1st class

# Principals of security

# مبادئ الامنيه

استاذة الماده:أ.م.د. هاله بهجت عبد الوهاب

# *Security Principles* *First class-Security Branch*

**References:**

1- Book, “Network security Foundation-2007” (PDF).
2- Book,” Cryptography and Network Security Principles and Practices”, 2005,(PDF).

**Contents:**

# Chapter 1

## 1- ***Introduction***:

Security is the sum of all measures taken to prevent loss of any kind. Loss can occur because of user error, defects in code, malicious acts, hardware failure, and acts of nature. With holistic computer security, a number of methods are used to prevent these events, but it's primarily focused on
Preventing user error and malicious acts. Security is the antithesis of convenience—generally, the more secure something is, the less convenient it is. Think about this in the context of your life: think of how easy it would be if you could just walk up and push a button to start your car without worrying about keys—or paying for car insurance. But the risk of theft and accidents makes these two security measures mandatory. Meanwhile, advanced technology like remote key fobs for cars is making automotive

security easier, just as biometric scanners can make logging on to computers both more secure and less annoying at the same time. Computer security is not complicated. It may seem that way, but the theory behind computer security is relatively simple. Hacking methods fall into just a few categories. And solutions to computer security problems are actually rather straightforward.

## 2- *Why Computers Aren't Secure*:

Most people question why computers are so insecure—after all, people have been hacking for a long time. The vast majority of hacking incidents occur because of one of the following pervasive problems:

### 1- Security is an annoyance.

Administrators often fail to implement security features in operating systems because doing so causes problems for users. Users also circumvent security—by choosing easy-to-use (easy to- guess) passwords like "123456," never changing those passwords, disclosing those passwords to co-workers, or sharing user accounts. Vendors ship software so that it will install in the most feature-filled configuration with its security features disabled so that unskilled users won't run into roadblocks and don't have to understand and configure it correctly before they use it. This means that the vast majority of installations are never properly secured. The fact that strong security is an annoyance that requires extra learning on the part of everyone involved is the most common reason for security failures.

## 2- Features are rushed to market.

Vendors concentrate their efforts on adding features that make their software more useful, with little thought to security. A perfect example of this is the addition of scripting language support to Microsoft Outlook and Outlook Express.

## 3- virus

*Virus is any program that automatically replicates itself*. When the Internet first took off, "e-mail *Virus"* scares propagated around the Net via e-mail. Computer security experts ignored them, knowing that a virus required an execution environment like a computer language in order to actually propagate. They laughed at the possibility that anyone would actually tie a computer language to an e-mail system because anyone with any security consciousness at all would never let this happen. Despite the warnings, and even though the scripting language support built in to Microsoft Office had already been exploited to create "Macro" viruses embedded in Word and Excel documents, Microsoft ignored the signs and the explicit warnings of its own employees and incorporated a scripting language into its e-mail software. Even worse, it was set up to automatically execute code contained in e-mail messages, configured to do so by default, and included features like "auto-preview" that even opened the messages upon arrival and executed the embedded code. To make matters even more egregious, Microsoft shipped this insecure software for free with every copy of their ubiquitous Windows operating system, thus ensuring that it would be widely deployed.

## 4- hacker

*Hacker is one who engages in hacking*. Thus, the plague that is e-mail viruses today arrived—well predicted, forewarned, and completely ignored by a vendor in order to implement a feature that less than 1 percent of legitimate users actually ever use. Microsoft simply didn't concern itself with even a cursory study of the security implications of adding this feature to its software. It couldn't have done a better job of implementing a new hacking exploit if it had been doing it on purpose. **Vendors who spend time on security are eclipsed by the competition.** Customers don't truly value security. If they did, they would use older, well-tested, security-proven software that doesn't have all the bells and whistles of the latest versions. Companies like Microsoft that retrofitted their existing products to work on the Internet decimated their competition. Had they waited to do it securely, they would have been beaten to market by someone who didn't. The end result? The least-secure products always get to market first and become standards.

## 4.1 Computers and software evolve very quickly.

Computers and networking technology have been evolving far faster than companies can predict what might go wrong with them. Moore's law states that computer hardware will double in power every two years. His prediction has been eerily accurate for over three decades now. Protocols that were not developed to be secure were adapted to purposes that they were never intended for and then grew in popularity to a far wider audience than the original creators could have imagined.

- **Windows is a** family of single-user operating systems developed by Microsoft for small computers. The most recent version has incorporated enhancements to allow multiple users to run programs directly on the machine.

- **Programmers can't accurately predict flaws.** Programmers rarely consider that the state of their functions might be externally changed to any possible value while the code is running, so they only check for values that they send to it themselves. Once the code passes its normal debugging checks, it's shipped without having been tested to pass a barrage of random data thrown at it. Even if they did attempt to predict flaws, the 10 programmers who created a project could never come up with the complete set of attacks that the million hackers who attempt to exploit it will.

## 5- Security Concepts

Computer security is based on the same concepts that physical security is: trust, knowledge of a secret to prove authenticity, possession of a key to open locks, and legal accountability. The metaphors are so apt that most computer security mechanisms even have the same names as their physical counterparts.

### 5-1 security mechanisms:

- **Trust**

All computer security springs from the concept of inherent or original trust. Just as a child inherently trusts its parents, a secure computer system inherently trusts those who set it up. While this may seem rather obvious, it is an important concept because it is the origination of all subsequent security measures. There's more inherent trust in computer security than simply the original establishment of a system. For example, you trust that there are no "back doors" in the software you use that could be exploited by a knowledgeable person to gain access. You trust that the login screen that you are looking at is actually the system's true login screen and not a mimic designed to collect your password and then pass it to a remote system. Finally, you trust that the designers of the system have not made any serious mistakes that could obviate your security measures.

- **Authentication**

  Authentication **is** the process of determining the identification of a user. *Authentication* is the process of determining the identity of a user. Forcing the user to prove that they know a secret that should be known only to them proves that they are who they say they are.

  1- **User account:** A record containing information that identifies a user, including a secret password. *User accounts* are associated with some form of secret, such as a password, PIN, biometric hash, or a device like a *smart card* that contains a larger, more secure password than a human could remember. To the system, there is no concept of a human; there is only secret, information tied to that secret, and information to which that secret has access.

  **2- Smart card:** An electronic device containing a simple calculator preprogrammed with a code that cannot be retrieved. When given a challenge, it can calculate a response that proves it knows the code without revealing what the code is.

Authentication is only useful in so far as it is accurate. Passwords are probably the least reliable form of authentication in common use today, but they're also the most easily implemented—they require no special hardware and no sophisticated algorithms for basic use. However, they are easily guessed, and even when they're carefully chosen it's still possible to simply guess the entire range of possible passwords on many systems in short order. A less common but more secure method of authentication is to physically possess a unique key. This is analogous to most physical locks. In computer security systems, “keys” are actually large numbers generated by special algorithms that incorporate information about the user and are stored on removable media like smart cards. The problem with keys is that, like physical keys, they can be lost or stolen. However, when combined with a password, they are very secure and difficult to thwart. Another form of authentication provides inherent identification by using a physical property of the user. This is called biometric authentication, and it relies upon unique and unchangeable physical properties of a human, such as handwriting characteristics, fingerprints, facial characteristics, and so forth. Biometric authentication has the potential to be the most reliable form of authentication because it’s easy to use, nearly impossible to fake when correctly implemented, and can’t be circumvented for the sake of convenience. Some forms of biometric authentication are easier to “forge” than others, and naïve implementations can sometimes be easily faked. But when well implemented, biometric authentication is the most secure form of authentication and the only form that can be truly said to uniquely and unmistakably identify a user.

- **Chain of Authority**

***Trust provider*** A trusted third party that certifies the identity of all parties in a secure transaction. Trust providers do this by verifying the identity of each party and generating digital certificates that can be used to determine that identity. Trust providers perform a function analogous to a notary public. During the installation of a security system, the original administrator will create the root account. From the root account (called the “administrator” account in Windows and the “Supervisor” account in NetWare), all other accounts, keys, and certificates spring. Every account on

a system, even massive systems containing millions of accounts, spring from this chain of authority. The concept of chains of authority isn’t often discussed because it is inherent in a secure system. Certificate systems are also based on a chain of authority. Consider the case of separate businesses that do a lot of work together. It would be convenient if users from Business Alpha could automatically log on to computers at Business Beta. But because these two systems have two different chains of authority, there’s no way for Business Alpha to trust that users who say they are from Business Beta actually are. This problem is solved by having both businesses trust a third-party *trust provider,*or a company that specializes in verifying identity and creating secure certificates that can be used to prove identity to foreign systems. As long as both businesses trust the same trust provider, they are rooted in the same chain of authority and can trust certificates that are generated by that trust provider. Trust providers are the digital equivalent of a notary public. Examples of trust providers
are VeriSign and Thawed.

- **Accountability**

Accountability is where the secret meets the user. Users don’t try to circumvent security because their identity would be known and they would be held legally accountable for their actions. It is accountability, rather than access controls, that prevents illegal behavior. In pure accountability-based systems, no access control mechanisms are present. Users simply know that their every action is being logged, and since their identity is known and their activities are tracked, they won’t do things that could jeopardize their position (unless something happens to make them no longer care). The problem with accountability-based systems is twofold—they only work if identity can’t be faked, and there are rare occasions where users lose their inhibitions. Without access control, these users can destroy the entire system. For these reasons, accountability-based security is normally used to augment access control systems rather than to replace them.

- **Access Control**

Access control is the security methodology that allows access to information based on identity. Users who have been given permission or keys to information can access it—otherwise, access is denied.

**Permissions-Based Access Control**

- **File :**A sequence of related information referenced by a filename in a directory.

Once the system knows the identity of an individual because they've been Authenticated, the system can selectively allow or deny access to resources like stored files based on that identity. This is called permissions-based security because users are either granted or denied permission to access a *File* or other resource. The question of who has access to which files is typically either defined by administrators when the system is implemented or created according to some set of default rules programmed into the system; for instance, the original creator (owner) of a file is the only user who can change it. Access controls are typically implemented either as directory permissions that apply to all files within the directory or by an access control list, which is a component of a file that explicitly lists which users can access it. Typically, when a **Encryption-Based Access Control (Privacy)**

- **private key :** The key used to decode public key messages that must be kept private.

A totally different way to control access is to simply encrypt data using public key encryption. Access to the encrypted data is given to those who want it, but it's worthless to them unless they have the
*private key* required to decode it. Using PKE to secure data works very well, but it requires considerably more processing power to encode and decode data.

# Chapter 2

- ***Understanding Hacking***

Know thy enemy. Hackers are the reason you need to implement computer security, and an in-depth defense against any adversary requires an
in-depth understanding of that adversary. This chapter describes hackers, their motivations, and their methods. By knowing a hacker's motivations, you can predict your own risk level and adapt your specific defenses to ward off the type of hackers you expect to attack your network while retaining as much usability as possible for your legitimate users.

## What Is Hacking?

Hacking is quite simply the attempt to gain access to a computer system without authorization. Originally, the term *hacker* simply referred to an adept computer user, and gurus still use the term to refer to themselves in that original sense. But when breaking into computer systems (technically known as *cracking*) became popular, the media used the *hacker* to refer only to computer criminals, thus popularizing only the negative connotation. In this book, we refer only to that negative connotation as well.

## Types of Hackers

Learning to hack takes an enormous amount of time, as does perpetrating actual acts of hacking. Because of the time it takes, there are only two serious types of hackers: the underemployed and those hackers being paid by someone to hack. The word *hacker* conjures up images of skinny teenage boys aglow in the phosphor of their monitors. Indeed, this group makes up the largest portion of the teeming millions of hackers, but they are far from the most serious threat.Hackers fall quite specifically into these categories, in order of increasing threat:

1-Security experts

2-Script kiddies

3-Underemployed adults

4-Ideological hackers

5-Criminal hackers

6-Corporate spies

7-Disgruntled employees

- **Security Experts**

Most security experts are capable of hacking but decline to do so for moral or economic reasons. Computer security experts have found that there’s more money in preventing hacking than in perpetrating it, so they spend their time keeping up with the hacking community and current techniques in order to make themselves more effective in the fight against it. A number of larger Internet service companies employ ethical hackers to test their security systems and those of their large customers, and hundreds of former hackers now consult independently as security experts to medium-sized businesses. These experts often are the first to find new hacking exploits, and they often write software to test or exacerbate a condition.

Practicing hackers can exploit this software just as they can exploit any other software.

- **Script Kiddies**

**script kiddie** A novice hacker.*Script kiddies* are students who hack and are currently enrolled in some scholastic endeavor—junior high, high school, or college. Their parents support them, and if they have a job, it's only part-time. They are usually enrolled in whatever computer-related courses are available, if only to have access to the computer lab. These hackers may use their own computers, or (especially at colleges) they may use the more powerful resources of the school to perpetrate their hacks. Script kiddies joyride through cyberspace looking for targets of opportunity
and are concerned mostly with impressing their peers and not getting caught. They usually are not motivated to harm you, and in most instances, you'll never know they were there unless you have software that detects unusual activity and notifies you or a firewall that logs attacks—or unless they make a mistake. These hackers constitute about 90 percent of the total manual hacking activity on the Internet. If you consider the hacking community as an economic endeavor, these hackers are the consumers. They use the tools produced by others, stand in awe of the hacking feats of others, and generally produce a fan base to whom more serious script kiddies and underemployed adult hackers play. Any serious attempt at security will keep these hackers at bay. In addition to the desire to impress their peers, script kiddies hack primarily to get free stuff: software and music, mostly. They share pirated software amongst themselves, make MP3 compressed audio tracks from CDs of their favorite music, and trade the serial numbers needed to unlock the full functionality of demo software that can be downloaded from the Internet.

- **Vectors That Hackers Exploit**

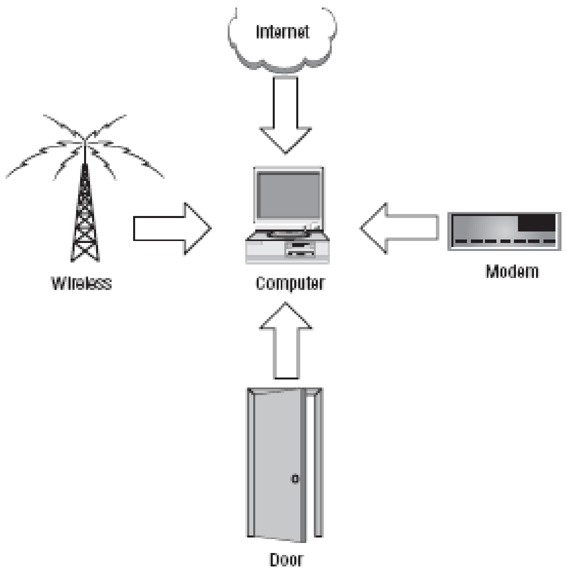
There are only four ways for a hacker to access your network:

1-By connecting over the Internet

2-By using a computer on your network directly

3-By dialing in via a Remote Access Service (RAS) server

4-By connecting via a nonsecure wireless network



**Direct Intrusion**

Hackers are notoriously nonchalant and have, on numerous occasions, simply walked into businesses, sat down at a local terminal or network client, and begun setting the stage for further remote penetration.

In large companies, there’s no way to know everyone by sight, so an unfamiliar worker in the IT department isn’t uncommon or suspicious at all. In companies that don’t have ID badges or security guards, it isn’t anybody’s job to check credentials, so penetration is relatively easy. And even in small companies, it’s easy to put on a pair of coveralls and pretend to be with a telephone or network wiring company or even pose as the spouse of a fictitious employee. With a simple excuse like telephone problems in the area, access to the server room is granted (oddly,these are nearly always

colocated with telephone equipment). If left unattended,a hacker can simply create a new administrative user account. In less than a
minute, a small external modem or wireless access point can be attached without even rebooting your server. Solving the direct intrusion problem is easy: Employ strong physical security at your premises and treat any cable or connection that leaves the building as a security concern. This means putting firewalls between your WAN links and your internal network or behind wireless links. By employing your firewalls to monitor any connections that leave the building, you are able to eliminate direct
intrusion as a vector.

- **Dial-Up**

Dial-up hacking, via modems, used to be the only sort of hacking that existed, but it has quickly fallen to second place after Internet intrusions. (Hacking over the Internet is simply easier and more interesting for hackers.)This doesn't mean that the dial-up vector has gone away—hackers with a specific target will employ any available means to gain access.

Although the dial-up problem usually means exploiting a modem attached to a Remote Access Service (RAS) server, it also includes the problem of dialing into individual computers. Any modem that has been set to answer for the purpose of allowing remote access or remote control for the employee who uses the computer presents a security concern. Many organizations allow employees to remotely access their computers from home using this method. Containing the dial-up problem is conceptually easy: Put your RAS servers outside your firewall in the public security zone, and force legitimate users to
authenticate with your firewall first to gain access to private network resources. Allow no device to answer a telephone line behind your firewall.

This eliminates dial-up as a vector by forcing it to work like any other Internet connection.

- **Internet**

Internet intrusion is the most available, most easily exploited, and most problematic vector of intrusion into your network. This vector is the primary topic of this book. If you follow the advice in this section, the Internet will be the only true vector into your network. You already know that the Internet vector is solved by using firewalls, so there’s no point in belaboring the topic here. The remainder of this book is about solving the Internet intrusion vector.

- **Hacking Techniques**

Hacking attacks progress in a series of stages, using various tools and techniques. A hacking session consists of the following stages:

1-Target selection

2-Information gathering

3-Attack

The hacker will attempt to find out more about your network through each successive attack, so these stages actually feed back into the process as more information is gathered from failed attacks.

# Firewall

## 1 Introduction:

In general firewall process is the same as process of farmer who is attempting to protect his hen house from the fox. The lay of land places the hen house in the center of the barnyard, which is surrounded by a fence to keep the animals in place. Another fence is in place at the perimeter around the farmer’s property. Keep in mind where the chickens are housed. The farmer places a chair at the front gate on the outer perimeter, and sits down to watch for the fox. Now he is in position to act the same as firewall would act in a network environment. Both are protecting the main point of entry into the system.

## 2. Firewall Definition:

A firewall protects networked computers from intentional hostile intrusion that could compromise confidentiality or results in data corruption or denial of service. It may be a hardware device as shown in figure (1), or a software program running on secure host computer as shown in figure (2) .

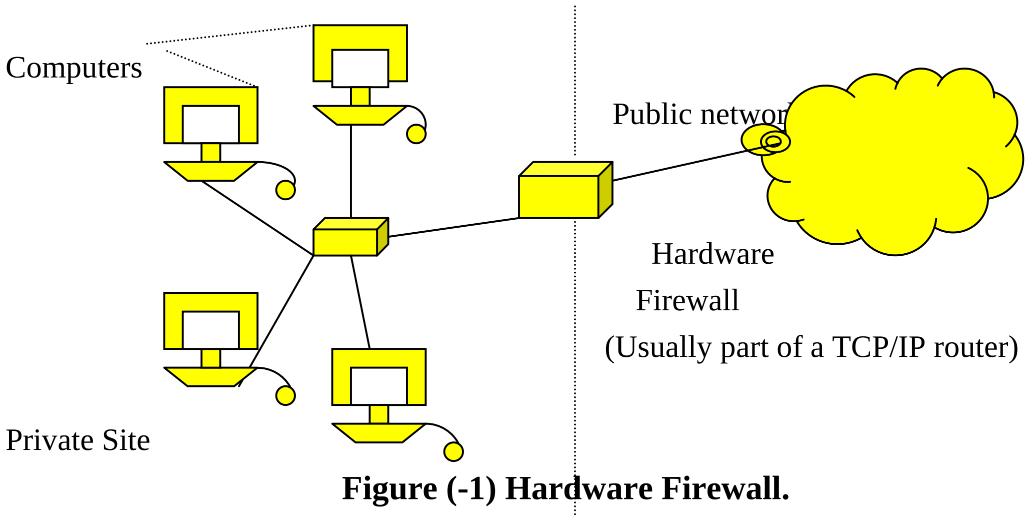


**Figure (-1) Hardware Firewall.**

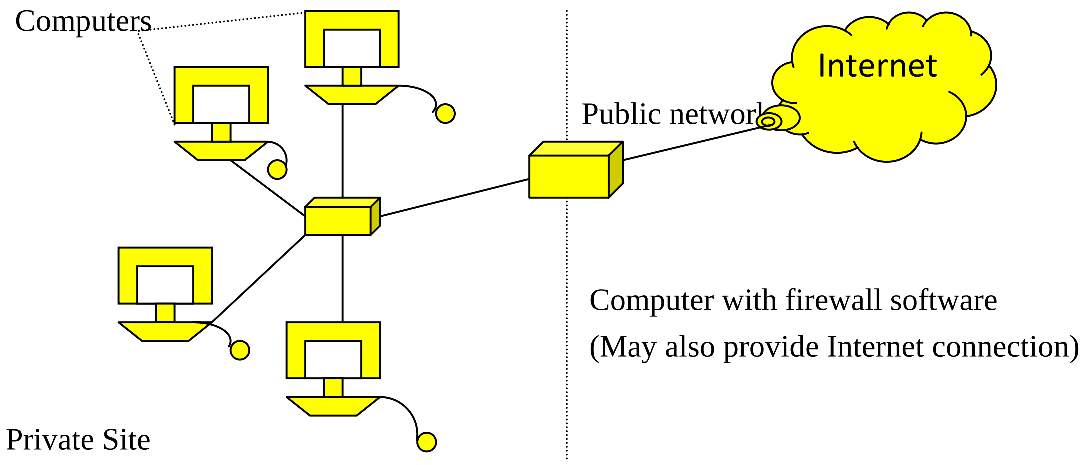

Computers
Internet
Public network
Computer with firewall software
(May also provide Internet connection)
Private Site

**Figure (2-2) Computer with Firewall software.**

A firewall is placed between the site and the rest of the Internet; as shown in figure (3) [23].

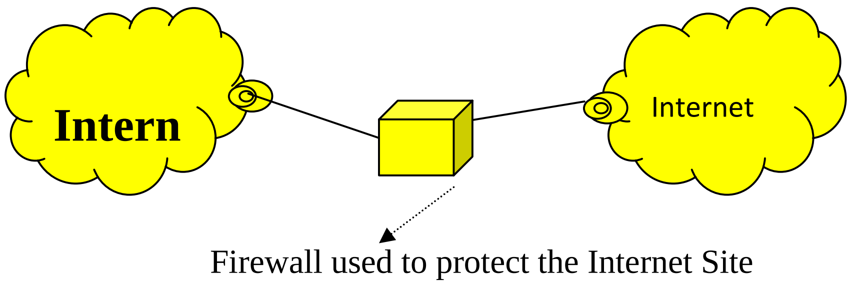


**Figure (3) Position of the firewall used to protect the Internet site.**

The term **firewall** is derived from the fireproof physical boundary placed between two structures to prevent fire from moving between them [23]. If the site has multiple Internet connections, a firewall must be placed at each, and all the site's firewalls must be configured to enforce the site's security policy [23].

## <u>3. Firewall Concept, Conditions,</u>

- ***Firewall Concept:***

All traffic from inside to outside, and vice versa through the main entry, must pass through the firewall. This is achieved by physically blocking all access to the site except via the firewall. Various configurations are possible.

Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies [50,34]. In general Firewall is one of the famous types of intruder detection [45,59].

Firewall effective against worms because it is much difficult for a worm to authenticate itself to a firewall operating a tight exclusive policy than to a general, widely used system. It is effective against Trojan horse . However, even firewall are not invincible [15,17].

- ***Firewall Conditions:***

There are some conditions must be considered in firewalls the most important conditions are declared in the following points:

1- The firewall must be ease to use, has powerful graphical user Interfaces (GUI), which simplifies the job of installation, configuration, and management [43].

2- The firewall must has high performance, should be fast enough so users do not feel the screening of packets. The volume of data throughput and transmission speed associated with the product should be consistent with the company’s bandwidth to the Internet [43].

3- The firewall must be flexible , should be open enough to accommodate the security policy of the company, as well as to allow for changes in the features. Remember that a security policy should a very seldom change, but security procedures should always be reviewed, especially in light of new Internet and web centric application [43].

# Chapter 3

## Encryption and Authentication

Nearly all modern security mechanisms are based on keeping secrets private to certain individuals. Security systems use encryption to keep secrets, and they use authentication to prove the identity of individuals. These two basic security mechanisms are the foundation upon which nearly all security mechanisms are based.

1-Secret key encryption

2-Hashes and one-way functions

3-Public key encryption

4-Password authentication

5-Challenge/response authentication

6-Sessions

7-Public key authentication

8-Digital signatures

9-Certificates

10-Biometric authentication

* **Data security** is the sciences and study of methods of protecting data in computer and communication systems for unauthorized disclosure and modification.

Or

* **Cipher** is a secret method of writing.

Example: (Caser method) use k=3

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

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Plaintext: Ali

(Encipher process)…………………………. **(P+K) mod 26**

Sol:

A=0

0+3 mod 26 = 3 = D

L=11

11+3 mod 26=14=O

I=8

8+3 mod 26=11= L

Cipher text: DOL

(Decipher process)……………………………. **(C-K) mod 26**

Sol:

D=3

3-3 mod 26=0=A

O=14

14-3mod26=11=L

L=11

11-3mod26=8=I

## *Cryptography*

Suppose a sender wants to send a message to a receiver. Moreover, this sender wants to send the message securely: she wants to make sure an eavesdropper can not read the message.

A message is *plaintext* [sometime called clear text]. The process of disguising a message in such a way as to hide its substance is *encryption*. An encrypted message is *cipher text*. The process of truing cipher text back into plaintext is *decryption*; this is all shown in figure (1).

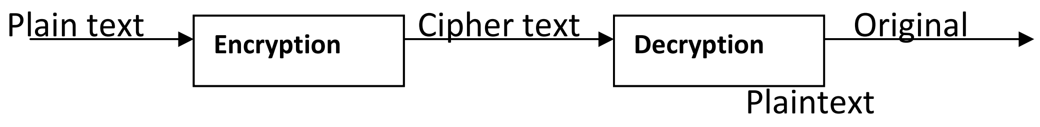


Figure (1): Encryption and Decryption

Plain text is denoted by **M,** for message or **P** for plaintext. It can be a stream of bits, a text file

, a bitmap, a stream of digitized voice, a digital video image. Whatever. As far as a computer is concerned, **M** is simply binary data.

Cipher text is denoted by **C**. it is also binary data: some the same size as **M**. some time larger (by combining encryption with compression. **C** may be smaller than **M**.)

The encryption functions E operate on M to produce C, or, in mathematical notation.

$$E\ (M) = C$$

In the reverse process, the decryption functions D operate on C to produce M:

$$D(C) = M$$

Since the whole point of encryption and then decryption a message is to recover the original plain text, the following identity must hold true.

$$D\ (E\ (M)) = M$$

***- The components of the cryptographic system. (Cryptosystem)***

1) A plain text message space , M.

2) A cipher text message space, C.

3) A key space, K.

4) A family of enciphering transformations, $E_K$: m $\longrightarrow$ c, where k€ K.

5) A family of deciphering transformations, $D_K$: c $\longrightarrow$ m, where k€ K.

Example//

$D_K(E_K(m))=m$

***- Cryptosystem must satisfy three general requirements:***

1) The enciphering and deciphering transformations must be efficient for all keys.

2) The system must be easy to use.

3) The security of the system should depend only on the secrecy of the keys and not on the secrecy of the algorithms E or D.

- **Encryption;**The process of encoding a plain-text message so that it cannot be understood by intermediate parties who do not know

the key to decrypt it. The primary purpose of *encryption* is to keep secrets. It has other uses, but encryption was first used to protect messages so that only the person that knew the trick to decoding a message could read it. Today, encryption allows computers to keep secrets by transforming data to an unintelligible form using a mathematical function. Just like simple arithmetic, encryption functions combine the message and the encryption key to produce an encrypted result. Without knowing the *secret key*,the result makes no sense.

- **secret key;** A key that must be kept secret by all parties because it can be used to both encrypt and decrypt messages. For example, let's say I need to hide the combination to a lock. In this case, the combination (also called the *message* ) is 9-19-69. To keep things simple, I'm

going to add (adding is the *algorithm* ) 25 (which is the key) to each of the numbers to produce the encrypted value: 34-44-94. I can post this value right On the combination lock so I won't forget it because that number won't do anyone who doesn't know how to use it any good. I just need to remember the algorithm, subtract, and the key, 25. The encrypted text is worthless without the key. I can also simply tell my friends what the key and the algorithm are, and they can combine that knowledge with the encrypted data to decode the original combination.

- **Algorithm;** A method expressed in a mathematical form (such as computer code) for performing a specific function or operation.

You may have noticed that in this example I used the opposite mathematical operation to decode the encrypted text; I added 25 to encode and subtracted 25 to decode. Simple arithmetic algorithms are called

- *symmetrical algorithms* because the algorithm used to encode can be reversed in order to decode the data. Since most mathematical operations can be easily reversed, symmetrical algorithms are common.

- **symmetrical algorithm**

An algorithm that uses the same secret key for encryption and for decryption. Although this example may seem simplistic, it is exactly what happens with modern secret-key *cryptosystems.* The only differences are in the complexity of the algorithm and the length of the key. This example, despite its simplicity, shows exactly how all symmetric encryption systems work. Here is another example, using a slightly more complex key. Notice how the key is repeated as many times as necessary to encode the entire message.

The most common use for encryption with computers is to protect communications between users and communications devices. This use of encryption is an extension of the role codes and *ciphers* have played throughout history. The only difference is that, instead of a human being laboriously converting messages to and from an encoded form, the computer does all the hard work.

- **Cipher :**An algorithm specifically used for encryption. Encryption isn't just for communication. It can also be used to protect data in storage, such as data on a hard drive. Most modern operating systems like Unix or Windows are configured to allow only authorized users to access files while the operating system is running, but when you turn your computer off, all those security features go away and your data is left defenseless. An intruder could load another operating system on the computer or even remove the hard drive and place it in another computer that does not respect the security settings of the original computer, and your data would be accessible. Encryption solves this problem by ensuring that the data is unintelligible if the correct key isn't provided, irrespective of whether the computer is still running in order to protect the data.

**1- Secret Key Encryption**

**secret key encryption ;**Encryption by means of a secret key. Our example in the last section was an example of *secret key encryption*. In secret
key encryption, the same key is used to both encode and decode the message, so it is said to be symmetrical—because both keys are the same. Secret key encryption requires that both parties know the algorithm and the

key in order to decode the message. Until the development of *public key encryption* by cryptographers in the 1970s, secret key encryption was the only type of encryption available.

## *Encryption Algorithms*

- **Traditional Systems Cipher**
- ***Introduction***

Traditional systems related to all ciphers used before and they are divided in two:

1) Transposition.
2) Substitution.

1) **Transposition**: means a management of letters according to some schema. i.e. rearrange bits or characters in the data.

$$\text{Plaintext} \xrightarrow{\text{written}} \text{figure} \xrightarrow{\text{take off}} \text{cipher text}$$

((Transposition))

a) The plaintext was written into the figure according to some (written) path.
b) The cipher text was taken off the figure according to some (take off) path.
c) The key consisted of the figure together with the written in & take off path.

- ***Transposition Types:***

***1) Simple transposition (colummer transposition):***

- The plaintext can be written into a matrix by rows the cipher text can obtain by taking the columns in some order.

***Example 1*** :

CRYPTOGRAPHY = plain text

3 1 4 2 = key

(Encipherment process)

```
3    1    4    2
C    R    Y    P
T    O    G    R   + key  ➔  ROP  PRY  CTA  YGH
A    P    H    Y              1    2    3    4
```

(Decipherment process)

key = 3 1 4 2

cipher text = ROP PRY CTA YGH

```
              1    2    3    4
3    1    4    2
C    R    Y    P
T    O    G    R   ➔  CRYPTOGRAPHY
A    P    H    Y
```

***Example 2:***

Plaintext = this is transposition

Key = code

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

Code=1 4 2 3

(Encipherment process)

```
1    4    2    3
t    h    i    s
i    s    t    r
a    n    s    p
```

o s i t

i o n ***x***

cipher text = tiaoi itsin srptx hsnso

(Decipherment process)

cipher text = tiaoi itsin srptx hsnso

1 2 3 4

key= code = 1 4 2 3

1 4 2 3

t $\xrightarrow{\text{h i s}}$

i s t r = this is transposition

a n s p

o s i t

i o n ***X***

***2) A fixed period d with a permutation function.***

F: Zd➔Zd

Example :

Plain text: CRYPTOGRAPHY , d=4 , f=2 4 1 3

(Encipherment process)

d=4 d=4 d=4

CRYP TOGR APHY

1 2 3 4 1 2 3 4 1 2 3 4

Cipher text :RPCY ORTG PYAH

(Decipherment process)

Cipher text :RPCY ORTG PYAH , d=4, f= 2413

2 4 1 3  2413    2413

Plaintext :  CRYP   TOGR  APHY

1 2 3 4   1 2 3 4   1 2 3 4

***Note: d represent block length***

## 2- Substitution Ciphers

Four types:

a) simple
b) homophonic
c) polyalphabetic
d) polygram

***a) Simple Substitution***

- replace each plaintext letter with a corresponding cipher text letter.

1- Keyword

Example:

Keyword= CRYPTOGRAPHIC SYSTEM

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

To encipher the plaintext=  CRYPTOGRAPHY

ciphertext=YKXFNDGKCFAX

2- Shift alphabets

F(a)=(a+k)mod n       encipher

M=(c-k)mod n           decipher

i.e.  c=(p+k)mod 26,      where a =0,…………..z=25

- this is called shifted alphabey by k position
- example ( caeser cipher use k=3)

P= a b c d…….z

C=  d e f  g……c

Multiplication

F(a)=(a.k)mod n

Or

C=(P.k)mod 26 , where GCD(k,26)=1

Example:

K=9

P: A B C D E F G H I J K ………….Z

C: A J S B K T C L U D……………..R

**b) *homophonic substitution ciphers:***

- a *homophonic substitution* cipher maps each plaintext letter into a set of ciphertext elements( called homophones )

Example:

| A | B | C | D . . . | M | N . . . . | Z |
|---|---|---|---|---|---|---|
| 3 | 17 | 60 | 4 . . . | 71 | 11 . . . . | 31 |
| 25 | 55 | 80 | | 26 | 50 | 83 |
| 31 | 90 | | | | | |
| 36 | | | | | | |
| 79 | | | | | | |

- notice : no. of homophones assigned to each plain text letter on proportional to its frequency.

- *Beal ciphers*:

I use the first letter of each word from a plain text ( ex : a book)

II number the words rx: 1 ,2,3,4,…, text length as words.

III to encipher a plain text a letter use its no, from II.

- Example

01 02 03 04 05 06 07

- The plain text a book : when , in the course of human event ….

- To encipher: THE = 03 06 07

- *Higher Order HomophonicAlgorithm*

1- matrix n ×n
2- fill the matrix with numbers 1 ➔ $n^2$
3- encipher the plaintext with a long with a dummy message x

M= m1, m2, ….

X= x1, x2,….

Ci= K[m1 xi], i= 1,2, ……

Example

Let n=5 , plain alphabet { E, I, L , M, S}

|   | E | I | L | M | S |
|---|---|---|---|---|---|
| E | 10 | 22 | 18 | 02 | 11 |
| I | 12 | 01 | 25 | 05 | 20 |
| L | 19 | 06 | 23 | 13 | 07 |
| M | 03 | 16 | 08 | 24 | 15 |
| S | 17 | 09 | 21 | 15 | 04 |

M= S M I L E

X= L I M E S (DUMMY MESSAGE)

C= 21 16 05 19 11

## 2- public key encryption

Encryption by means of a public key; an encryption methodology that allows the distribution of an encryption key that does not compromise the secrecy of the decrypting private key due to the utilization of a related pair of one-way functions. Secret key encryption works well for keeping secrets, but both parties have to know the same secret key in order to decode the message. There's no secure way to transfer the key from one party to the other without going to extraordinary lengths, like having both parties meet in the same secluded area to exchange keys. There's certainly no way to exchange keys over an electronic medium without the possibility of a wiretap intercepting the key.

- **One-Way Functions (Hashes)**

Hashes are used to verify the correctness of information and are based on mathematical algorithms called one-way functions. Some mathematical functions cannot be reversed to retrieve the original number. For example, let's say that we're going to divide 46,835,345 by 26,585. This results in 1,761 with a remainder of 19,160. So let's say that we have an algorithm that simply returns the remainder (19,160) and discards the quotient (1,761). Now, if we have just the remainder (called a modulus) and one of the original numbers, there's no way to reconstruct the other operand because the quotient has been discarded. The remainder alone does not retain enough information to reconstruct the original number.

- **Plaintext ;**An unencrypted text, either before it has

been encrypted or after it has been decrypted. The encrypted version is referred to as a ciphertext. The following illustration shows an extremely simple hash. In this algorithm, two numbers are added, and if the result is even, a binary 1 is the result. If the result is odd, a 0 is the result. The combination of binary digits forms the hash. Because the actual numbers are lost, the hash cannot be reconstructed, but it can be used to determine with

reasonable certainty whether or not two plaintexts match. Simple hashes are often called *checksums.* This hashing algorithm is not appropriate for security because there are many other plaintexts that will produce
the same result, so the probability of an accidental match is too risky.

- **one-way function**

An algorithm that has no reciprocal function and cannot therefore be
reversed in order to discover the data originally encoded. Of course, since we can't reconstruct the original number, we can't use *oneway functions*
to encode and decode information. But we can use them to be certain that two people know the same number without revealing what that number is. If two people were aware of the original dividend (46,835,345) in the previous scenario and you told them to divide the number they knew by 26,585, discard the whole product, and tell you the remainder, they would both
report 19,160—thus proving that they knew the same original number or that they were amazingly lucky, because there is only a 1 in 26,585 chance that they could have guessed the correct remainder. By simply making the number large enough that the odds of guessing the correct remainder are impossibly high, you can use this one-way nonreversible function to prove that two parties know a number without ever revealing what that number actually is to anyone who might overhear you. For example, let's say a system requires logon authentication in the form of a user providing their name and password. They could simply enter their name and password and the computer could check that against a list of stored passwords. If the passwords matched, the user would be allowed in. But let's say that hackers
gained access to the computer. They could steal the list and then have everyone's password. Once other people know a user's password, you can no longer hold

that user accountable for their actions because you can’t guarantee that they actually performed those actions on a system.
We can’t use secret key encryption to encrypt the password file because the secret key would need to be stored on the system in order for the system to decode the password file. Because we’ve already stipulated that hackers have access to the system, they would also have access to the secret key and could use that to decrypt the password file.

- **password**

A secret key that is remembered by humans.
This is a situation in which one-way functions work well. Rather than storing the user’s *password*, we can store a *hash,*or the result of a one-way function, instead. Then, when they enter their password, we perform the one-way function on the data they entered and compare it to the hash that’s stored on disk. Because only the hash is stored and the hashing algorithm can’t be reversed to find the original password, hackers can’t compromise the authentication system by stealing this list of password hashes.

- **hash**

The result of applying a one-way function to a value. Hashes allow a user to prove that they know the password without the system having to know what the password actually is. Protecting passwords is the most common use for using hashing algorithms in computer security.

## Public Key Encryption

- **asymmetrical algorithm**

A mathematical function that has no reciprocal function. Whereas symmetric ciphers like secret key algorithms use the same key to encrypt and decrypt messages, public key encryption uses one key to encrypt a message and a different key to decrypt it, so they are called *asymmetric algorithms*. In a public key encryption system, the encryption key is called the public key because it can be made public. The decryption key is called the *private key* because it must be kept private in order to remain secure.

- **private key;** A secretly held key for an asymmetrical encryption algorithm that can be used only to decode messages or encode

digital signatures. The problem with secret key encryption is this: Both the sender and the recipient must have the same key in order to exchange encrypted messages over an  non secure medium. If two parties decide to exchange private messages, or if two computers' network devices or programs must establish a secure channel, the two parties must decide on a common key. Either party may simply decide on a key, but that party will have no way to send it to the other without the risk of it being intercepted on its way. It's a chicken-and-egg problem: Without a secure channel, there is no way to establish a secure channel.

- **public key;** A publicly distributed key for an asymmetrical encryption algorithm; a public key can be used only to encode messages or decode digital signatures. With public key encryption, the receiver can send a public encryption key to the sender. There is no need to keep the *public key* a secret because it can be used only to encode messages, not decode them. You can publish your public key to

the world for anyone to use for encoding message they send to you.
When the receiver gets a message that has been encoded with their public key, they can use their private key to decode the message. Revealing their public key to the world for encoding does not allow anyone else to decode their private messages.

- **Cryptography;** The study of codes, ciphers, and encryption.

Public key cryptography is a relatively new development in *cryptography,* One that solves many long-standing problems with cryptographic systems—especially the chicken-and-egg conundrum of how to exchange secret keys. In 1976, Witfield Diffie and Martin Hellman figured out a way out of the secure channel dilemma. They found that some one-way functions could be undone by using a different key for decryption than was used for encryption. Their solution (called public key cryptography) takes advantage of a characteristic of prime and almost prime numbers, specifically, how hard it is to find the two factors of a large number that has only two factors, both of which are prime. Since Diffie and Hellman developed their system, some other public key ciphers have been introduced using even more difficult one-way functions. One problem that plagues secure public key ciphers is that they are slow—much slower than symmetric ciphers. You can expect a good public key cipher to take 1,000 times as long to encrypt the same amount of data as a good symmetric cipher would take. This can be quite a drag on your computer's performance if you have a lot of data to transmit or receive.

- **Hybrid Cryptosystems:**Although it is much slower than symmetric systems, the public key/private key system neatly solves the problem that bedevils symmetric cryptosystems—exchanging secret keys.

**hybrid cryptosystem ;**A cryptosystem that exchanges secret keys using public key encryption to secure the key exchange and then using the higher speed allowed by secret key encryption to transmit subsequent data.

But there's no need to give up the speed of secret key cryptosystems just because secret keys can't be exchanged securely.

*Hybrid cryptosystems* use public key encryption to exchange secret keys and then use the secret keys to establish a communication channel. Nearly all modern cryptosystems work this way. When two people (or devices) need to

establish a secure channel for communication, one of them can generate a random secret key and then encrypt that secret key using the receiver’s public key. The encrypted key is then sent to the receiver. Even if the key is intercepted, only the intended recipient, by using their
private key, can decrypt the message containing the secret key.
Once both parties have the secret key, they can begin using a much faster
secret key cryptosystem to exchange secret messages.

- **Authentication**

**Authentication:** The process of determining a user’s
identity in order to allow access. *Authentication* is used to verify the identity of users to control access to resources, to prevent unauthorized users from gaining access to the system, and to record the activities of users in order to hold them accountable for their activities. Authentication is used to verify the identity of users logging on to computers, it’s used to ensure that software you download from the Internet comes from a reputable source, and it’s used to ensure that the person who sends a message is really who they say they are.

**Password Authentication:**Passwords are the oldest and simplest form of authentication—they’ve been used since time immemorial to prove that an individual should be given access to some resource. Technically, passwords are secret keys. Password authentication is simple: By knowing a secret key, you could prove that the individual who invented the secret trusted you with that secret key and that the trust should be bestowed upon you. This sort of password authentication proves only that access to a resource should be allowed—it does not prove identity.
To prove identity, a password must be unique to a specific person. In secure

computer systems, this is accomplished by creating user accounts, which are assigned to individuals. The account contains information about who the owner is and includes a unique account name and password.
When a user logs on to the system, they simply type in their account name to assert their identity, and then they provide the password associated with that user account to prove that they are allowed to use that account. If the entered password matches the stored password, the user is allowed access to the system.Password authentication can fail in a number of ways:

1. **brute-force attack;** An attack in which every possible combination of values is tried in sequence against a password system.

Given an unlimited amount of time, these attacks will always succeed, but they are impractical against long passwords, which could require more time than the age of the universe to crack.
There's no way to control password distribution. If the account holder
loses control of the password, it can be distributed to anyone. Once a password has been compromised, password authentication can no longer be used to prove identity.
Passwords are often simple and easy to guess, and many systems limit passwords to lengths that lend themselves to *brute-force*
guessing—that is, simply trying all possible combinations. This can lead to password compromise and is one of the most common ways that hackers gain access to systems.

2. **replay attack:** An attack in which a secret value like a hash is captured and then reused at a later time to gain access to a system

without ever decrypting or decoding the hash. Replay attacks only work against systems that don't uniquely encrypt hashes for each session.
Naïve implementations may not protect the password in transit or may be

compromised through simple techniques like *replay*.

Despite the common failure of password-based systems to actually prove identity and restrict access, they are by a wide margin the most common way tosecure computers.

3. **Password Hashing;**To prevent hackers from capturing your password from your computer's hard disk or while it transits the network, passwords can be encrypted using a one-way function or hashing algorithm to keep them from being revealed. In most modern operating systems, the operating system does not compare your

password to a stored password. Instead, it encrypts your password using a one way cryptographic function and then compares the result to the original result that was stored when you created your password. Because the hashing function is one way, it cannot be reversed to decrypt your password.

However, password hashing is susceptible to brute-force indexed decryption. Using this technique, hackers create a "dictionary" of all possible passwords by encrypting every possible password (i.e., AAAAAA through ZZZZZZ in a system limited to six letters) using the same hashing function as the password system and storing the results in a database along with the original text. Then, by capturing your hash, they can look up the matching value in their database of hashed values and find the original password. Although compiling this sort of dictionary can take a long time for a single computer, it is the sort of problem that could be easily distributed over a network of computers (like the Internet) to speed up completion. Once finished, the hashing algorithm would be all but worthless for encrypting

passwords and a new algorithm would be required to maintain password security.

# Chapter -4

**Understanding Viruses**

**Macro ;**A list of instructions stored as data that
is interpreted by a scripting host. To combat viruses effectively, you need to understand how they propagate and what defenses are available. Computers store two entirely different types of information:
***executable****code* and *data.*
Code specifies how the computer should operate, while data
provides the information on which the operation is performed. For example, in the equation 1+1=2, the 1, 1, and 2 are the data and the + and = are code. The difference between code and data is crucial to virus defense because only code is susceptible to virus infection. Viruses can corrupt data but cannot propagate using pure data because data does not provide an execution environment for viruses to run in.

**interpreter**

A programming language application that loads scripts as data and then interprets commands step-by-step rather than by compiling them to machine language. But it's not always clear what is data and what is code. Is a Word document code or data? It's mostly data, but because Word documents can contains *macros* that Word interprets in order to perform complex operations, it can also contain code. The same goes for any macro-enabled application that stores a mixture of code and data in a single document. Applications that look at data and then perform wide-ranging operations based on that data are called *execution environments, interpreters,*
Or *scripting hosts.*

**scripting hosts**

Execution environments that can be called from applications in order to execute scripts contained in the application's data. You could consider any use of data to be interpretation, but for viruses to propagate, the execution environment needs to be complicated enough to allow for self-replication. This is a very high-level function that is typically only available when a scripting host running an interpreted computer language is built into
an application. What does all this really mean? Simple. You don't have to worry about viruses in pictures, audio files, online movies, and other programs that merely edit or display content, but you do need to worry about programs that use content to control the program's behavior. If the control mechanisms are complex enough to allow self-replication to occur, the application could host viruses. When you search the Web, you'll often see the term *virii* used as the plural for virus.Since *virus* comes from Greek and not Latin, its correct plural is *viruses* , not *virii.* But most hackers don't know that, so they use the term "virii" for the plural.

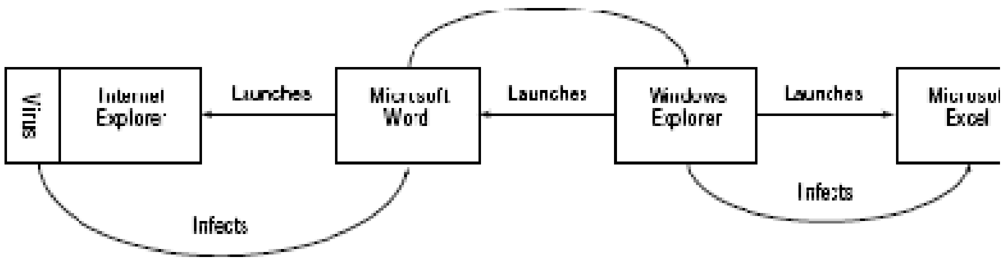
**Understanding Virus Propagation**

When you launch a program on any computer, you are directing an existing running program to launch it for you. In Windows, this program is (usually) Windows Explorer. It could technically be many other programs, like the command prompt or Internet Explorer (if you download the program and choose to open it from its current location). On a Macintosh it's usually the Finder, and in Unix it's the command shell or the X Windows manager. The important concept is that every program is started by another program in an unbroken chain all the way back to the initial bootstrap code stored permanently in the computer's motherboard.

So how do viruses wedge themselves into this process? How do they know which programs will spread them and which will simply cause them to lie dormant? Usually, they don't. Viruses only know the program that was running when they were first executed (like Internet Explorer) and the program that started that program. When a virus is first executed, it attaches itself to the beginning of the current program. The next time that program is run, the virus takes the information about the program that started the current program to determine what file it should infect next and attaches itself to the beginning of that program. In this way, the virus propagates one step closer to the beginning of the chain each time the programs in the startup chain are launched.

**Worms**

Viruses that spread over a network automatically without human intervention. Viruses also attach themselves to each program that the current program launches. In most cases this is nothing—most applications can't launch other programs. But some do, and when those applications are found, the virus automatically spreads. The graphic that follows shows how a virus attached to Internet Explorer can propagate back to the program that launched it (Word), then back to Windows Explorer, and from there to other applications.



It’s important to note that viruses require human activity—booting a floppy, executing a program, or opening an attachment—in order to activate and spread. Viruses that can spread without human activity are referred to as *worms*. Worms typically exploit buffer overrun attacks against common Internet services like mail or web.

**Common Types of Virus Attacks**

Types of viruses are defined mostly by the propagation method they use. In many cases, an entire class of viruses is composed of permutations of just a single virus,so they’re nearly equivalent. Viruses are categorized by their ultimate target, asdescribed in the following sections.

**Boot Sector Viruses**

**boot sector**

The first executable code that is stored on a disk and is used to load the operating system.*Boot sector* viruses were the original viruses, and they spread by the only common means of sharing information in the early days of computers—on floppy disks. Twenty years ago, networks were uncommon. Most data was shared by copying it to floppy disks. It was common at that time to boot floppy disks for special purposes
like playing games or simply because the floppy had been left in the drive when the computer was turned off. Boot sector viruses would copy themselves to the boot sector of the host when the floppy was booted and then subsequently infect every floppy that was inserted into the computer.
Thanks to the proliferation of networks, these viruses are practically extinct.

**Executable Viruses**

**shell**

A command-line interface to an operating system. Executable viruses infect the startup code for programs and propagate back to the *shell* or desktop application of the computer in order to infect all programs launched from it. Because of the native immunity to this activity in modern permissionsbased operating systems, these viruses have become rare, except in places where older operating systems are common.

**Macro Viruses**

**macro virus:** Viruses that exist in the interpreted code embedded in Office documents. These viruses are not capable of escaping the confines of their interpreted environment, so they cannot infect executables. *Macro viruses* are written in a higher-level language, such as the Visual Basic scripting language built into Microsoft Office, so they are related to other interpreted language viruses like those that can infest Java applications. Macro viruses attach themselves to document templates and can spread to other documents that are saved after opening the infected one. They spread like wildfire through corporate networks where users share documents indiscriminately. Luckily, most Office document macro viruses are relatively harmless, and Microsoft has worked to close the security holes in the Office macro languages (the most common macro viruses are specific to Word and Excel). The latest version of Office ships with immunity enabled by default, so macro viruses will become obsolete when this software becomes widespread.

**E-Mail Viruses**

Unfortunately, the same application language has been built into Microsoft Outlook, the e-mail software that comes with Office (and its free-with-the-operating system sibling, Outlook Express). Viruses written for Outlook can automatically read your list of contacts from the Address Book or Contacts folder and mail themselves to everyone you know, thus propagating

extremely rapidly. Currently, Outlook e-mail viruses are by far the fastest spreading and largest threat in the
virus world.

**Understanding Worms and Trojan Horses**

Worms are viruses that spread automatically, irrespective of human behavior, by exploiting bugs in applications that are connected to the Internet. You've probably heard the names of the most widely successful ones in the mainstream media: Code Red, Nimda, and Slammer. From an infected machine, the worm scans the network searching for targets. It then contacts the target, initiates a benign exchange, exploits a bug in the receiver's server software to gain control of the server momentarily, and uploads itself to the target. Once the target is infected, the process starts again on it. Worms usually carry a Trojan horse along with them as payload and set up
a listening service on the computer for hackers to connect to. Once a worm is in the wild, hackers will begin port scanning wide ranges of computers looking for the port opened up by the worm's listening service. When a hacker (let's call him Sam) finds a compromised computer, he will typically create an administrative account for himself and then clean up the worm and patch the computer against further exploits—to keep other hackers out so that he can reliably use the computer in the future. The computer is now "owned" by Sam and has become his "zombie," in hacker terms. Because this all happens behind the scenes (and often at night), the real owner of the computer often never knows. But like a parasitic symbiote, people who have been "owned" are sometimes better off having a knowledgeable hacker protecting their zombie from further attacks. Your computer has probably already been hacked if you have a broadband Internet connection and you

don’t have a cable/DSL router or a software firewall. It wouldn’t show up on a virus scan because the hacker would have cleaned up the worm within a few hours of infection. To take back ownership of your computer, change the passwords on every account on the machine. Hackers like to collect from a few dozen up to (in some cases) a few thousand zombies so that they can perpetrate attacks from many different IP addresses on the Internet. Some hackers actually sell (using auction sites, believe it or not) large banks of zombies to spammers who use them to transmit bulk spam. Anti-hacking researchers leave unprotected computers out on the Internet to allow them to be exploited so that they can track down hackers by watching the activity on the exploited computers, so hackers will typically “bounce” through multiple zombies before perpetrating an attack to throw investigators off their trail. This is going on all around you on the Internet, right now. Worms are basically impossible for end users to prevent, and they typically exploit newly found bugs that are either unpatched or not widely patched in a vendor’s code. When they attack extremely common systems like Windows or Linux, they spread very quickly and can cause enormous damage before they’re stopped. Here are some suggestions to defend against worms:

Avoid software that is routinely compromised, like Microsoft Internet Information Server and Internet Explorer. (Mozilla, a free download From www.mozilla.org is an excellent replacement for IE on Windows computers.)

Stay up-to-date on patches and security fixes for all your public computers. Strongly consider using automatic updates for any public server, and schedule them for a nightly reboot to make sure that patches become effective as quickly as possible.

Keep client computers behind firewalls or cable/DSL routers.

Run only those services you intend to provide on public servers—don't just install everything for the sake of convenience when you set up a public server. Use firewalls to prevent worms from reaching the interior of your network from the Internet. Keep your virus-scanning software updated.

But even with all these precautions, you can only be protected against worms that the vendors know about, and it's quite likely that a worm will infest your public servers at some point, so keep good backups as well.

## Protecting Against Worms

There are two common ways to protect against worms. Firewalling services that you don't use is the primary method. However, some services (like web and e-mail) must be open to the Internet and usually cannot be protected against by a firewall. In this case, using software specifically designed to filter the protocol—such as a proxy-based firewall, a supplemental security service like e-eye Secure IIS, or simple URL filtering on the characters used by hackers to insert buffer overruns— can stop the attacks before they get to the firewall. For mail servers, simply putting a mail proxy server from a different operating system in front of your actual mail server will prevent the interior mail server from being affected by any buffer overrun that can affect the proxy. Finally, virus scanners receive signatures that allow them to recognize and (sometimes) clean worms that have already infected a server. In cases where the virus scanner cannot automatically clean up the worm, antivirus software vendors will provide a downloadable tool that will clean up the infection. Unfortunately, this method doesn't stop worm infection; it merely removes it.