or an employer. When you make a telephone call, there is an authentication to the account holder of the telephone, and so forth.

Sometimes we do not want an action associated with an identity. For example, an anonymous tip or "whistle-blower's" telephone line is a means of providing anonymous tips of illegal or inappropriate activity. If you know your boss is cheating the company, confronting your boss might not be a good career-enhancing move. You probably don't even want there to be a record that would allow your boss to determine who reported the fraud. So, you report it anonymously. You might take the precaution of calling from a public phone so there would be no way to trace the person who called. In that case, you are purposely taking steps so that no common identifier could link you to the report.

Because of the accumulation of data, however, linking may be possible. As you leave your office to go to a public phone, there is a record of the badge you swiped at the door. A surveillance camera

shows you standing at the public phone. The record of the coffee shop has a timestamp showing when you bought your coffee (using your customer loyalty card) before returning to your office. The time of these details matches the time of the anonymous tip by telephone. In the abstract these data items do not stand out from millions of others. But someone probing a few minutes around the time of the tip can construct those links. In this example, linking would be done by hand.

Ever-improving technology permits more parallels like these to be drawn by computers from seemingly unrelated and uninteresting datapoints. Therefore, to preserve our privacy we may thwart attempts to link records. A friend gives a fictitious name when signing up for customer loyalty cards at stores. Another friend makes dinner reservations under a pseudonym. In one store they always ask for my telephone number when I buy something, even if I pay cash. Records clerks do not make the rules, so it is futile asking them why they need my number. If all they want is a number, I gladly give them one; it just doesn't happen to correspond to me.

Anonymized Records:

Part of privacy is linkages: Some person is named Erin; some person has the medical condition diabetes; neither of those facts is sensitive. The linkage that Erin has diabetes becomes sensitive. Medical researchers want to study populations to determine incidence of diseases, common factors, trends, and patterns. To preserve privacy, researchers often deal with anonymized records, records from which identifying information has been removed. If those records can be reconnected to the identifying information, privacy suffers. If, for example, names have been removed from records, but telephone numbers remain, a researcher can use a different database of telephone numbers to determine the patient, or at least the name assigned to the telephone. Removing enough information to prevent identification is difficult and can also limit the research possibilities.

14.5 SUMMARY

- To summarize, some points about privacy:
 - Privacy is controlled disclosure: The subject chooses what personal data to give out and to whom.
 - After disclosing something, a subject relinquishes much control to the receiver.
 - What data are sensitive is at the discretion of the subject; people consider different things sensitive.
 - Why a person considers something sensitive is less important

than that it is.

- Individuals, informal groups, and formal organizations all have things they consider private.
- Privacy has a cost; choosing not to give out certain data may limit other benefits.
- The first step in establishing privacy is the same as the other areas of computer security: We must first define a privacy *policy* that documents what privacy we require. The early work by Ware's committee laid out very important fundamental principles of information privacy.
- Identification and authentication are two different activities that are easy to confuse. Part of the confusion arises because people do not clearly distinguish the underlying concepts. The confusion is also the result of using one data item for more than one purpose.
- Authentication depends on something that confirms a property. In life few sound authenticators exist, so we tend to overuse those we do have: an identification number, birth date, or family name. But, as we described, those authenticators are also used as database keys, with negative consequences to privacy.
- We have also studied cases in which we do not want to be identified. Anonymity and pseudonymity are useful in certain contexts. But data collection and correlation, on a scale made possible only with computers, can defeat anonymity and pseudonymity. As we computer professionals introduce new computer capabilities, we need to encourage a public debate on the related privacy issues.

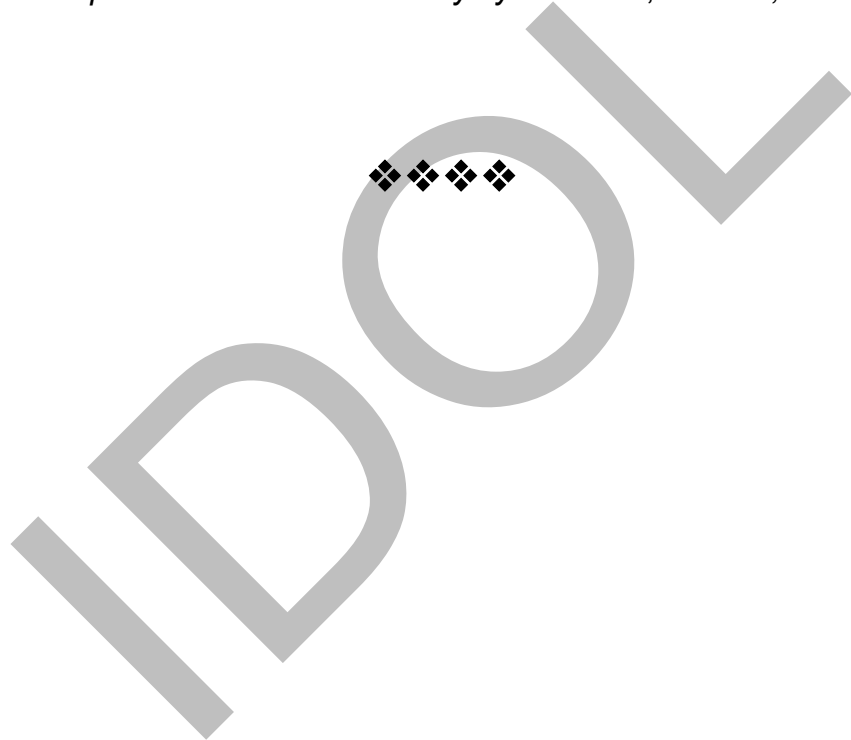
14.6 REVIEW QUESTIONS

- a) What are the aspects of information privacy?
- b) Write a short note on computer related privacy problems.
- c) Explain fair information policies in detail.
- d) What are the controls placed on commercial web sites? Explain with suitable examples.
- e) Explain anonymity and pseudonymity.
- f) What are data access risks?
- g) What are the steps to protect against privacy loss?
- h) What does authentication mean?
- i) Explain individual authentication and identity authentication.

14.7 BIBLIOGRAPHY, REFERENCES AND FURTHER

READING

- *Security in Computing* by C. P. Pfleeger, and S. L. Pfleeger, Pearson Education.
- *Computer Security: Art and Science* by Matt Bishop, Pearson Education.
- *Cryptography and Network Security: Principles and practice* by Stallings
- *Network Security* by Kaufman, Perlman, Speciner
- *Network Security : A Beginner's Guide* by Eric Maiwald, TMH
- *Java Network Security* by Macro Pistoia, Pearson Education
- *Principles of information security* by Whitman, Mattord, Thomson



LEGAL AND ETHICAL ISSUES IN COMPUTER SECURITY

Unit Structure:

- 15.0 Objectives
- 15.1 Introduction
- 15.2 Protecting Programs and Data
 - 15.2.1. Copyrights
 - 15.2.2. Patents
 - 15.2.3. Trade Secrets
 - 15.2.4. Protection for Computer Objects
- 15.3 Information and the Law
 - 15.3.1. Information as an Object
 - 15.3.2. Legal Issues Relating to Information
 - 15.3.3. Protecting Information
- 15.4 Rights of Employees and Employers
 - 15.4.1. Ownership of Products
- 15.5 Redress for Software Failures
 - 15.5.1. Selling Correct Software
 - 15.5.2. Reporting Software Flaws
- 15.6 Computer Crime
 - 15.6.1. Why a Separate Category for Computer Crime is needed?
 - 15.6.2. Why Computer Crime is hard to define
 - 15.6.3. Why Computer Crime is hard to prosecute
 - 15.6.4. Why Computer Criminals are hard to catch
 - 15.6.5. What Computer Crime does not address
- 15.7 Ethical Issues in Computer Society
 - 15.7.1. Differences between the Law and Ethics
 - 15.7.2. Studying Ethics
 - 15.7.3. Ethical Reasoning
- 15.8 Summary
- 15.9 Review Questions
- 15.10 Bibliography, References and Further Reading

15.0 OBJECTIVES

At the end of this chapter, you will understand

- Program and data protection by patents, copyrights, and trademarks
- Computer crime
- Ethical analysis of computer security situations
- Codes of professional ethics

15.1 INTRODUCTION

In this chapter we study human controls applicable to computer security: the legal system and ethics. The legal system has adapted quite well to computer technology by reusing some old forms of legal protection (copyrights and patents) and creating laws where no adequate ones existed (malicious access). Still, the courts are not a perfect form of protection for computer resources, for two reasons. First, the courts tend to be reactive instead of proactive. That is, we must wait for a transgression to occur and then adjudicate it, rather than try to prevent it in the first place. Second, fixing a problem through the courts can be time consuming (sometimes taking years) and expensive; the latter characteristic prevents all but the wealthy from addressing most security issues.

15.2 PROTECTING PROGRAMS AND DATA

Copyrights, patents, and trade secrets are legal devices that can protect computers, programs, and data. However, in some cases, precise steps must be taken to protect the work before anyone else is allowed access to it. In this section, we explain how each of these forms of protection was originally designed to be used and how each is currently used in computing. We focus primarily on U.S. law, to provide examples of intent and consequence.

15.2.1. Copyrights:

In the United States, the basis of copyright protection is presented in the U.S. Constitution. The body of legislation supporting constitutional provisions contains laws that elaborate on or expand the constitutional protections. Relevant statutes include the U.S. copyright law of 1978, which was updated in 1998 as the Digital Millennium Copyright Act (DMCA) specifically to deal with computers and other electronic media such as digital video and music. The 1998 changes brought U.S. copyright law into general conformance with the World Intellectual Property Organization treaty of 1996, an international copyright standard to which 95 countries adhere.

Copyrights are designed to protect the expression of ideas. Thus, a copyright applies to a creative work, such as a story,

photograph, song, or pencil sketch. The right to copy an *expression* of an idea is protected by a copyright. Ideas themselves, the law alleges, are free; anyone with a bright mind can think up anything anyone else can, at least in theory. The intention of a copyright is to allow regular and free exchange of ideas. The law protects an individual's right to earn a living, while recognizing that exchanging ideas supports the intellectual growth of society. The copyright says that a way of expressing an idea belongs to the author. For example, in music, there may be two or three copyrights related to a single creation: A composer can copyright a song, an arranger can copyright an arrangement of that song, and an artist can copyright a specific performance of that arrangement of that song. The price you pay for a ticket to a concert includes compensation for all three creative expressions. Copyright gives the author the *exclusive* right to make copies of the expression and sell them to the public. That is, only the author (or booksellers or others working as the author's agents) can sell copies of the author's book.

Definition of Intellectual Property:

The U.S. copyright law (§102) states that a copyright can be registered for “original works of authorship fixed in any tangible medium of expression, ... from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device”. Only the originator of the expression is entitled to copyright; if an expression has no determinable originator, copyright cannot be granted. Certain works are in the public domain, owned by the public, by no one in particular. Works of the U.S. government and many other governments are in the public domain and therefore not subject to copyright. Finally, copyright lasts for only a limited period, so certain very old works, such as the plays of Shakespeare, are in the public domain, their possibility of copyright having expired. The copyrighted expression must also be in some tangible medium. A story or artwork must be written, printed, painted, recorded (on a physical medium such as a plastic record), stored on a magnetic medium (such as a disk or tape), or fixed in some other way. Furthermore, the purpose of the copyright is to promote distribution of the work; therefore, the work must be distributed, even if a fee is charged for a copy.

Originality of Work:

The work being copyrighted must be original to the author. As noted previously, some expressions in the public domain are not subject to copyright. A work can be copyrighted even if it contains some public domain material, if there is some originality, too. The author does not even have to identify what is public and what is original. For example, a music historian could copyright a collection of folksongs even if some are in the public domain. To be subject to copyright, something in or *about* the collection must be original. The historian might argue that collecting the songs, selecting which ones

to include, and putting them in order was the original part. In this case, the copyright law would not protect the folksongs (which would be in the public domain) but would instead protect that specific selection and organization.

Fair Use of Material:

The copyright law indicates that the copyrighted object is subject to fair use. A purchaser has the right to use the product in the manner for which it was intended and in a way that does not interfere with the author's rights. Specifically, the law allows "fair use of a copyrighted work, including such use by reproduction in copies... for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship or research." The purpose and effect of the use on the potential market for or value of the work affect the decision of what constitutes fair use. For example, fair use allows making a backup copy of copyrighted software you acquired legally: Your backup copy protects your use against system failures, but it doesn't affect the author because you have no need for nor do you want use of two copies at once. The copyright law usually upholds the author's right to a fair return for the work, while encouraging others to use the underlying ideas. Unfair use of a copyrighted item is called piracy.

The copyright law also has the concept of a first sale: after having bought a copyrighted object, the new owner can give away or resell the object. That is, the copyright owner is entitled to control the first sale of the object. This concept works fine for books: An author is compensated when a bookstore sells a book, but the author earns no additional revenue if the book is later resold at a second-hand store.

Requirements for Registering a Copyright:

The copyright is easy to obtain, and mistakes in securing a copyright can be corrected. The first step of registration is notice. Any potential user must be made aware that the work is copyrighted. Each copy must be marked with the copyright symbol ©, the word *Copyright*, the year, and the author's name. Each copy distributed must be so marked, although the law will forgive failure to mark copies if a reasonable attempt is made to recall and mark anyone's distributed without a mark. The copyright must also be officially filed. In the United States a form is completed and submitted to the Copyright Office, along with a nominal fee and a copy of the work. A U.S. copyright now lasts for 70 years beyond the death of the last surviving author or, if the item was copyrighted by a company or organization, for 95 years after the date of publication. The international standard is 50 years after the death of the last author or 50 years from publication.

Copyright Infringement:

[Unedited Version: Information Security](#)

The holder of the copyright must go to court to prove that someone has infringed on the copyright. The infringement must be substantial, and it must be copying, not independent work. In theory, two people might write identically the same song independently, neither knowing the other. These two people would *both* be entitled to copyright protection for their work. Neither would have infringed on the other, and both would have the right to distribute their work for a fee. Again, copyright is most easily understood for written works of fiction because it is extremely unlikely that two people would express an idea with the same or similar wording. However, it is far less likely that two textbook authors would have the same pattern of presentation and the same examples from beginning to end.

Copyrights for Computer Software:

The original copyright law envisioned protection for things such as books, songs, and photographs. People can rather easily detect when these items are copied. The separation between public domain and creativity is clear. And the distinction between an idea (feeling, emotion) and its expression is obvious. Works of nonfiction understandably have less leeway for independent expression. Because of programming language constraints and speed and size efficiency, computer programs have less leeway still. Can a computer program be copyrighted? Yes. The 1976 copyright law was amended in 1980 to include an explicit definition of computer software. However, copyright protection may not be an especially desirable form of protection for computer works.

To see why, consider the algorithm used in each program. The algorithm is the idea, and the statements of the programming language are the expression of the idea. Therefore, protection is allowed for the program statements themselves, but not for the algorithmic concept: copying the code intact is prohibited, but reimplementing the algorithm is permitted. Remember that one purpose of copyright is to promote the dissemination of ideas. The algorithm, which is the idea embodied in the computer program, is to be shared. A second problem with copyright protection for computer works is the requirement that the work be published. A program may be published by distribution of copies of its object code, for example, on a disk. However, if the source code is not distributed, it has not been published. An alleged infringer cannot have violated a copyright on source code if the source code was never published.

Copyrights for Digital Objects:

The Digital Millennium Copyright Act (DMCA) of 1998 clarified some issues of digital objects (such as music files, graphics images, data in a database, and computer programs), but it left others unclear. Among the provisions of the DMCA are these:

- Digital objects *can be* subject to copyright.
- It is a crime to circumvent or disable antipiracy functionality built into an object.
- It is a crime to manufacture, sell, or distribute devices that disable antipiracy functionality or that copy digital objects.
- However, these devices can be used (and manufactured, sold, or distributed) for research and educational purposes.
- It is acceptable to make a backup copy of a digital object as a protection against hardware or software failure or to store copies in an archive.
- Libraries can make up to three copies of a digital object for lending to other libraries.

So, a user can make reasonable copies of an object in the normal course of its use and as a protection against system failures. If a system is regularly backed up and so a digital object (such as a software program) is copied onto many backups, that is not a violation of copyright.

The uncertainty comes in deciding what is a device to counter piracy. A disassembler or decompiler could support piracy or could be used to study and enhance a program. Someone who decompiles an executable program, studies it to infer its method, and then modifies, compiles, and sells the result is misusing the decompiler. But the distinction is hard to enforce, in part because the usage depends on intent and context. Reaction to the Digital Millennium Copyright Act has not been uniformly favourable. Some say it limits computer security research. Worse, others point out it can be used to prevent exactly the free interchange of ideas that copyright was intended to promote. In 2001 a Princeton University professor, Edward Felten, and students presented a paper on cryptanalysis of the digital watermarking techniques used to protect digital music files from being copied. They had been pressured not to present in the preceding April by music industry groups who threatened legal action under the DMCA. Digital objects are more problematic than paper ones because they can be copied exactly. Unlike fifth-generation photocopies, each digital copy of a digital object can be identical to the original. An emerging principle is that software, like music, is acquired in a style more like rental than purchase. You purchase not a piece of software, but the right to use it. Clarifying this position, the U.S. No Electronic Theft (NET) Act of 1997 makes it a criminal offense to reproduce or distribute copyrighted works, such as software or digital recordings, even without charge. The area of copyright protection applied to computer works continues to evolve and is subject to much interpretation by the courts. Therefore, it is not certain what aspects of a computer work are subject to copyright. Although copyright protection can be applied to computer works, the

copyright concept was conceived before the electronic age, and thus the protection may be less than what we desire. Copyrights do not address all the critical computing system elements that require protection.

15.2.2. Patents:

Patents are unlike copyrights in that they protect inventions, tangible objects, or ways to make them, not works of the mind. The distinction between patents and copyrights is that patents were intended to apply to the results of science, technology, and engineering, whereas copyrights were meant to cover works in the arts, literature, and written scholarship. A patent can protect a “new and useful process, machine, manufacture, or composition of matter”. The U.S. law excludes “newly discovered laws of nature... [and] mental processes”. A patent is designed to protect the device or process for *carrying out* an idea, not the idea itself.

Requirement of Novelty:

If two composers happen to compose the same song independently at different times, copyright law would allow both to have copyright. If two inventors devise the same invention, the patent goes to the person who invented it first, regardless of who first filed the patent. A patent can be valid only for something that is truly novel or unique, so there can be only one patent for a given invention. An object patented must also be nonobvious. If an invention would be obvious to a person ordinarily skilled in the field, it cannot be patented. The law states that a patent *cannot* be obtained “if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains”.

Procedure for Registering a Patent:

One registers a copyright by filing a brief form, marking a copyright notice on the creative work, and distributing the work. The whole process takes less than an hour.

To obtain a patent, an inventor must convince the U.S. Patent and Trademark Office that the invention deserves a patent. For a fee, a patent attorney will research the patents already issued for similar inventions. This search accomplishes two things. First, it determines that the invention to be patented has not already been patented (and, presumably, has not been previously invented). Second, the search can help identify similar things that have been patented. These similarities can be useful when describing the unique features of the invention that make it worthy of patent protection. The Patent Office compares an application to those of all other similar patented inventions and decides whether the application covers something

truly novel and nonobvious. If the office decides the invention is novel, a patent is granted.

Typically, an inventor writes a patent application listing many claims of originality, from very general to very specific. The Patent Office may disallow some of the more general claims while upholding some of the more specific ones. The patent is valid for all the upheld claims. The patent applicant reveals what is novel about the invention in enough detail to allow the Patent Office and the courts to judge novelty; that degree of detail may also tell the world how the invention works, thereby opening the possibility of infringement. The patent owner uses the patented invention by producing products or by licensing others to produce them. Patented objects are sometimes marked with a patent number to warn others that the technology is patented. The patent holder hopes this warning will prevent others from infringing.

Patent Infringement:

A patent holder *must* oppose all infringement. With a copyright, the holder can choose which cases to prosecute, ignoring small infringements and waiting for serious infractions where the infringement is great enough to ensure success in court or to justify the cost of the court case. However, failing to sue a patent infringement even a small one or one the patent holder does not know about can mean losing the patent rights entirely. But, unlike copyright infringement, a patent holder does not have to prove that the infringer copied the invention; a patent infringement occurs even if someone independently invents the same thing, without knowledge of the patented invention. Every infringement must be prosecuted. Prosecution is expensive and time consuming, but even worse, suing for patent infringement could cause the patent *holder* to lose the patent. Someone charged with infringement can argue all the following points as a defence against the charge of infringement.

- *This isn't infringement:* The alleged infringer will claim that the two inventions are sufficiently different that no infringement occurred.
- *The patent is invalid:* If a prior infringement was not opposed, the patent rights may no longer be valid.
- *The invention is not novel:* In this case, the supposed infringer will try to persuade the judge that the Patent Office acted incorrectly in granting a patent and that the invention is nothing worthy of patent.
- *The infringer invented the object first:* If so, the accused infringer, and not the original patent holder, is entitled to the patent.

The first defence does not damage a patent, although it can limit the novelty of the invention. However, the other three defences

can destroy patent rights. Worse, all four defences can be used every time a patent holder sues someone for infringement. Finally, obtaining and defending a patent can incur substantial legal fees. Patent protection is most appropriate for large companies with substantial research and development (and legal) staffs.

Applicability of Patents to Computer Objects:

The Patent Office has not encouraged patents of computer software. For a long time, computer programs were the representation of an algorithm, and an algorithm was a fact of nature, which is not subject to patent. An early software patent case, *Gottschalk v. Benson*, involved a request to patent a process for converting decimal numbers into binary. The Supreme Court rejected the claim, saying it seemed to attempt to patent an abstract idea, in short, an algorithm. But the underlying algorithm is precisely what most software developers would like to protect.

In 1981, two cases (*Diamond v. Bradley* and *Diamond v. Diehr*) won patents for a process that used computer software, a well-known algorithm, temperature sensors, and a computer to calculate the time to cure rubber seals. The court upheld the right to a patent because the claim was not for the software or the algorithm alone, but for the process that happened to use the software as one of its steps. An unfortunate inference is that using the software without using the other patented steps of the process would not be infringement.

Since 1981 the patent law has expanded to include computer software, recognizing that algorithms, like processes and formulas, are inventions. The Patent Office has issued thousands of software patents since these cases. But because of the time and expense involved in obtaining and maintaining a patent, this form of protection may be unacceptable for a small-scale software writer.

15.2.3. Trade Secrets:

A trade secret is unlike a patent or copyright in that it must be kept a *secret*. The information has value only as a secret, and an infringer is one who divulges the secret. Once divulged, the information usually cannot be made secret again.

Characteristics of Trade Secrets:

A trade secret is information that gives one company a competitive edge over others. For example, the formula for a soft drink is a trade secret, as is a mailing list of customers or information about a product due to be announced in a few months. The distinguishing characteristic of a trade secret is that it must always

be kept secret. Employees and outsiders who have access to the secret must be required not to divulge the secret. The owner must take precautions to protect the secret, such as storing it in a safe, encrypting it in a computer file, or making employees sign a statement that they will not disclose the secret.

If someone obtains a trade secret improperly and profits from it, the owner can recover profits, damages, lost revenues, and legal costs. The court will do whatever it can to return the holder to the same competitive position it had while the information was secret and may award damages to compensate for lost sales. However, trade secret protection evaporates in case of independent discovery. If someone else happens to discover the secret independently, there is no infringement and trade secret rights are gone.

Reverse Engineering:

Another way trade secret protection can vanish is by reverse engineering. Suppose a secret is the way to pack tissues in a cardboard box to make one pop up as another is pulled out. Anyone can cut open the box and study the process. Therefore, the trade secret is easily discovered. In reverse engineering, one studies a finished object to determine how it is manufactured or how it works. Through reverse engineering someone might discover how a telephone is built; the design of the telephone is obvious from the components and how they are connected. Therefore, a patent is the appropriate way to protect an invention such as a telephone. However, something like a soft drink is not just the combination of its ingredients. Making a soft drink may involve time, temperature, presence of oxygen or other gases, and similar factors that could not be learned from a straight chemical decomposition of the product. The recipe of a soft drink is a closely guarded trade secret. Trade secret protection works best when the secret is not apparent in the product.

Applicability to Computer Objects:

Trade secret protection applies very well to computer software. The underlying algorithm of a computer program is novel, but its novelty depends on nobody else's knowing it. Trade secret protection allows distribution of the *result* of a secret (the executable program) while keeping the program design hidden. Trade secret protection does not cover copying a product (specifically a computer program), so it cannot protect against a pirate who sells copies of someone else's program without permission. However, trade secret protection makes it illegal to steal a secret algorithm and use it in

another product. The difficulty with computer programs is that reverse engineering works. Decompiler and disassembler programs can produce a source version of an executable program. Of course, this source does not contain the descriptive variable names or the comments to explain the code, but it is an accurate version that someone else can study, reuse, or extend.

Difficulty of Enforcement:

Trade secret protection is of no help when someone infers a program's design by studying its output or, worse yet, decoding the object code. Both are legitimate (that is, legal) activities, and both cause trade secret protection to disappear. The confidentiality of a trade secret must be ensured with adequate safeguards. If source code is distributed loosely or if the owner fails to impress on people (such as employees) the importance of keeping the secret, any prosecution of infringement will be weakened. Employment contracts typically include a clause stating that the employee will not divulge any trade secrets received from the company, even after leaving a job. Additional protection, such as marking copies of sensitive documents or controlling access to computer files of secret information, may be necessary to impress people with the importance of secrecy.

15.2.4. Protection for Computer Objects:

The previous sections have described three forms of protection: the copyright, patent, and trade secret laws. Each of these provides a different form of protection to sensitive things. In this section we consider different kinds of computer objects and describe which forms of protection are most appropriate for each kind. Table 15-1 shows how these three forms of protection compare in several significant ways.

Table 15-1 Comparing Copyright, Patent, and Trade Secret Protection

	Copyright	Patent	Trade Secret
Protects	Expression of idea, not idea itself	Invention—the way something works	A secret, competitive advantage
Protected object made public	Yes; intention is to promote publication	Design filed at Patent Office	No
Requirement to distribute	Yes	No	No
Ease of filing	Very easy, do-it-yourself	Very complicated; specialist lawyer suggested	No filing
Duration	Varies by country; approximately 75–100 years is typical	19 years	Indefinite
Legal protection	Sue if unauthorized copy sold	Sue if invention copied	Sue if secret improperly obtained

Computer artefacts are new and constantly changing, and they are not yet fully appreciated by the legal system based on centuries of precedent. Perhaps in a few years the issue of what protection is most appropriate for a given computer object will be more clear-cut. Possibly a new form of protection or a new use of an old form will apply specifically to computer objects. Until the law provides protection that truly fits computer goods, here are some guidelines for using the law to protect computer objects.

Protecting Hardware:

Hardware, such as chips, disk drives, or floppy disk media, can be patented. The medium itself can be patented, and someone who invents a new process for manufacturing it can obtain a second patent.

Protecting Firmware:

The situation is a little less clear about microcode. Certainly, the physical devices on which microcode is stored can be patented. Also, a special-purpose chip that can do only one specific task can probably be patented. However, the data (instructions, algorithms, microcode, programs) contained in the devices are probably not patentable. Can they be copyrighted? Are these the expression of an idea in a form that promotes dissemination of the idea? Probably not. And if these devices were copyrighted, what would be the definition of a copy that infringed on the copyright? Worse, would the manufacturer really want to register a copy of the internal algorithm with the Copyright Office? Copyright protection is probably inappropriate for computer firmware.

Trade secret protection seems appropriate for the code embedded in a chip. Given enough time, we can reverse-engineer and infer the code from the behaviour of the chip. The behaviour of the chip does not reveal what algorithm is used to produce that behaviour. The original algorithm may have better (or worse) performance (speed, size, fault tolerance) that would not be obvious from reverse engineering. The courts have affirmed that computer software *is* an appropriate subject for copyright protection and that protection should be no less valid when the software is in a chip rather than in a conventional program.

Protecting Object Code Software:

Object code is usually copied so that it can be distributed for profit. The code is a work of creativity, and most people agree that object code distribution is an acceptable medium of publication. Thus, copyright protection seems appropriate. A copyright application is usually accompanied by a copy of the object being protected. With a book or piece of music (printed or recorded), it is easy to provide a copy. The Copyright Office has not yet decided what is an appropriate medium in which to accept object code. A binary listing of the object code will be taken, but the Copyright Office does so without acknowledging the listing to be acceptable or enough. The Office will accept a source code listing. Some people argue that a source code listing is not equivalent to an object code listing, in the same way that a French translation of a novel is different from its original language version. It is not clear *in the courts* that registering a source code version provides copyright protection to object code. However, someone should not be able to take the object code of a system, rearrange the order of the individual routines, and say that the result is a new system. Without the original source listings, it would be very difficult to compare two binary files and determine that one was the functional equivalent of the other simply through rearrangement.

Protecting Source Code Software:

Software developers selling to the mass market are reticent to distribute their source code. The code can be treated as a trade secret, although some lawyers also encourage that it be copyrighted. A copyright protects the right to distribute copies of the *expression* of an idea, not the idea itself. Therefore, a copyright does not prevent someone from reimplementing an algorithm, expressed through a copyrighted computer program. As just described, source code may be the most appropriate form in which to register a copyright for a program distributed in object form. It is difficult to register source code with the Copyright Office while still ensuring its secrecy.

Protecting Documentation:

Unedited Version: [Information Security](#)

If we think of documentation as a written work of nonfiction (or, perhaps, fiction), copyright protection is effective and appropriate for it. Notice that the documentation is distinct from the program. A program and its documentation must be copyrighted separately. Furthermore, copyright protection of the documentation may win a judgment against someone who illegally copies both a program and its documentation. In cases where a written law is unclear or is not obviously applicable to a situation, the results of court cases serve to clarify or even extend the words of the law. As more unfair acts involving computer works are perpetrated, lawyers will argue for expanded interpretations of the law. Thus, the meaning and use of the law will continue to evolve through judges' rulings. In a sense, computer technology has advanced much faster than the law has been able to.

Protecting Web Content:

Content on the web is media, much the same as a book or photograph, so the most appropriate protection for it is copyright. This copyright would also protect software you write to animate or otherwise affect the display of your web page. And, in theory, if your web page contains malicious code, your copyright covers that, too. As we discussed earlier, a copyrighted work does not have to be exclusively new; it can be a mixture of new work to which you claim copyright and old things to which you do not. You may purchase or use with permission a piece of web art, a widget (such as an applet that shows a spinning globe), or some music. Copyright protects your original works. Protecting Domain Names and URLs Domain names, URLs, company names, product names, and commercial symbols are protected by a trademark, which gives exclusive rights of use to the owner of such identifying marks.

15.3 INFORMATION AND THE LAW

Source code, object code, and even the "look and feel" of a computer screen are recognizable, if not tangible, objects. The law deals reasonably well, although somewhat belatedly, with these things. But computing is in transition to a new class of object, with new legal protection requirements. Electronic commerce, electronic publishing, electronic voting, electronic banking, these are the new challenges to the legal system. In this section we consider some of these new security requirements.

15.3.1. Information as an Object:

The shopkeeper used to stock "things" in the store, such as buttons, automobiles, and pounds of sugar. The buyers were

customers. When a thing was sold to a customer, the shopkeeper's stock of that thing was reduced by one, and the customer paid for and left with a thing. Sometimes the customer could resell the thing to someone else, for more or less than the customer originally paid. Other kinds of shops provided services that could be identified as things, for example, a haircut, root canal, or defence for a trial. The value of a service in a free economy was somehow related to its desirability to the buyer and the seller. But today we must consider a third category for sale: information. No one would argue against the proposition that information is valuable. Students are tempted to pay others for answers during examinations, and businesses pay for credit reports, client lists, and marketing advice. But information does not fit the familiar commercial paradigms with which we have dealt for many years. Let us examine why information is different from other commercial things.

Information Is Not Depletable:

Unlike tangible things and services, information can be sold again and again without depleting stock or diminishing quality. For example, a credit bureau can sell the same credit report on an individual to an unlimited number of requesting clients. Each client pays for the information in the report. The report may be delivered on some tangible medium, such as paper, but it is the *information*, not the medium, that has the value.

Information Can Be Replicated:

The value of information is what the buyer will pay the seller. But after having bought the information, the buyer can then become a seller and can potentially deprive the original seller of further sales. Because information is not depletable, the buyer can enjoy or use the information and can also sell it many times over, perhaps even making a profit.

Information Has a Minimal Marginal Cost:

The marginal cost of an item is the cost to produce another one after having produced some already. If a newspaper sold only one copy on a particular day, that one issue would be prohibitively expensive because it would have to cover the day's cost (salary and benefits) of all the writers, editors, and production staff, as well as a share of the cost of all equipment for its production. These are fixed costs needed to produce a first copy. With this model, the cost of the second and subsequent copies is minuscule, representing basically just the cost of paper and ink to print them. Fortunately, newspapers have very large press runs and daily sales, so the fixed costs are spread evenly across many copies printed. The cost of information similarly depends on fixed costs plus costs to reproduce. Typically,

the fixed costs are large whereas the cost to reproduce is extremely small, even less than for a newspaper because there is no cost for the raw materials of paper and ink. However, unlike a newspaper, information is far more feasible for a buyer to resell. A copy of digital information can be perfect, indistinguishable from the original, the same being true for copies of copies of copies of copies.

The Value of Information Is Often Time Dependent:

If you knew for certain what the trading price of a share of Microsoft stock would be next week, that information would be extremely valuable because you could make an enormous profit on the stock market. Of course, that price cannot be known today. But suppose you knew that Microsoft was certain to announce something next week that would cause the price to rise or fall. That information would be almost as valuable as knowing the exact price, and it could be known in advance. However, knowing *yesterday's* price for Microsoft stock or knowing that *yesterday* Microsoft announced something that caused the stock price to plummet is almost worthless because it is printed in every major financial newspaper. Thus, the value of information may depend on when you know it.

Information Is Often Transferred Intangibly:

A newspaper is a printed artefact. The news agent hands it to a customer, who walks away with it. Both the seller and the buyer realize and acknowledge that something has been acquired. Furthermore, it is evident if the newspaper is seriously damaged; if a serious production flaw appears in the middle, the defect is easy to point out. But times are changing. Increasingly, information is being delivered as bits across a network instead of being printed on paper. If the bits are visibly flawed (that is, if an error detecting code indicates a transmission error), demonstrating that flaw is easy. However, if the copy of the information is accurate but the underlying information is incorrect, useless, or not as expected, it is difficult to justify a claim that the information is flawed.

15.3.2. Legal Issues Relating to Information:

These characteristics of information significantly affect its legal treatment. If we want to understand how information relates to copyright, patent, and trademark laws, we must understand these attributes. We can note first that information has some, limited legal basis for the protection. For example, information can be related to trade secrets, in that information is the stock in trade of the information seller. While the seller has the information, trade secret protection applies naturally to the seller's legitimate ability to profit from information. Thus, the courts recognize that information has value. Other forms of protection are offered by copyrights and

patents. As we have seen earlier, neither of these applies perfectly to computer hardware or software, and they apply even less well to information. The pace of change in the legal system is slow, helping to ensure that the changes that do occur are fair and well considered. The deliberate pace of change in the legal system is about to be hit by the supersonic rate of change in the information technology industry. Laws do not, and cannot, control all cyber threats. Let us look at several examples of situations in which information needs are about to place significant demands on the legal system.

Information Commerce:

Information is unlike most other goods traded, even though it has value and is the basis of some forms of commerce. The market for information is still young, and so far, the legal community has experienced few problems. Nevertheless, several key issues must be resolved. For example, we have seen that software piracy involves copying information without offering adequate payment to those who deserve to be paid. Several approaches have been tried to ensure that the software developer or publisher receives just compensation for use of the software: copy protection, freeware, and controlled distribution. More recently, software is being delivered as mobile code or applets, supplied electronically as needed. The applet approach gives the author and distributor more control. Each applet can potentially be tracked and charged for, and each applet can destroy itself after use so that nothing remains to be passed for free to someone else. But this scheme requires a great deal of accounting and tracking, increasing the costs of what might otherwise be reasonably priced.

Electronic Publishing:

Many newspapers and magazines post a version of their content on the Internet, as do wire services and television news organizations. For example, the British Broadcasting Company (BBC) and the Reuters news services have a significant web presence. We should expect that some news and information will eventually be published and distributed exclusively on the Internet. Indeed, encyclopaedias such as the Britannica and Expedia are mainly web-based services now, rather than being delivered as the large number of book volumes they used to occupy. Here again the publisher has a problem ensuring that it receives fair compensation for the work. Cryptography-based technical solutions are under development to address this problem. However, these technical solutions must be supported by a legal structure to enforce their use.

Protecting Data in a Database:

Databases are a form of software that has posed significant

Unedited Version: Information Security

problems for legal interpretation. The courts have had difficulty deciding which protection laws apply to databases. How does one determine that a set of data came from a database (so that the database owner can claim some compensation)? Who even owns the data in a database if it is public data, such as names and addresses?

Electronic Commerce:

Laws related to trade in goods have evolved literally over centuries. Adequate legal protections exist to cover defective goods, fraudulent payment, and failure to deliver when the goods are tangible and are bought through traditional outlets such as stores and catalogues. However, the situation becomes less clear when the goods are traded electronically. If you order goods electronically, digital signatures and other cryptographic protocols can provide a technical protection for your "money." However, suppose the information you order is not suitable for use or never arrives or arrives damaged or arrives too late to use. How do you prove conditions of the delivery? For catalogue sales, you often have receipts or some paper form of acknowledgment of time, date, and location. But for digital sales, such verification may not exist or can be easily modified. These legal issues must be resolved as we move into an age of electronic commerce.

15.3.3. Protecting Information:

Clearly, current laws are inadequate for protecting the information itself and for protecting electronically based forms of commerce. So how is information to be protected legally? As described, copyrights, patents, and trade secrets cover some, but not all, issues related to information. Nevertheless, the legal system does not allow free traffic in information; some mechanisms can be useful.

Criminal and Civil Law:

Statutes are laws that state explicitly that certain actions are illegal. A statute is the result of a legislative process by which a governing body declares that the new law will be in force after a designated time. Often, a violation of a statute will result in a criminal trial, in which the government argues for punishment because an illegal act has harmed the desired nature of society. The goal of a criminal case is to punish the criminal, usually by depriving him or her of rights in some way (such as putting the criminal in prison or assessing a fine).

Civil law is a different type of law, not requiring such a high standard of proof of guilt. In a civil case, an individual, organization,

company, or group claims it has been harmed. The goal of a civil case is restitution: to make the victim "whole" again by repairing the harm.

Tort Law:

Special legal language describes the wrongs treated in a civil case. The language reflects whether a case is based on breaking a law or on violating precedents of behaviour that have evolved over time. In other words, sometimes judges may make determinations based on what is reasonable and what has come before, rather than on what is written in legislation. A tort is harm not occurring from violation of a statute or from breach of a contract but instead from being counter to the accumulated body of precedents. Thus, statute law is written by legislators and is interpreted by the courts; tort law is unwritten but evolves through court decisions that become precedents for cases that follow. The basic test of a tort is what a reasonable person would do. Computer information is perfectly suited to tort law. The court merely must decide what is reasonable behaviour, not whether a statute covers the activity. Because tort law is written only as a series of court decisions that evolve constantly, prosecution of a tort case can be difficult. If you are involved in a case based on tort law, you and your lawyer are likely to try two approaches: First, you might argue that your case is a clear violation of the norms of society, that it is not what a fair, prudent person would do. This approach could establish a new tort. Second, you might argue that your case is like one or more precedents, perhaps drawing a parallel between a computer program and a work of art. The judge or jury would have to decide whether the comparison was apt. In both ways, law can evolve to cover new objects.

Contract Law:

A third form of protection for computer objects is contracts. A contract is an agreement between two parties. A contract must involve three things:

- an offer
- an acceptance
- a consideration

A contract must include consideration of money or other valuables. The basic idea is that two parties exchange things of value, such as time traded for money or technical knowledge for marketing skills. A written contract can involve hundreds of pages of terms and conditions qualifying the offer and the consideration. One final aspect of a contract is its freedom: the two parties must enter the contract voluntarily. A contract signed under duress or with fraudulent action is not binding. A contract does not have to be fair, in the sense of equivalent consideration for both parties, if both

parties freely accept the conditions.

Information is often exchanged under contract. Contracts are ideal for protecting the transfer of information because they can specify any conditions. "You have the right to use but not modify this information," "you have the right to use but not resell this information," or "you have the right to view this information yourself but not allow others to view it" are three potential contract conditions that could protect the commercial interests of an owner of information. Computer contracts typically involve the development and use of software and computerized data. As with tort law, the most common legal remedy in contract law is money.

15.4 RIGHTS OF EMPLOYEES AND EMPLOYERS

Employers hire employees to generate ideas and make products. The protection offered by copyrights, patents, and trade secrets appeals to employers because it applies to the ideas and products. However, the issue of who owns the ideas and products is complex. Ownership is a computer security concern because it relates to the rights of an employer to protect the secrecy and integrity of works produced by the employees. In this section we study the respective rights of employers and employees to their computer products.

15.4.1. Ownership of Products:

There are many different situations and interpreting the laws of ownership is difficult. Let us consider each type of protection in turn.

Ownership of a Patent:

The person who owns a work under patent or copyright law is the inventor. Under patent law, it is important to know who files the patent application. If an employee lets an employer patent an invention, the employer is deemed to own the patent and therefore the rights to the invention. The employer also has the right to patent if the employee's job functions included inventing the product. For instance, in a large company a scientist may be hired to do research and development, and the results of this inventive work become the property of the employer. Even if an employee patents something, the employer can argue for a right to use the invention if the employer contributed some resources in developing the invention.

Ownership of a Copyright:

Owning a copyright is like owning a patent. The author

(programmer) is the presumed owner of the work, and the owner has all rights to an object. However, a special situation known as *work for hire* applies to many copyrights for developing software or other products.

Work for Hire:

In a work for hire situation, the employer, *not* the employee, is considered the author of a work. Work for hire is not easy to identify and depends in part on the laws of the state in which the employment occurs. The relationship between an employee and employer is considered a work for hire if some or all the following conditions are true:

- The employer has a supervisory relationship, overseeing the way the creative work is done.
- The employer has the right to fire the employee.
- The employer arranges for the work to be done before the work was created.
- A written contract between the employer and employee states that the employer has hired the employee to do certain work.

Licenses:

An alternative to a work for hire arrangement is licensed software. In this situation, the programmer develops and retains full ownership of the software. In return for a fee, the programmer grants to a company a license to use the program. The license can be granted for a definite or unlimited period, for one copy or for an unlimited number, to use at one location or many, to use on one machine or all, at specified or unlimited times. This arrangement is highly advantageous to the programmer, just as a work for hire arrangement is highly advantageous to the employer. The choice between work for hire and license is largely what the two parties will agree to.

Trade Secret Protection:

A trade secret is different from either a patent or a copyright in that there is no registered inventor or author; there is no registration office for trade secrets. In the event a trade secret is revealed, the owner can prosecute the revealer for damages suffered. But first, ownership must be established because only the owner can be harmed. A company owns the trade secrets of its business-confidential data. As soon as a secret is developed, the company becomes the owner. As with copyrights, an employer may

argue about having contributed to the development of trade secrets. If your trade secret is an improved sorting algorithm and part of your job involves investigating and testing sorting algorithms, your employer will probably claim at least partial ownership of the algorithm you try to market.

Employment Contracts:

An employment contract often spells out rights of ownership. But sometimes the software developer and possible employer have no contract. Having a contract is desirable both for employees and employers so that both will understand their rights and responsibilities. Typically, an employment contract specifies that the employee be hired to work as a programmer exclusively for the benefit of the company. The company states that this is a work for hire situation. The company claims all rights to any programs developed, including all copyright rights and the right to market. The contract may further state that the employee is receiving access to certain trade secrets as a part of employment, and the employee agrees not to reveal those secrets to anyone. An agreement not to compete is sometimes included in a contract. The employee states that simply having worked for one employer will make the employee very valuable to a competitor. The employee agrees not to compete by working in the same field for a set period after termination. Agreements not to compete are not always enforceable in law; in some states the employee's right to earn a living takes precedence over the employer's rights.

15.5 REDRESS FOR SOFTWARE FAILURES

So far, we have considered programs, algorithms, and data as objects of ownership. But these objects vary in quality, and some of the legal issues involved with them concern the degree to which they function properly or well. In fact, people have legitimate differences of opinion on what constitutes "fair," "good," and "prudent" as these terms relate to computer software and programmers and vendors. The law applies most easily when there is broad consensus.

Program development is a human process of design, creation, and testing, involving a great deal of communication and interaction. For these reasons, there will always be errors in the software we produce. We sometimes expect perfect consumer products, such as automobiles or lawn mowers. At other times, we expect products to be "good enough" for use, in that most instances will be acceptable. But the situation with software is very different. To be fair, an operating system is a great deal more complex than many consumer products, and more opportunities for failure exist. For this

reason, this section addresses three questions:

- What are the legal issues in selling correct and usable software?
- What are the moral or ethical issues in producing correct and usable software?
- What are the moral or ethical issues in finding, reporting, publicizing, and fixing flaws?

In some ways, the legal issues are evolving. Everyone acknowledges that all vendors *should* produce good software, but that does not always happen. The more difficult concerns arise in the development and maintenance communities about what to do when faults are discovered.

15.5.1. Selling Correct Software:

Software is a product. It is built with a purpose and an audience in mind, and it is purchased by a consumer with an intended use in an expected context. And the consumer has some expectations of a reasonable level of quality and function. In that sense, buying software is like buying a radio. If you buy a faulty radio, you have certain legal rights relating to your purchase and you can enforce them in court if necessary. You may have three reactions if you find something wrong with the radio: You want your money back, you want a different (not faulty) radio, or you want someone to fix your radio. With software you have the same three possibilities, and we consider each one in turn. To consider our alternatives with software, we must first investigate the nature of the faulty code. Why was the software bad? One possibility is that it was presented on a defective medium. The second possibility is that the software worked properly, but you don't like it when you try it out. It may not do all it was advertised to do. Or you don't like the "look and feel," or it is slower than you expected it to be. The bottom line is that there is some attribute of the software that disappoints you, and you do not want this software. The final possibility is that the software malfunctions, so you cannot use it with your computer system. Here, too, you do not want the software and hope to return it.

I Want a Refund:

If the item were a radio, you would have the opportunity to look at it and listen to it in the shop, to assess its sound quality, measure its size (if it is to fit in a particular space), and inspect it for flaws. Do you have that opportunity with a program? Probably not. The U.S. Uniform Commercial Code (UCC) governs transactions between buyers and sellers in the United States. Section 2-601 says that "if the goods or the tender of delivery fail in any respect to conform to the contract, the buyer may reject them." You may have

had no opportunity to try out the software before purchase, particularly on your computer. Your inspection often could not occur in the store. So, you take home the software, only to find that it is free from flaws but does not fit your needs. You are entitled to a reasonable period to inspect the software, long enough to try out its features. If you decide within a reasonably short period of time that the product is not for you, you can cite UCC §2-601 to obtain a refund. More often, though, the reason you want to return the software is because it simply is not of high enough quality. Unfortunately, correctness of software is more difficult to enforce legally.

I Want It to Be Good:

Quality demands for mass market software are usually outside the range of legal enforcement for several reasons:

- Mass-market software is seldom totally bad. Certain features may not work, and faults may prevent some features from working as specified or as advertised. But the software works for most of its many users or works most of the time for all its users.
- The manufacturer has "deep pockets." An individual suing a major manufacturer could find that the manufacturer has a permanent legal staff of dozens of full-time attorneys. The cost to the individual of bringing a suit is prohibitive.
- Legal remedies typically result in monetary awards for damages, not a mandate to fix the faulty software.
- The manufacturer has little incentive to fix small problems. Unless a problem will seriously damage a manufacturer's image or possibly leave the manufacturer open to large damage amounts, there is little justification to fix problems that affect only a small number of users or that do not render the product unfit for general use.

The "fit for use" provision of the UCC dictates that the product must be usable for its intended purpose; software that doesn't work is clearly not usable. Some manufacturers are very attentive to their customers. When flaws are discovered, the manufacturers promptly investigate the problems and fix serious ones immediately, perhaps holding smaller corrections for a later release. These companies are motivated more by public image or moral obligation than by legal requirement.

15.5.2. Reporting Software Flaws:

Who should publicize flaws the user or the manufacturer? A user might want the recognition of finding a flaw; delaying the release might let someone else get that credit. A manufacturer might

want to ignore a problem or fail to credit the user. And either could say the other was wrong. And how should these flaws be reported? Several different viewpoints exist.

What You Don't Know Can Hurt You:

The several variants of Code Red in 2001 sparked a debate about whether we should allow full disclosure of the mechanisms that allow malicious code to enter and thrive in our systems. For example, the first variant of Code Red was relatively benign, but the third and fourth variants were powerful. When the first Code Red variant appeared, it was studied by many security analysts, including those at eEye Digital Security in Aliso Viejo, California. To pressure vendors and software managers to take seriously the threats they represent, eEye practices full disclosure of what it knows about security flaws. However, some observers claim that such open sharing of information is precisely what enables hackers to learn about vulnerabilities and then exploit them. And many security analysts encourage users and managers to apply patches right away, closing security holes before they can be exploited. But the patches require resources and may introduce other problems while fixing the initial one.

Each software-using organization must analyze and balance the risks and cost of not acting with the risks and costs of acting right away.

The Vendor's Interests:

Microsoft argues that producing one patch for each discovered vulnerability is inefficient both for the vendor and the user. The vendor might prefer to bundle several patches into a single service pack or, for noncritical vulnerabilities, to hold them until the next version. So, Microsoft would like to control if or when the report of a vulnerability goes public. Scott Culp argued that "a vendor's responsibility is to its customers, not to a self-described security community." He opposed what he called "information anarchy, ... the practice of deliberately publishing explicit, step-by-step instructions for exploiting security vulnerabilities without regard for how the information may be used." But he also acknowledged that the process of developing, distributing, and applying patches is imperfect, and his own company "need[s] to make it easier for users to keep their systems secure."

Users' Interests:

David Litchfield, a security researcher noted for locating flaws in vendors' programs, announced in May 2002 that he would no longer automatically wait for a vendor's patch before going public with a vulnerability announcement. Citing "lethargy and an

unwillingness to patch security problems as and when they are found," Litchfield criticized the approach of holding fixes of several vulnerabilities until enough had accumulated to warrant a single service pack. He makes the point that publicized or not, the vulnerabilities still exist. If one reporter has found the problem, so too could any number of malicious attackers. For a vendor to fail to provide timely patches to vulnerabilities of which the vendor is aware leaves the users wide open to attacks of which the user may be unaware. Litchfield's solution is to put pressure on the vendor. He announced he would give vendors one week's notice of a vulnerability before publicizing the vulnerability but not the details of how to exploit it to the world.

"Responsible" Vulnerability Reporting:

Clearly the conflicting interests of vendors and users must meet at some compromise position. Christey and Wysopal have proposed a vulnerability reporting process that meets constraints of timeliness, fair play, and responsibility. They call the user reporting a suspected vulnerability a "reporter" and the manufacturer the "vendor." A third party such as a computer emergency response center called a "coordinator" could also play a role when a conflict or power issue arises between reporter and vendor. Basically, the process requires reporter and vendor to do the following:

- The vendor must acknowledge a vulnerability report confidentially to the reporter.
- The vendor must agree that the vulnerability exists (or argue otherwise) confidentially to the reporter.
- The vendor must inform users of the vulnerability and any available countermeasures within 30 days or request additional time from the reporter as needed.
- After informing users, the vendor may request from the reporter a 30-day quiet period to allow users time to install patches.
- At the end of the quiet period the vendor and reporter should agree upon a date at which time the vulnerability information may be released to the general public.
- The vendor should credit the reporter with having located the vulnerability.
- If the vendor does not follow these steps, the reporter should work with a coordinator to determine a responsible way to publicize the vulnerability.

Such a proposal can only have the status of a commonly agreed-on process, since there is no authority that can enforce adherence on either users or vendors.

Quality Software:

Boris Beizer, a consultant, has said, "Software should be shipped with bugs. The zero-defect notion is mythological and theoretically unachievable. That doesn't mean shipping ill-behaved or useless software; it means being open with users about the bugs we find, sending notices or including the bug list, publishing the workarounds when we have them, and being honest and open about what we have and haven't yet tested and when we do and don't plan to test in the near future."

The whole debate over how and when to disclose vulnerabilities avoids the real issue. The world does not need faster patches, it needs better software with fewer vulnerabilities after delivery to the user. The issue is not how promptly a vulnerability is patched or how much detail is released with a vulnerability announcement. The issue is that, as the Anderson report noted over three decades ago, "penetrate and patch" is a fatally flawed concept: after a flaw was patched, the penetrators always found other old flaws or new flaws introduced because of or in the patch. The issue is technical, psychological, sociological, managerial, and economic. Until we produce consistently solid software, our entire computing infrastructure is seriously at risk.

15.6 COMPUTER CRIME

The law related to contracts and employment is difficult, but at least employees, objects, contracts, and owners are standard entities for which legal precedents have been developed over centuries. The definitions in copyright and patent law are strained when applied to computing because old forms must be made to fit new objects; for these situations, however, cases being decided now are establishing legal precedents. But crimes involving computers are an area of the law that is even less clear than the other areas.

15.6.1. Why a Separate Category for Computer Crime is needed?

Crimes can be organized into certain recognized categories, including *murder*, *robbery*, and *littering*. We do not separate crime into categories for different weapons, such as *gun crime* or *knife crime*, but we separate crime victims into categories, depending on whether they are *people* or *other objects*. Nevertheless, driving into your neighbour's picture window can be as bad as driving into his evergreen tree or pet sheep. Let us look at an example to see why these categories are not enough and why we need special laws relating to computers as subjects and objects of crime.

Rules of Property:

The legal system has explicit rules about what constitutes property. Generally, property is tangible, unlike magnetic impulses. For example, unauthorized use of a neighbour's lawn mower constitutes theft, even if the lawn mower was returned in essentially the same condition as it was when taken. To a computer professional, taking a copy of a software package without permission is clear-cut theft. Fortunately, laws evolve to fit the times, and this interpretation from the 1980s has been refined so that bits are now recognized items of property. A similar problem arises with computer services. We would generally agree that unauthorized access to a computing system is a crime. For example, if a stranger enters your garden and walks around, even if nothing is touched or damaged, the act is considered trespassing. However, because access by computer does not involve a physical object, not all courts punish it as a serious crime.

Rules of Evidence:

Computer printouts have been used as evidence in many successful prosecutions. Frequently used are computer records generated in the ordinary course of operation, such as system audit logs. Under the rules of evidence, courts prefer an original source document to a copy, under the assumption that the copy may be inaccurate or may have been modified in the copying process. The biggest difficulty with computer-based evidence in court is being able to demonstrate the authenticity of the evidence. Law enforcement officials operate under a chain of custody requirement: From the moment a piece of evidence is taken until it is presented in court, they track clearly and completely the order and identities of the people who had personal custody of that object. With computer-based evidence, it can be difficult to establish a chain of custody. If a crime occurred on Monday but was not discovered until Wednesday, who can verify that the log file was not altered?

Threats to Integrity and Confidentiality:

The integrity and secrecy of data are also issuing in many court cases. Parker and Nycom describe a case in which a trespasser gained remote access to a computing system. The computing system contained confidential records about people, and the integrity of the data was important. The prosecution of this case had to be phrased in terms of theft of computer time and valued as such, even though that was insignificant compared with loss of privacy and integrity. Why? Because the law as written recognized theft of computer time as a loss, but not loss of privacy or destruction of data. Now, however, several federal and state laws recognize the privacy of data about individuals. For example, disclosing grades or financial information without permission is a crime, and tort law would

recognize other cases of computer abuse.

Value of Data:

In another computer crime, a person was found guilty of having stolen a substantial amount of data from a computer data bank. However, the court determined that the "value" of that data was the cost of the paper on which it was printed, which was only a few dollars. Because of that valuation, this crime was classified as a misdemeanor and considered to be a minor crime. Fortunately, the courts have since determined that information and other intangibles can have significant value. The concept of what we value and how we determine its value is key to understanding the problems with computer-based law. Over time, the legal system will find ways to place a value on data that is representative of its value to those who use it. Although these methods of valuation are accepted in civil suits, they have not yet been widely accepted in criminal prosecution.

Acceptance of Computer Terminology:

The law is also lagging technology in its acceptance of definitions of computing terms. For example, according to a federal statute, it is unlawful to commit arson within a federal enclave (18 USC 81). Part of that act relates to "machinery or building material or supplies" in the enclave, but court decisions have ruled that a motor vehicle located within a federal enclave at the time of the burning was not included under this statute. Because of that ruling, it is not clear whether computer hardware constitutes "machinery" in this context; "supplies" almost certainly does not include software. Computers and their software, media, and data must be understood and accepted by the legal system.

15.6.2. Why Computer Crime is hard to define:

From these examples, the legal community has not accommodated advances in computers as rapidly as has the rest of society. Some people in the legal process do not understand computers and computing, so crimes involving computers are not always treated properly. Creating and changing laws are slow processes, intended to involve substantial thought about the effects of proposed changes. This deliberate process is very much out of pace with a technology that is progressing as fast as computing. Adding to the problem of a rapidly changing technology, a computer can perform many roles in a crime. A computer can be the subject, object, or medium of a crime. A computer can be attacked (attempted unauthorized access), used to attack (impersonating a legitimate node on a network), and used to commit crime (Trojan horse or fake

login). Computer crime statutes must address all these evils.

15.6.3. Why Computer Crime is hard to prosecute:

Even when everyone acknowledges that a computer crime has been committed, computer crime is hard to prosecute for the following reasons:

- *Lack of understanding:* Courts, lawyers, police agents, or jurors do not necessarily understand computers. Many judges began practicing law before the invention of computers, and most began before the widespread use of the personal computer. Fortunately, computer literacy in the courts is improving as judges, lawyers, and police officers use computers in their daily activities.
- *Lack of physical evidence:* Police and courts have for years depended on tangible evidence, such as fingerprints. As readers of Sherlock Holmes know, seemingly minuscule clues can lead to solutions to the most complicated crimes. But with many computer crimes there simply are no fingerprints and no physical clues of any sort.
- *Lack of recognition of assets:* We know what cash is, or diamonds, or even negotiable securities. But are twenty invisible magnetic spots equivalent to a million dollars? Is computer time an asset? What is the value of stolen computer time if the system would have been idle during the time of the theft?
- *Lack of political impact:* Solving and obtaining a conviction for a murder or robbery is popular with the public, and so it gets high priority with prosecutors and police chiefs. Solving and obtaining a conviction for an obscure high-tech crime, especially one not involving obvious and significant loss, may get less attention. However, as computing becomes more pervasive, the visibility and impact of computer crime will increase.
- *Complexity of case:* Basic crimes that everyone understands, such as murder, kidnapping, or auto theft, can be easy to prosecute. A complex money-laundering or tax fraud case may be more difficult to present to a jury because jurors have a hard time following a circuitous accounting trail. But the hardest crime to present may be a high-tech crime, described, for example, as root access by a buffer overflow in which memory was overwritten by other instructions, which allowed the attacker to copy and execute code at will and then delete the code, eliminating all traces of entry.
- *Age of defendant:* Many computer crimes are committed by juveniles. Society understands immaturity and disregards even very serious crimes by juveniles because the juveniles did not understand the impact of their actions. A more serious, related problem is that many adults see juvenile computer crimes as

childhood pranks, the modern equivalent of tipping over an outhouse.

Even when there is clear evidence of a crime, the victim may not want to prosecute because of possible negative publicity. Banks, insurance companies, investment firms, the government, and healthcare groups think their trust by the public will be diminished if a computer vulnerability is exposed. Also, they may fear repetition of the same crime by others: so-called copycat crimes. For all these reasons, computer crimes are often not prosecuted.

15.6.4. Why Computer Criminals are hard to catch:

As if computer crime laws and prosecution were not enough, it is also difficult for law enforcement agencies to catch computer criminals. There are two major reasons for this. First, computer crime is a multinational activity that must usually be pursued on a national or local level. There are no international laws on computer crime. Even though the major industrial nations cooperate very effectively on tracking computer criminals, criminals know there are "safe havens" from which they cannot be caught. Often, the trail of a criminal stops cold at the boundary of a country. Riptech Inc. studies Internet attack trends by many factors. For the period January-June 2002 the United States led the world in source of Internet attacks (40 percent) followed by Germany (7 percent). But when you normalize these data for number of users, a very different pattern emerges. Per Internet user, Israel and Hong Kong lead among those nations with more than 1 million users, and Kuwait and Iran top the list among nations with fewer than 1 million users. Nations all over the globe appear on these lists, which demonstrates that attackers can and do operate from many different countries.

Complexity is an even more significant factor than country of origin. As we have stated throughout this book, networked attacks are hard to trace and investigate because they can involve so many steps. A smart attacker will "bounce" an attack through many places to obscure the trail. Each step along the way makes the investigator complete more legal steps. If the trail leads from server A to B to C, the law enforcement investigators need a search warrant for data at A, and others for B and C. Even after obtaining the search warrants, the investigator must find the right administrator and serve the warrants to begin obtaining data. In the time the investigator must get and serve warrants, not to mention follow leads and correlate findings, the attacker has carefully erased the digital evidence.

15.6.5. What Computer Crime does not address:

Even with the definitions included in the statutes, the courts

must interpret what a computer is. Legislators cannot define precisely what a computer is because computer technology is used in many other devices, such as robots, calculators, watches, automobiles, microwave ovens, and medical instruments. More importantly, we cannot predict what kinds of devices may be invented ten or fifty years from now. Therefore, the language in each of these laws indicates the kinds of devices the legislature seeks to include as computers and leaves it up to the court to rule on a specific case. Unfortunately, it takes a while for courts to build up a pattern of cases, and different courts may rule differently in similar situations. The interpretation of each of these terms will be unsettled for some time to come. Value presents a similar problem. As noted in some of the cases presented, the courts have trouble separating the intrinsic value of an object (such as a sheet of paper with writing on it) from its cost to reproduce. The courts now recognize that a Van Gogh painting is worth more than the cost of the canvas and paint. But the courts have not agreed on the value of printed computer output. The cost of a blank diskette is miniscule, but it may have taken thousands of hours of data gathering and machine time to produce the data encoded on a diskette. The courts are still striving to determine the fair value of computer objects. Both the value of a person's privacy and the confidentiality of data about a person are even less settled.

15.7 ETHICAL ISSUES IN COMPUTER SOCIETY

This final section helps clarify thinking about the ethical issues involved in computer security. We list and explain some ethical principles. The primary purpose of this section is to explore some of the ethical issues associated with computer security and to show how ethics functions as a control.

15.7.1. Differences between the Law and Ethics:

As we noted earlier, law is not always the appropriate way to deal with issues of human behaviour. It is difficult to define a law to preclude only the events we want it to. For example, a law that restricts animals from public places must be refined to *permit* guide dogs for the blind. Lawmakers, who are not computer professionals, are hard pressed to think of all the exceptions when they draft a law concerning computer affairs. Even when a law is well conceived and well written, its enforcement may be difficult. The courts are overburdened and prosecuting relatively minor infractions may be excessively time consuming relative to the benefit. Thus, it is impossible or impractical to develop laws to describe and enforce all forms of behaviour acceptable to society. Instead, society relies on ethics or morals to prescribe generally accepted standards of proper behaviour.

An ethic is an objectively defined standard of right and wrong. Ethical standards are often idealistic principles because they focus on one objective. In each situation, however, several moral objectives may be involved, so people must determine an action that is appropriate considering all the objectives. Even though religious groups and professional organizations promote certain standards of ethical behaviour, ultimately each person is responsible for deciding what to do in a specific situation. Therefore, through our choices, each of us defines a personal set of ethical practices. A set of ethical principles is called an ethical system.

An ethic is different from a law in several important ways. First, laws apply to everyone: One may disagree with the intent or the meaning of a law, but that is not an excuse for disobeying the law. Second, the courts have a regular process for determining which law supersedes which if two laws conflict. Third, the laws and the courts identify certain actions as right and others as wrong. From a legal standpoint, anything that is not illegal is right. Finally, laws can be enforced to rectify wrongs done by unlawful behaviour. By contrast, ethics are personal: two people may have different frameworks for making moral judgments. What one person deems perfectly justifiable; another would never consider doing. Second, ethical positions can and often do come into conflict. As an example, the value of a human life is very important in most ethical systems. Most people would not cause the sacrifice of one life, but in the right context some would approve of sacrificing one person to save another, or one to save many others. The value of one life cannot be readily measured against the value of others, and many ethical decisions must be founded on precisely this ambiguity. Yet, there is no arbiter of ethical positions: when two ethical goals collide, each person must choose which goal is dominant. Third, two people may assess ethical values differently; no universal standard of right and wrong exists in ethical judgments. Nor can one person simply look to what another has done as guidance for choosing the right thing to do. Finally, there is no enforcement for ethical choices. These differences are summarized in Table 11-3.

Table 11-3. Contrast of Law vs. Ethics.

Law	Ethics
Described by formal, written documents	Described by unwritten principles
Interpreted by courts	Interpreted by each individual
Established by legislatures representing all people	Presented by philosophers, religions, professional groups
Applied to everyone	Chosen personally
Priority determined by courts if two laws conflict	Priority determined by an individual if two principles conflict
"Right" arbitrated finally by court	Not arbitrated externally
Enforced by police and courts	Enforced by intangibles such as principles and beliefs

15.7.2. Studying Ethics:

The study of ethics is not easy because the issues are complex. Sometimes people confuse ethics with religion because many religions supply a framework in which to make ethical choices. However, ethics can be studied apart from any religious connection. Difficult choices would be easier to make if there were a set of universal ethical principles to which everyone agreed. But the variety of social, cultural, and religious beliefs makes the identification of such a set of universal principles impossible. In this section we explore some of these problems and then consider how understanding ethics can help in dealing with issues of computer security.

Ethics and Religion:

Ethics is a set of principles or norms for justifying what is right or wrong in each situation. To understand what ethics *is* we may start by trying to understand what it *is not*. Ethical principles are different from religious beliefs. Religion is based on personal notions about the creation of the world and the existence of controlling forces or beings. Many moral principles are embodied in the major religions, and the basis of a personal morality is a matter of belief and conviction, much the same as for religions. However, two people with different religious backgrounds may develop the same ethical philosophy, while two exponents of the same religion might reach opposite ethical conclusions in a situation. Finally, we can analyze a situation from an ethical perspective and reach ethical conclusions without appealing to any religion or religious framework. Thus, it is important to distinguish ethics from religion.

Ethical Principles Are Not Universal:

Ethical values vary by society, and from person to person within a society. For example, the concept of privacy is important in Western cultures. But in Eastern cultures, privacy is not desirable because people associate privacy with having something to hide. Not only is a Westerner's desire for privacy not understood but in fact it has a negative connotation. Therefore, the attitudes of people may be affected by culture or background. Also, an individual's standards of behaviour may be influenced by past events in life. A person who grew up in a large family may place greater emphasis on personal control and ownership of possessions than would an only child who seldom had to share. Major events or close contact with others can also shape one's ethical position. Despite these differences, the underlying principles of how to make moral judgment are the same. Although these aspects of ethics are quite reasonable and understandable, they lead people to distrust ethics because it is not founded on basic principles all can accept. Also, people from a scientific or technical background expect precision and universality.

Ethics Does Not Provide Answers:

Ethical pluralism is recognizing or admitting that more than one position may be ethically justifiable even equally so in each situation. Pluralism is another way of noting that two people may legitimately disagree on issues of ethics. We expect and accept disagreement in such areas as politics and religion. However, in the scientific and technical fields, people expect to find unique, unambiguous, and unequivocal answers. In science, one answer must be correct or demonstrable in some sense. Science has provided life with fundamental explanations.

Ethics is rejected or misunderstood by some scientists because it is "soft," meaning that it has no underlying framework, or it does not depend on fundamental truths. Scientists are uncomfortable with ethics because ethics does not provide these clean distinctions. Worse, there is no higher authority of ethical truth. Two people may disagree on their opinion of the ethics of a situation, but there is no one to whom to appeal for a final determination of who is "right." Conflicting answers do not deter one from considering ethical issues in computer security. Nor do they excuse us from making and defending ethical choices.

15.7.3. Ethical Reasoning:

Most people make ethical judgments often, perhaps daily. (Is it better to buy from a hometown merchant or from a nationwide chain? Should I spend time with a volunteer organization or with my friends? Is it acceptable to release sensitive data to someone who

might not have justification for access to that data?) Because we all engage in ethical choice, we should clarify how we do this so that we can learn to apply the principles of ethics in professional situations, as we do in private life. Study of ethics can yield two positive results. First, in situations in which we already know what is right and what is wrong, ethics should help us justify our choice. Second, if we do not know the ethical action to take in a situation, ethics can help us identify the issues involved so that we can make reasoned judgments.

Examining a Case for Ethical Issues:

How, then, can issues of ethical choice in computer security be approached? Here are several steps to making and justifying an ethical choice:

- *Understand the situation:* Learn the facts of the situation. Ask questions of interpretation or clarification. Attempt to find out whether any relevant forces have not been considered.
- *Know several theories of ethical reasoning:* To make an ethical choice, you must know how those choices can be justified.
- *List the ethical principles involved:* What different philosophies could be applied in this case? Do any of these include others?
- *Determine which principles outweigh others:* This is a subjective evaluation. It often involves extending a principle to a logical conclusion or determining cases in which one principle clearly supersedes another.

Examples of Ethical Principles:

There are two different schools of ethical reasoning: one based on the good that results from actions and one based on certain prima facie duties of people.

Consequence-Based Principles:

The teleological theory of ethics focuses on the consequences of an action. *Teleology* is the general name applied to many theories of behaviour, all of which focus on the goal, outcome, or consequence of the action. There are two important forms of teleology. Egoism is the form that says a moral judgment is based on the positive benefits to the person taking the action. An egoist weighs the outcomes of all possible acts and chooses the one that produces the most personal good for him or her with the least negative consequence. The effects on other people are not relevant. The principle of utilitarianism is also an assessment of good and bad results, but the reference group is the entire universe. The utilitarian

chooses that action that will bring the greatest collective good for all people with the least possible negative for all. In this situation, the utilitarian would assess personal good and bad, good and bad for the company, good and bad for the customer, and, perhaps, good and bad for society at large. For example, a developer designing software to monitor smokestack emissions would need to assess its effects on everyone breathing. The utilitarian might perceive greater good to everyone by taking the time to write high-quality code, despite the negative personal consequence of displeasing management.

Rule-Based Principles:

Another ethical theory is deontology, which is founded in a sense of duty. This ethical principle states that certain things are good in and of themselves. These things that are naturally good are good rules or acts, which require no higher justification. Something just *is* good; it does not have to be judged for its effect. Examples of intrinsically good things are:

- truth, knowledge, and true opinion of various kinds; understanding, wisdom
- just distribution of good and evil; justice
- pleasure, satisfaction; happiness; life, consciousness
- peace, security, freedom
- good reputation, honour, esteem; mutual affection, love, friendship, cooperation; morally good dispositions or virtues
- beauty, aesthetic experience

Rule-deontology is the school of ethical reasoning that believes certain universal, self-evident, natural rules specify our proper conduct. Certain basic moral principles are adhered to because of our responsibilities to one another; these principles are often stated as rights: the right to know, the right to privacy, the right to fair compensation for work. Sir David Ross lists various duties incumbent on all human beings:

- *fidelity*, or truthfulness
- *reparation*, the duty to recompense for a previous wrongful act
- *gratitude*, thankfulness for previous services or kind acts
- *justice*, distribution of happiness in accordance with merit
- *beneficence*, the obligation to help other people or to make their lives better
- *nonmaleficence*, not harming others
- *self-improvement*, to become continually better, both in a mental sense and in a moral sense

Another school of reasoning is based on rules derived by everyone. Religion, teaching, experience, and reflection lead each person to a set of personal moral principles. The answer to an ethical question is found by weighing values in terms of what a person believes to be right behaviour.

15.8 SUMMARY

- Contracts help fill the voids among criminal, civil, and tort law. That is, in the absence of relevant statutes, we first see common tort law develop. But people then enhance these laws by writing contracts with the specific protections they want.
- Enforcement of civil law, torts or contracts, can be expensive because it requires one party to sue the other. The legal system is informally weighted by money. It is attractive to sue a wealthy party who could pay a hefty judgment. And a big company that can afford dozens of top-quality lawyers will more likely prevail in a suit than an average individual.
- There are four aspects of the relationship between computing and the law.
 - First is the legal mechanisms of copyright, patent, and trade secret as means to protect the secrecy of computer hardware, software, and data. These mechanisms were designed before the invention of the computer, so their applicability to computing needs is somewhat limited. However, program protection is especially desired, and software companies are pressing the courts to extend the interpretation of these means of protection to include computers.
 - Second is the relationship between employers and employees, in the context of writers of software. Well-established laws and precedents control the acceptable access an employee has to software written for a company.
 - Third, is the legal side of software vulnerabilities: Who is liable for errors in software, and how is that liability enforced?
 - Fourth, we noted some of the difficulties in prosecuting computer crime. Several examples showed how breaches of computer security are treated by the courts. In general, the courts have not yet granted computers, software, and data appropriate status, considering value of assets and seriousness of crime. The legal system is moving cautiously in its acceptance of computers.

- In this study of ethics, we have tried not to decide right and wrong, or even to brand certain acts as ethical or unethical. The purpose is to stimulate thinking about ethical issues concerned with confidentiality, integrity, and availability of data and computations.
 - The important first step in acting ethically in a situation is to obtain the facts, ask about any uncertainties, and acquire any additional information needed. In other words, first we must understand the situation.
 - The second step is to identify the ethical principles involved. Honesty, fair play, proper compensation, and respect for privacy are all ethical principles. Sometimes these conflict, and then we must determine which principles are more important than others. This analysis may not lead to one principle that obviously overshadows all others. Still, a ranking to identify the major principles involved is needed.
 - The third step is choosing an action that meets these ethical principles. Deciding and acting are difficult, especially if the action has evident negative consequences. However, acting based on a *personal* ranking of principles is necessary. The fact that other equally sensible people may choose a different action does not excuse us from taking some action. Decisions may vary, based on fine differences between two situations. Or a person's views can change over time in response to experience and changing context. Learning to reason about ethical situations is not quite the same as learning "right" from "wrong." Terms such as *right* and *wrong* or *good* and *bad* imply a universal set of values. Yet we know that even widely accepted principles are overridden by some people in some situations. Therefore, our purpose in introducing this material has been to stimulate you to recognize and think about ethical principles involved in cases related to computer security. Only by recognizing and analyzing principles can you act consistently, thoughtfully, and responsibly.

15.9 REVIEW QUESTIONS

- a) Write a short note on copyrights
- b) Write a short note on patents
- c) Write a short note on trade secrets
- d) Compare copyright, patent and trade secret.
- e) Why is information different from other commercial things?
- f) Write a short note on tort law.
- g) Explain the contract law.

- h) How can software failures be redressed?
- i) Explain the various viewpoints for reporting software flaws.
- j) Why a separate category for computer crime is needed?
- k) Why is computer crime hard to prosecute?
- l) Differentiate between law and ethics.
- m) How can issues of ethical choice in computer security be approached?
- n) Explain consequence-based and rule-based principles of ethical reasoning.

15.10 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- *Security in Computing* by C. P. Pfleeger, and S. L. Pfleeger, Pearson Education.
- *Computer Security: Art and Science* by Matt Bishop, Pearson Education.
- *Cryptography and Network Security: Principles and practice* by Stallings
- *Network Security* by Kaufman, Perlman, Speciner
- *Network Security : A Beginner's Guide* by Eric Maiwald, TMH
- *Java Network Security* by Macro Pistoia, Pearson Education
- *Principles of information security* by Whitman, Mattord, Thomson

