Introduction: Security Basics – Aspects of network security – Attacks – Different Types – Hackers – Crackers – Common Intrusion Techniques - Trojan Horse – Virus – Worms – Security Services and Mechanism

Introduction

The information security has undergone two major changes with the evolution of computers. The need for automated tools for protecting information stored on the computer became evident. The collection of tools designed to protect data and to thwart hackers is computer security**.** The introduction of distributed systems and the use of networks and communications facilities for carrying data between terminal user and computer and between computer and computer leaded to network security.

Aspects of network security

The aspects of network security includes

- Security attack
- Security mechanism
- Security service

Security Attack

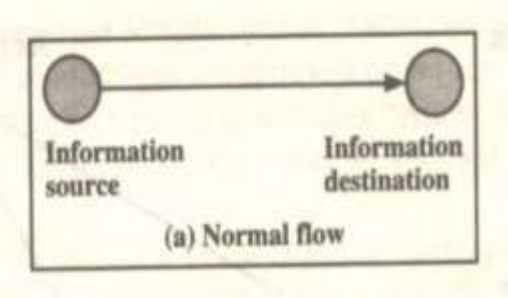
Security is essential when the function of the computer system is to provide information. The normal flow of information from source to destination is shown in figure.
There are four general categories of attack.

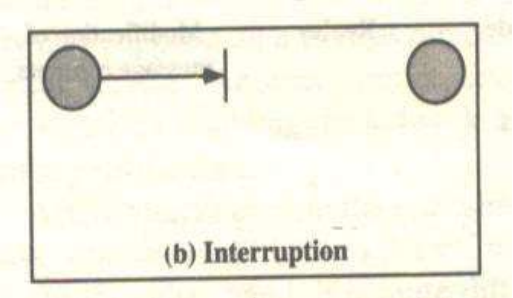
1) Interruption
2) Interception
3) Modification
4) Fabrication

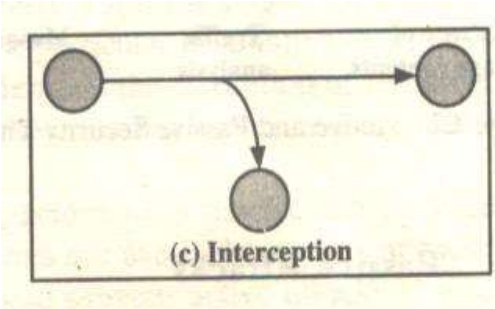
- Interruption

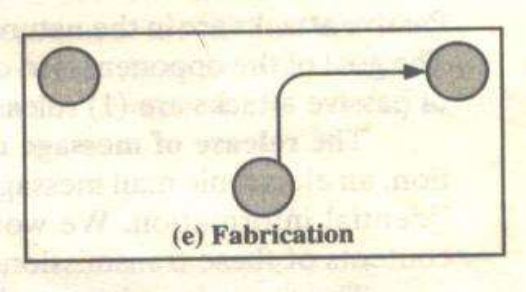
An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on availability. Examples include destruction of a piece of hardware, such as a hard disk, the cutting of a communication line, or the disabling of the file management system.

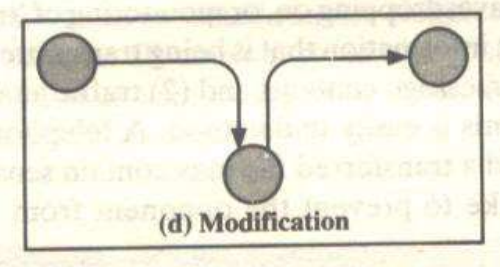
Interception:











Interception:

An unauthorized party gains access to an asset. This is an attack on confidentiality. The unauthorized party could be a person, a program, or a computer. Examples include wiretapping to capture data in a network, and the unauthorized copying of files or programs.

Modification:

An unauthorized party not only gains access to but tampers with an asset. This is an attack on integrity. Examples include changing values in a data file, altering a program so that it performs differently, and modifying the content of messages being transmitted in a network.

Fabrication:

An unauthorized party inserts counterfeit objects into the system. This is an attack on authenticity. Examples include the insertion of spurious messages in a network or the addition of records to a file.

Attack is also categorized as

1) Active attacks

2) Passive attacks

Active attacks

These attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories

- Masquerade
- Replay
- Modification of message contents
- Denial of service

Masquerade

A masquerade takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.

Replay

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.

Modification of message contents

Modification of messages means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect. For example, a message meaning ―Allow John Smith to read confidential file *accounts"* is modified to mean ―Allow Fred Brown to read confidential file *accounts."*

Denial of service

The denial of service prevents or inhibits the normal use or management of communication facilities. This attack may have a specific target; for

example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

Passive Attacks

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are

- Release of message contents
- Traffic analysis.

Release of message contents

A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent the opponent from learning the contents of these transmissions

Traffic analysis:

Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of

messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. However, it is feasible to prevent the success of these attacks. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

Active attacks	**Passive attacks**
Difficult to prevent	Easy to prevent
Easy to detect	Difficult to detect

Security Services

• Confidentiality

• Authentication

• Integrity

• Non repudiation

• Access control

• Availability

Confidentiality

Confidentiality is the protection of transmitted data from passive attacks. With respect to the release of message contents, several levels of protection can be identified. The broadest service protects all user data transmitted between two users over a period of time. For example, if a virtual circuit is set up between two systems, this broad protection would prevent the release of any user data transmitted over the virtual circuit. Narrower forms of this service can also be defined, including the protection of a single message or even specific fields within a message. These refinements are more complex and expensive to implement.

The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility.

Authentication

The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved. First, at the time of connection initiation, the service assures that the two entities are authentic. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception.

Integrity

As with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields within a message. Again, the most useful and straightforward approach is total stream protection.

A connection-oriented integrity service, one that deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering, or replays. The destruction of data is also covered under this service. Thus, the connection-oriented integrity service addresses both message stream modification and denial of service. On the other hand, a connectionless integrity service, one that deals with individual messages only without regard to any larger context, generally provides protection against message modification only.

We can make a distinction between the service with and without recovery. Because the integrity service relates to active attacks, we are concerned with detection rather than prevention. If a violation of integrity is detected, then the service may simply report this violation, and some other portion of software or human intervention is required to recover from the violation. Alternatively, there are mechanisms available to recover from the loss of integrity of data, as we will review subsequently. The incorporation of automated recovery mechanisms is, in general, the more attractive alternative.

Non repudiation

Non repudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the message was in fact sent by the alleged sender. Similarly, when a message is received, the sender can prove that the message was in fact received by the alleged receiver.

Access Control

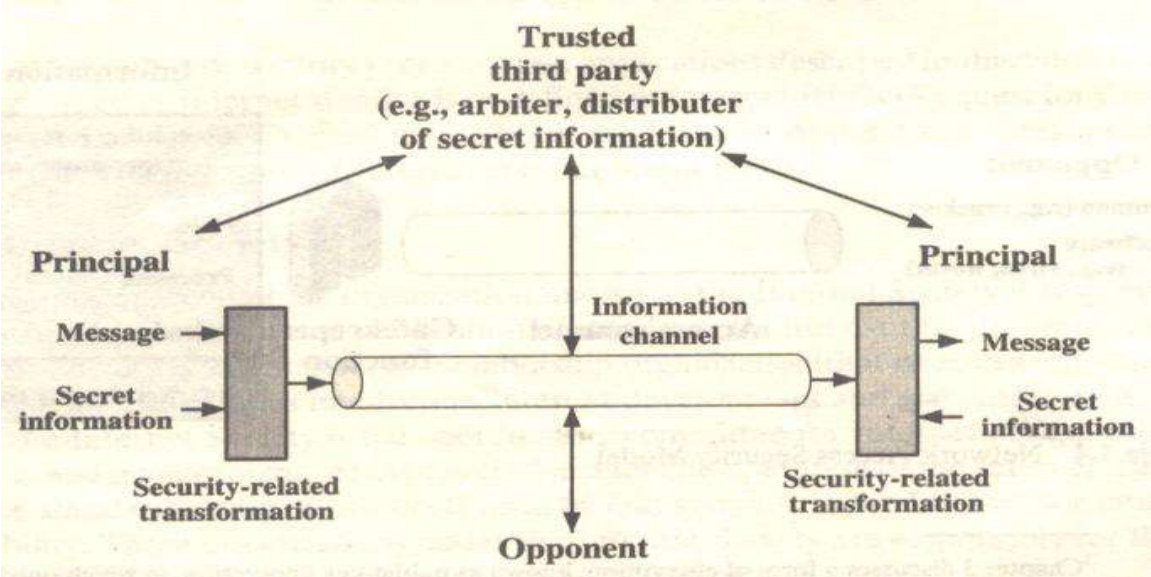
In the context of network security, access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this control, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual.

Availability

A variety of attacks can result in the loss of or reduction in availability. Some of these attacks are amenable **to** automated countermeasures, such as authentication and encryption, whereas others require some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

Mechanisms

A mechanism that is designed to detect, prevent or recover from security attack.



A MODEL FOR NETWORK SECURITY

A message is to be transferred from one party to another across some sort of internet. The two parties, who are the *principals* in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals. Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All the techniques for providing security have two components:

1.A security-related transformation on the information to be sent. Examples includes the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender.

2. Some secret information shared by the two principals which is unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission. This general model

shows that there are four basic tasks in designing a particular security service:

- Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.

- Generate the secret information to be used with the algorithm.

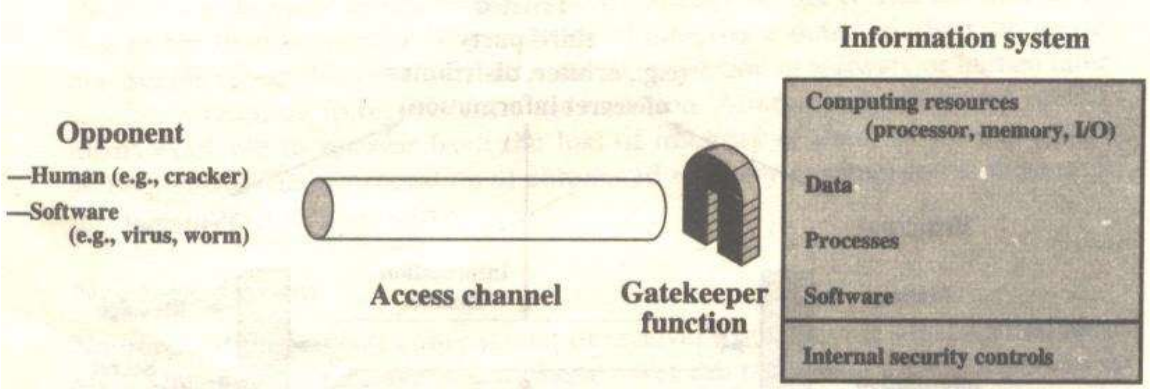
- Develop methods for the distribution and sharing of the secret information.

- Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

A general model is shown in fig. which reflects a concern for protecting an information system from unwanted access. Hacker is a person who attempt to penetrate systems that can be accessed over a network. The hacker can be someone who, simply gets satisfaction from breaking and entering a computer system. The intruder can be a disgruntled employee who wishes to do damage, or a criminal who seeks to exploit computer assets for financial gain (e.g., obtaining credit card numbers or performing illegal money transfers).

Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect

application programs as well as utility programs, such as editors and compilers. Two kinds of threats can be presented by programs:



Information access threats:- intercept or modify data on behalf of users who should not have access to that data.

- Service threats:- exploit service flaws in computers to inhibit use by legitimate users.

Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a diskette that contains the unwanted logic concealed in otherwise useful software. They can also be inserted into a system across a network. The security mechanisms needed to cope with unwanted access fall into two broad categories.

- Gatekeeper function:- It includes password-based login procedures that are designed to deny access to all but authorized users and screening logic that is designed to detect and reject worms, viruses, and other similar attacks.

- Line of defense:- consists of a variety of internal controls that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders.

Hackers

Hacker is a person who enjoys exploring the details of programmable systems. A hacker is a person who will have a very good knowledge about computers and utilize the effectively to find the defects in an application or network. They will have very good programming skills. They never do any mischief to earn money in a wrong way by may be paid for their expertise. They should be willing to share their knowledge and methods to others. He must know his abilities before something could cause harm to the system. They should not distribute, use or collect pirated software. After the system is hacked the system administrator should be notified about any security breaches.

Hacker is a person who builds the system and cracker is a person who breaks it. The basic hacking skills include

- Learning how to program
- Getting one of the open source program and learn how to run it
- Learning how to use World Wide Web to write HTML

Crackers

Software cracking is the process of modification of the software to remove protection methods such as copy prevention, data check etc.

The most common software crack is the modification of an application to prevent a specific key branch in the program‘s execution. This is accomplished by reverse engineering the compiled program code using a

debugger until the software cracker reaches the subroutine that contains the primary method of protecting the software. An example of this technique is a crack that removes the expiration time from a time limited trial of an application.

The most visible effect of software cracking is the releasing of fully operable property software without any copy protection. Cracking has also been an efficient factor in the companies such as Adobe systems who has benefited from privacy.

Common Intrusion Techniques

One of the two most publicized threats to security is the intruder (the other is viruses), generally referred to as a hacker or cracker. In an important early study of intrusion

• **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system‘s access controls to exploit a legitimate user‘s account

• **Misfeasor:** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his r her privileges

• **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection. The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider. Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system.

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Generally, this requires the intruder to acquire information that should have been protected. In most cases, this information is in the form of a user password. With knowledge of some other user‘s password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user. Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it and learn passwords.

The password file can be protected in one of two ways.

- One-way encryption: The system stores only an encrypted form of the user‘s password. When the user presents a password, the system encrypts that password and compares it with the stored value. In practice, the system usually performs a one-way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed-length output is produced.

- Access control: Access to the password file is Limited to one or a very few accounts.

If one or both of these countermeasures are in place, some effort is needed for a potential intruder to learn passwords. On the basis of a survey of the literature and interviews with a number of password crackers, reports the following techniques for learning passwords:

1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.

2. Exhaustively try all short passwords (those of one to three characters).

3. Try words in the system‘s online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards.

4. Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies.

5. Try users‘ phone numbers, Social Security numbers, and room numbers.

6. Try all legitimate license plate numbers for this state.

7. Use a Trojan horse (described in Section 92) to bypass restrictions on access.

8. Tap the line between a remote user and the host system.

The first six methods are various ways of guessing a password. If an intruder has to verify the guess by attempting to log in, it is a tedious and easily countered means of attack. For example, a system can simply reject any login after three password attempts, thus requiring the intruder to reconnect to the host to try again. Under these circumstances, it is not practical to try more than a handful of passwords. However, the intruder is unlikely to try such crude methods. For example, if an intruder can gain

access with a low level of privileges to an encrypted password file, then the strategy would be to capture that file and then use the encryption mechanism of that particular system at leisure until a valid password that provided greater privileges was discovered.

Guessing attacks are feasible, and indeed highly effective, when a large number of guesses can be attempted automatically and each guess verified, without the guessing process being detectable. Later in this section, we have much to say about thwarting guessing attacks.

The seventh method of attack listed earlier, the Trojan horse, can be particularly difficult to counter. A low-privilege user produced a game program and invited the system operator to use it in his or her spare time. The program did indeed play a game, but in the background it also contained code to copy the password file, which was unencrypted but access protected, into the user‘s file. Because the game was running under the operator‘s high-privilege mode, it was able to gain access to the password file

The eighth attack listed, line tapping, is a matter of physical security. Prevention is a challenging security goal and an uphill battle at all times. The difficulty stems from the fact that the defender must attempt to thwart all possible attacks, whereas the attacker is free to try to find the weakest link in the defense chain and attack at that point. Detection is concerned with learning of an attack, either before or after its success.

Password Protection

The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password. The password serves to authenticate the

ID of the individual logging on to the system. In turn, the ID provides security in the following ways

- The ID determines whether the user is authorized to gain access to a system. In some systems, only those who already have an ID filed on the system are allowed to gain access.

- The ID determines the privileges accorded to the user. A few users may have supervisory or ―super user‖ status that enables them to read files and perform functions that are especially protected by the operating system. Some systems have guest or anonymous accounts, and users of these accounts have more limited privileges.

Malicious Programs

Malicious programs threats can be divided into two categories:

• Those that need a host program,
• Those that is independent.

The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. The latter are self-contained programs that can be scheduled and run by the operating system.

We can also differentiate as

• Software threats that do not replicate
• Software threats that replicates

The former are fragments of programs that are to be activated when the host program is invoked to perform a specific function. The latter consist of either a program fragment (virus) or an independent program (worm, bacterium) that, when executed, may produce one or more copies of itself to be activated later on the same system or some other system.

Trap doors

A trap door is a secret entry point into a program that allows someone that is aware of the trap door to gain access without going through the procedures. Trap doors have been used legitimately for many years by programmers to debug and test programs. This usually is done when the programmer is developing an application that has an authentication procedure, or a long setup, requiring the user to enter many different values, to run the application. To debug the program, the developer may wish to gain special privileges or to avoid all the necessary setup and authentication. The programmer may also want to ensure that there is a method of activating the program should something be wrong with the authentication procedure that is being built into the application. The trap door is code that recognizes some special sequence of input or is triggered by being run from a certain user ID or by an unlikely sequence of events.

Trap doors become threats when they are used by unscrupulous programmers to gain unauthorized access. It is difficult to implement operating system controls for trap doors. Security measures must focus on the program development and software update activities.

Logic Bomb

One of the oldest types of program threat, predating viruses and worms, is the logic bomb. The logic bomb is code embedded in some legitimate program that is set to ―explode‖ when certain conditions are met. Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files, a particular day of the week or date, or a particular user running the application.eg: a logic bomb checked for a certain employee ID numb and then triggered if the ID failed to appear in two consecutive payroll calculations. Once triggered, a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage.

Trojan Horses

A Trojan horse is a useful program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function. Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly. For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changed the invoking user‘s file permissions so that the file‘ are readable by any user. The author could then induce users to run the program by placing it in a common directory and naming it such that it appears to be a useful utility. An example is a program that ostensibly produces a listing of the user‘s files in a desirable format. After another user has run the program, the author can then access the information in the user‘s files.

Another common motivation for the Trojan horse is data destruction. The program appears to be performing a useful function (e.g., a calculator program), but it may also be quietly deleting the user‘s files or bulletin board system.

Viruses

A virus is a program that can ―infect‖ other programs by modifying them: the modification includes a copy of the virus program, which can then go on to infect other programs.

A computer virus carries in its instructional code the recipe for making perfect copies of itself. Lodged in a host computer, the typical virus takes temporary control of the computer‘s disk operating system. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users, who either swap disks or send programs to one another over a network. In a network environment, the ability to access applications and system services on other computers provides a perfect culture for the spread of a virus.

The Nature of Viruses

A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs. During its lifetime, a typical virus goes through the following four stages

Dormant phase

The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.

Propagation phase

The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.

Triggering phase

The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

Execution phase

The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files. Most viruses carry out their work in a manner that is specific to a particular operating system and, in some cases, specific to a particular hardware platform. Thus, they are designed to take advantage of the details and weaknesses of particular systems.

Virus Structure

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

```
program V :=
 (gob main;
1234567;
subroutine infect-executable :=
(loop:
file := get-random-executable-file;
 if (first-line-of-file = 1234567)
 then goto loop
else prepend V to file; I
 subroutine do-damage :r
```

```
(whatever damage is to be done)
subroutine trigger-pulled
(return true if some condition holds)
main: main-program :=
(infect-executable;
if trigger-pulled then do-damage;
goto next:)
 next:
```

A very general depiction of virus structure is shown in Figure . In this case, the virus code, V, is prepended to infected programs, and it is assumed that the entry point to the program, when invoked, is the first line of the program. An infected program begins with the virus code and works as follows. The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus. When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system. This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions. Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress

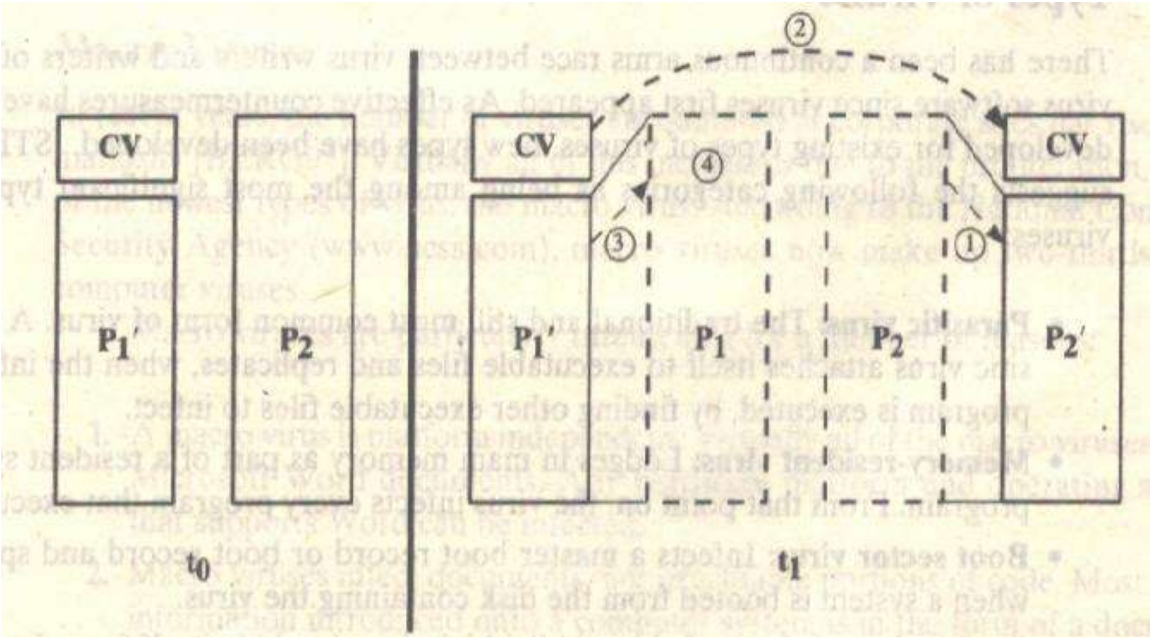
the executable file so that both the infected and uninfected versions are of identical length. We assume that program P1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps:

• For each uninfected file P2 that is found, the virus first compresses that file to produce P‘2, which is shorter than the original program by the size of the virus.

• A copy of the virus is prepended to the compressed program.

• The compressed version of the original infected program, P‘1, is uncompressed.

• The uncompressed original program is executed.



Initial Infection

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Most viral infections initiate with a disk from which programs are copied onto a machine. Many of these are disks that have games or simple but handy utilities that employees obtain for their home computers and then bring in and put on an office machine. Some are present on disks that come shrink-wrapped from the manufacturer of an application. Only a small fraction of infections begin across a network connection. Most of these are obtained from an electronic bulletin board system.

Types of Viruses

The main types of virus are,

Parasitic Virus

It is a virus that attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.

Memory-resident virus

It lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.

Boot sector virus

It infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.

Stealth virus

A form of virus explicitly designed to hide itself from detection by antivirus software.

Polymorphic virus

It is a virus that mutates with every infection, making detection by the "signature" of the virus impossible. One example of a stealth virus: a virus that uses compression so that the infected program is exactly the same length as an uninfected version.

A polymorphic virus creates copies during replication that are functionally equivalent but have distinctly different bit patterns. As with a stealth virus, the purpose is to defeat programs hat scan for viruses. In this case, the ―signature‖ of the virus will vary with each copy. To achieve this variation, the virus may randomly insert superfluous instructions or interchange the order of independent instructions. A more effective approach is to use encryption. A portion of the virus, generally called a *mutation engine,* creates a random encryption key to encrypt the remainder of the virus. The key is stored with the virus, and the mutation engine itself is altered. When an infected program is invoked, the virus uses the stored random key to decrypt the virus. When the virus replicates, a different random key is selected.

Another weapon in the virus writers' armory is the virus-creation toolkit. Such a toolkit enables a relative novice to create quickly a number of different viruses. Although viruses created with toolkits tend to be less sophisticated than viruses designed from scratch, the sheer number of new viruses that can be generated creates a problem for antivirus schemes.

Yet another tool of the virus writer is the virus exchange bulletin board. A number of such boards have sprung up in the United States and other

countries. These boards offer copies of viruses that can be downloaded, as well as tips for the creation of viruses.

Macro Viruses

According to the National Computer Security Agency (www.ncsa.com), macro viruses now make up two-thirds of all computer viruses. Macro viruses are particularly threatening for a number of reasons:

1.A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.

2,Macro viruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of a document rather than a program.

3.Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file. Typically, users employ macros to automate repetitive tasks and thereby save keystrokes. The macro language is usually some form of the Basic programming language. A user might define a sequence of keystrokes in a macro and set it up so that the macro is invoked when a function key or special short combination of keys is input. What makes it possible to create a macro virus is the auto executing macro. This is a macro

that is automatically invoked, without explicit user input. Common auto execute events are opening a file, closing a file, and starting an application. Once a macro is running, it can copy itself to other documents, delete files, and cause other sorts of damage to the user‘s system. In Microsoft Word, there are three types of auto executing macros.

Auto execute

If a macro named AutoExec is in the ―normal.dot‖ template or in a global template stored in Word‘s startup directory, it is executed whenever Word is started.

Auto macro

An auto macro executes when a defined event occurs, such as opening or closing a document, creating a new document, or quitting Word.

Command macro

If a macro in a global macro file or a macro attached to a document has the name of an existing Word command, it is executed whenever the user invokes that command

A common technique for spreading a macro virus is as follows. An auto macro or command macro is attached to a Word document that is introduced into a system by e-mail or disk transfer. At some point after the document is opened, the macro executes. The macro copies itself to the global macro file. When the next session of Word opens, the infected global macro is active

When this macro executes, it can replicate itself and cause damage. Successive releases of Word provide increased protection against macro viruses.

Worms

Network worm programs use network connections to spread from system to system. Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions.

To replicate itself, a network worm uses some sort of network vehicle. Examples include the following

• Electronic mail **facility:** A worm mails a copy of itself to other systems

• Remote execution capability A worm executes a copy of itself on another system.

• Remote login capability: A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion.

A network worm exhibits the same characteristics as a computer virus:

• Dormant phase

• Propagation phase

• Triggering phase

• Execution phase

The propagation phase generally performs the following functions:

• Search for other systems to infect by examining host tables or similar repositories of remote system addresses.

• Establish a connection with a remote system.

• Copy itself to the remote system and cause the copy to be run.

The network worm may also attempt to determine whether a system has previously been infected before copying itself to the system.

The Morris Worm

Until the current generation of worms, the best known was the worm released onto the Internet by Robert Morris in 1998. The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation. When a copy began execution, its first task was to discover other hosts to this host that would allow entry from this host. The worm performed this task by examining a variety of lists and tables, including system tables that declared which other machines were trusted by this host, users‘ mail forwarding files, tables by which users gave themselves permission for access to remote accounts, and from a program that reported

the status of network connections. For each discovered host, the worm tried a number of methods for gaining access:

1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file, and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password-cracking program that tried

a. Each user‘s account name and simple permutations of it

b. **A** list of 432 built-in passwords that Morris thought to be likely candidates

c. All the words in the local system directory

2. It exploited a bug in the finger protocol, which reports the whereabouts of a remote user.

3. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

4. If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter. It then sent this interpreter a short bootstrap program, issued a command to execute that program, and then logged off. The bootstrap program then called back the parent program

and downloaded the remainder of the worm. The new worm was then executed.

Recent Worm Attacks

The contemporary era of worm threats began with the release of the Code Red worm in July of 2001. Code Red exploits a security hole in the Microsoft Internet Information Server (IIS) to penetrate and spread. It also disables the system file checker in Windows. The worm probes random IP addresses to Spread to other hosts. During a certain period of time, it only spreads. It then initiates a denial-of-service attack against a government Web site by flooding the site with packets from numerous hosts. The worm then suspends activities and reactivates periodically. In the second wave of attack, Code Red infected nearly 360,000 servers in 14 hours. In addition to the havoc it causes at the targeted server, Code Red can consume enormous amounts of Internet capacity, disrupting service.

Code Red II is a variant that targets Microsoft IIS. In addition, this newer Worm installs a backdoor allowing a hacker to direct activities of victim computers. In late 2001, a more versatile worm appeared, known as Nimda. Nimda spreads by multiple mechanisms

From client to client via email

From client to client via open network shares

From web server to client via browsing of compromised website

From client to web server via active scanning for and exploitation of various Microsoft IIS 4.0/5.0 directory traversal vulnerabilities

From client to web server via scanning for the back doors left behind by the "Code Red II" worms

The worm modifies web documents and certain executable files found on the system it infects and create numerous copies of it under various filenames.

Bacteria

Bacteria are programs that do not explicitly damage any files. Their sole purpose is to replicate themselves. A typical bacteria program may do nothing more than execute two copies of itself simultaneously on a multiprogramming system, or perhaps create two new files, each of which is a copy of the original source file of the bacteria program. Both of those programs then may copy themselves twice, and so on. Bacteria reproduce exponentially, eventually taking up all the processor capacity, memory, or disk space, denying users access to those