Unit - V :

Protection: Goal, Domain of protection, Access matrix, Access control.

Security: The security problem, Program threats, System and network threats, User authentication.

Protection

- Goals of Protection
- Principles of Protection
- Domain of Protection
- Access Matrix
- Implementation of Access Matrix
- Access Control

Protection

- ✔ The processes in an operating system must be protected from one another's activities.
- ✔ To provide such protection, we can use various mechanisms to ensure that only processes that have gained proper authorization from the operating system can operate on the files, memory segments, CPU, and other resources of a system.
- ✔ Protection refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system.
- ✔ This mechanism must provide a means for specifying the controls to be imposed, together with a means of enforcement.
- ✔ We distinguish between protection and security, which is a measure of confidence that the integrity of a system and its data will be preserved.

Goals of Protection

- As computer systems have become more sophisticated and pervasive in their applications, the need to protect their integrity has also grown.
- Protection was originally conceived as an adjunct to multiprogramming operating systems, so that untrustworthy users might safely share a common logical name space such as a directory of files, or share a common physical name space, such as memory.
- Modern protection concepts have evolved to increase the reliability of any complex system that makes use of shared resources.
- **We need to provide protection for several reasons.**
- The most obvious is the need to prevent the mischievous, intentional violation of an access restriction by a user. Of more general importance, however, is the need to ensure that each program component active in a system uses system resources only in ways consistent with stated policies. This requirement is an absolute one for a reliable system.
- Protection can improve reliability by detecting latent errors at the interfaces between component subsystems. Early detection of interface errors can often prevent contamination of a healthy subsystem by a malfunctioning subsystem.
- Also, an unprotected resource cannot defend against use (or misuse) by an unauthorized or incompetent user. A protection-oriented system provides means to distinguish between authorized and unauthorized usage.

- The role of protection in a computer system is to provide a mechanism for the enforcement of the policies governing resource use.
- These policies can be established in a variety of ways. Some are fixed in the design of the system, while others are formulated by the management of a system.
- Still, others are defined by the individual users to protect their own files and programs.
- A protection system must have the flexibility to enforce a variety of policies.

Principles of Protection

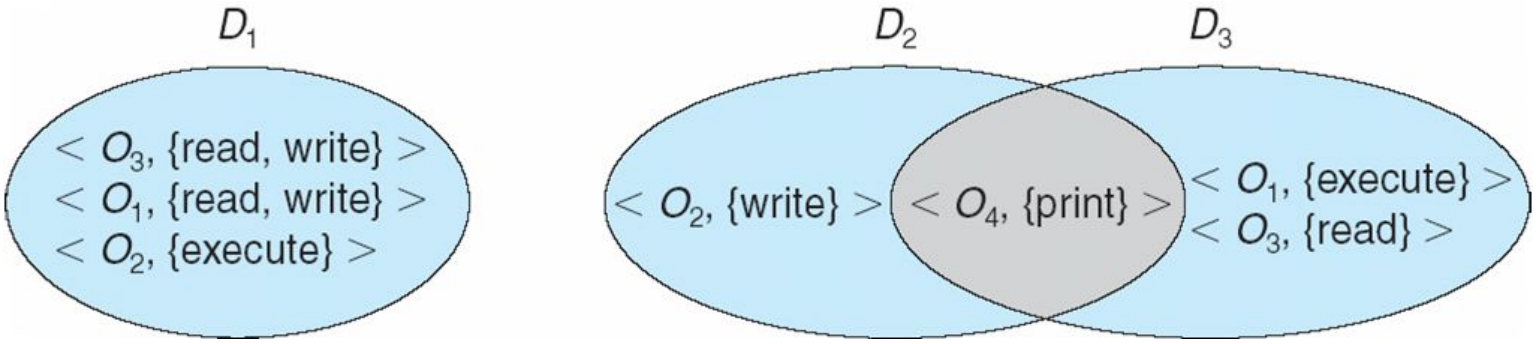
- Frequently, a guiding principle can be used throughout a project, such as the design of an operating system. Following this principle simplifies design decisions and keeps the system consistent and easy to understand.
- A key, time-tested guiding principle for protection is the **principle of least privilege.** It dictates that programs, users, and even systems be given just enough privileges to perform their tasks. The principle of least privilege can help produce a more secure computing environment.
- An operating system following the principle of least privilege implements its features, programs, system calls, and data structures so that failure or compromise of a component does the minimum damage and allows the n1inimum damage to be done.

Domain of Protection

- A computer system is a collection of processes and objects.
- By *objects,* we mean both hardware objects(such as the CPU, memory segments, printers, disks, and tape drives) and software objects(such as files, programs, and semaphores).
- Each object has a unique name that differentiates it from all other objects in the system, and each can be accessed only through well-defined and meaningful operations.
- Objects are essentially abstract data types.
- The operations that are possible may depend on the object.
- For example, on a CPU, we can only execute. Memory segments can be read and written, whereas a CD-ROM or DVD-ROM can only be read.
- Tape drives can be read, written, and rewound. Data files can be created, opened, read, written, closed, and deleted; program files can be read, written, executed, and deleted.
- A process should be allowed to access only those resources for which it has authorization.

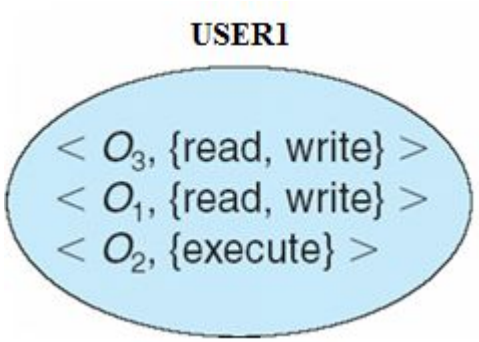
Domain Structure

- **Access-right** = <*object-name*, *rights-set*>
  where *rights-set* is a subset of all valid operations that can be performed on the object.
- Domain = set of access-rights



- Process operates within a protection domain.
- **Association between process and domain can be static or dynamic.**

- Static
  - Set of resources will be fixed.
- Dynamic
  - Set of resources can be changed
  - Process can switch domain



- Domain ca be visualized as :
- **User as domain**
  - Objects defined by user id
  - **Domain switch occurs when user log out**
- **Process as domain**
  - Objects defined by pid
  - **Domain switching occurs when process communicate to other process using IPC.**
- **Procedure as domain**
  - Objects are local variables
  - **Domain switch occurs via procedure call**

Access Matrix

- View protection as a matrix (*access matrix*)

- **Is a mechanism to apply policies of protection**
- Rows represent domains
- Columns represent objects
- ***Access(i, j)*** is the set of operations that a process executing in $Domain_i$ can invoke on $Object_j$ (access right)
- We must know which process executes in which domain.
- Content of access matrix is decided by user while which process will be in which domain is decided by OS

Use of Access Matrix (Cont)

- Access matrix design separates mechanism from policy
  - Policy
    - User dictates policy
    - Who can access what object and in what mode
  - Mechanism
    - Operating system provides access-matrix + rules
    - If ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced

Access Matrix illustration

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

4 domains and 4 objects (3 files and one printer).☐

When a process executes in D1, it can read files F1 and F3.☐

A process executing in D4 has the same privileges as it does in D1, it can also write onto files F1 and F3. ☐

Printer can be accessed by a process executing in D2.

Figure A

Process executing in domain D4 can execute read/write operation on object F1

Access Matrix

- Users decide the contents of the access-matrix entries.
- Access matrix provides mechanism for **defining and implementing strict control for both static and dynamic association between processes and domains**

- Controls changing content of access-matrix entries

- Dynamic: Controls switching between domains.
  - This can be done by including domain as an object in access matrix and switch access right.

Use of Access Matrix

- For static association:
- **For dynamic association: defining domain as object and switch access right.**

Access Matrix (static association)

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Process executing in domain D4
can execute read/write operation on object F1

Domain Switching:
Access Matrix with domains as objects

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Figure B

Process working in domain D1 can be switched to domain D2.

Access Matrix with Copy Rights

- Allowing controlled change to the contents of the access-matrix entries requires **3 additional operations**:
  - **Copy, Owner, and Control.**
- Copy
  - The ability to copy an access right from one domain (row) to another is denoted by an **asterisk (*) appended to the access right. (E.g read*)**
  - Copy right allows the copying of the access right only **within the column (that is, for the object) for which the right is defined.**

Access Matrix with Copy Rights

- need to update access matrix
- Special access rights: **(can change column entries)**
  - ***copy:***
    - ***Simple copy***
    - ***transfer***
    - ***Limited Copy:***

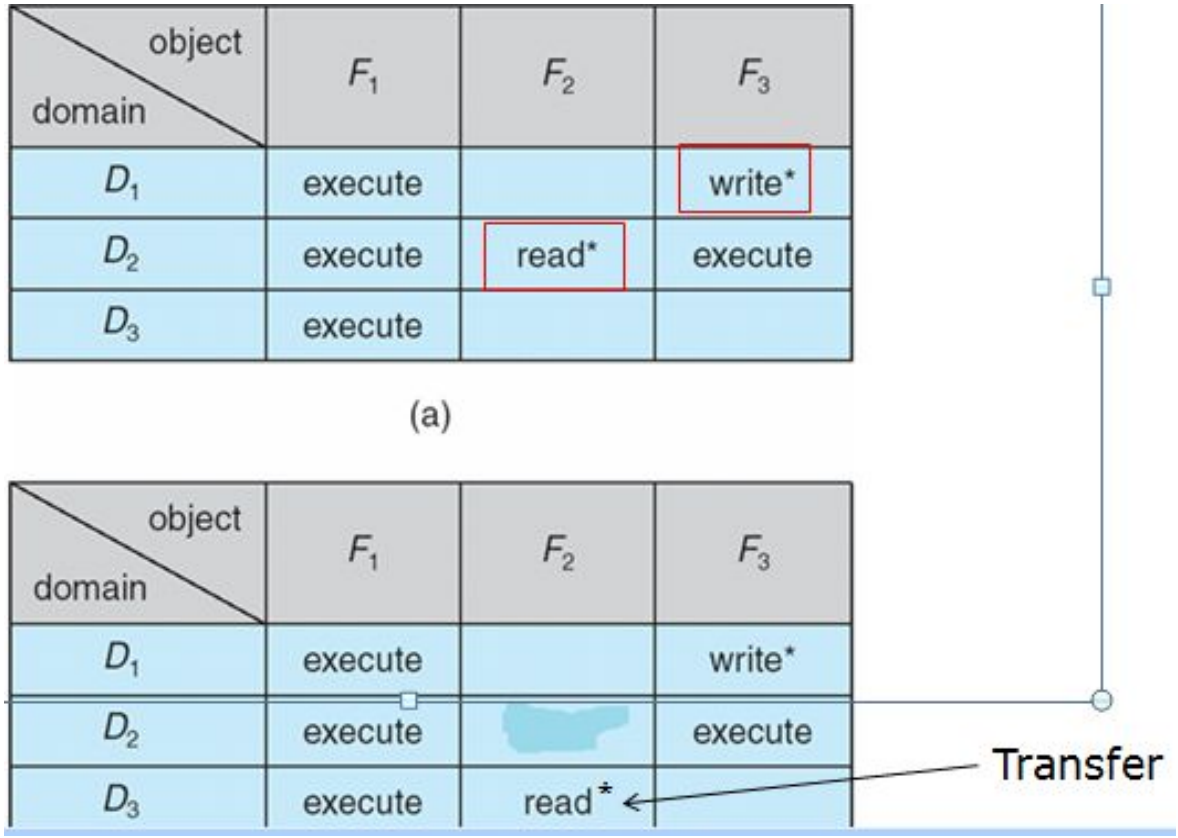
Access Matrix with Copy Rights

- ***Simple copy :*** *copy access right from Di to Dj.*
- When the right Read* is copied from access(i,j) to access(k,j), the Read* is created.So, a process executing in Dk can further copy the right Read*.

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	**read***	

Access Matrix with Copy Rights

- ***transfer –*** A right is copied from access(i,j) to access(k,j); it is then removed from access(i,j).



object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute		execute
D_3	execute	read*	

Access Matrix with *Copy* Rights

Propogation of copy right may be limited

- Limited Copy
- When the right Read* is copied from access(i,j) to access(k,j), only the Read (not Read*) is created. So, a process executing in Dkcannot further copy the right Read.

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

Limited Copy

(b)

Implementation of Access Matrix

- 1. Global Table: whenever operation M is to be performed on Oj within Di, global table is searched for triple <Di,Oj,Rk> where M belongs to Rk.

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Implementation of Access Matrix

- **2. Each column = Access-control list for one object**
  Defines who can perform what operation.

ACL for F1

Domain 1=Read
Domain 4=Read,write

ACL for F2

Domain 3=Read

ACL for F3

Domain 1 = Read
Domain 3 = Execute
Domain 4 = Read,write

ACL for Printer

Domain D2=Print

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

- **Each Row = Capability List (like a key)**
  For each domain, what operations allowed on what objects.

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Capability List for D1

F1-read
F3-read

Capability List for D2

Printer-print

Capability List for D3

F2-read
F3-execute

Capability List for D4

F1-read,write
F3-read,write

Implementation of Access Matrix

- **Lock-Key mechanism**
  - Each **object** has unique bit pattern called **lock**
  - Each **domain** has unique bit pattern called **key**
  - Process executing in a domain can access object if it has key that matches one of the lock of an object

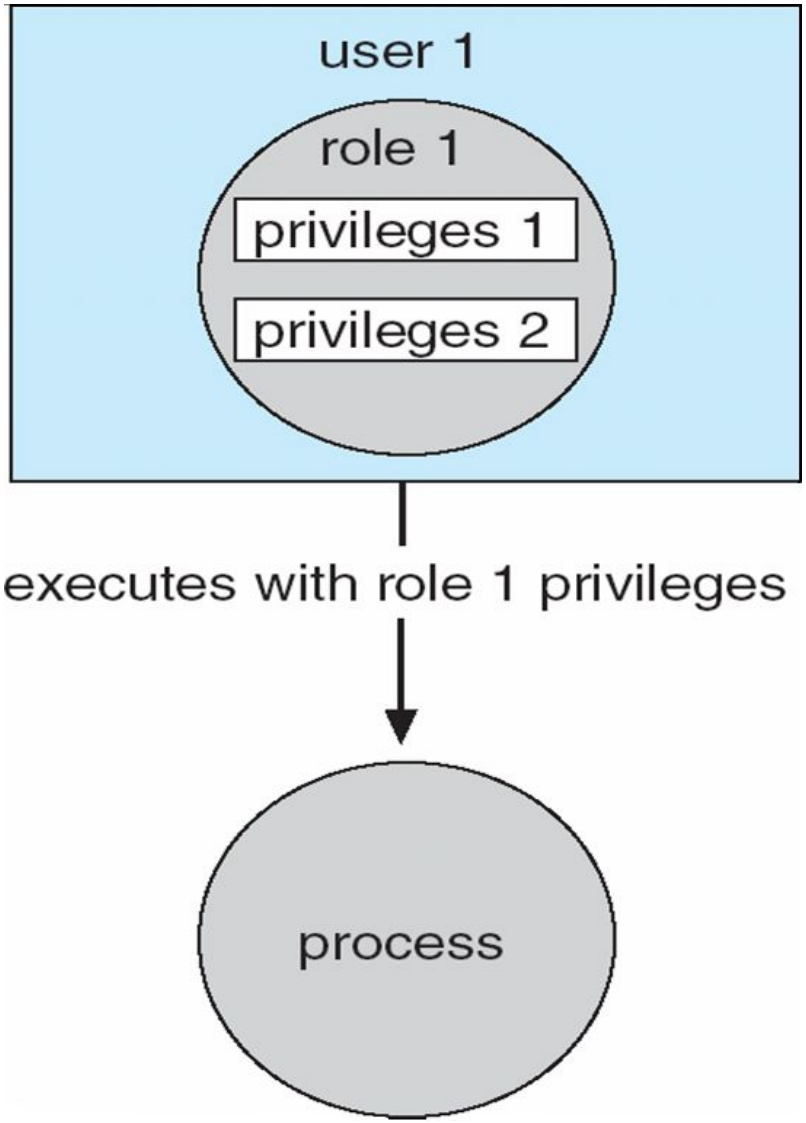
Access Control

- Privilege can be assigned to owner, group, or other users for a file or directory.
- Role based access control.

Access Control (RBAC in Solaris 10)

- Protection can be applied to non-file resources
- Solaris 10 provides role-based access control **(RBAC)** to implement least privilege
  - Privilege is right to execute system call or use an option within a system call
  - Can be assigned to processes
  - Users assigned roles granting access to privileges and programs

Role-based Access Control in Solaris 10



Revocation of Privileges & issues

- Removing privileges from the process:
- Types:
- Immediate versus delayed. Does revocation occur immediately, or is it delayed? If revocation is delayed, can we find out when it will take place?
- Selective versus general. When an access right to an object is revoked, does it affect *all the users who have an access right to that object, or can* we specify a select group of users whose access rights should be revoked?
- Partial versus total. Can a subset of the rights associated with an object be revoked, or must we revoke all access rights for this object?
- Temporary versus permanent. Can access be revoked permanently (that is, the revoked access right will never again be available), or can access be revoked and later be obtained again?

- Summary
- **Immediate / delayed** If delayed when to be revoked?
- **Selective / general** Revoked from selected / from all?
- **Total / partial** Revoked all / some privileges?
- **Permanent/temporary** Revocation permanent / temporary ?

Revocation for ACL / capability

- **Discuss schemes that implement revocation for capabilities.**
- **Access control list**: Revocation or deletion of privilege is easy
- But difficult in **capability list**
  - Problem of **reacquisition**
  - Some systems use
    - **back pointers** (pointers to capability is checked while revocation)
    - **Indirect method:** pointer to capability via global table
    - **Key:** Assigning master key to object and generating key while creating capability . If capability key matched with master key access is allowed.

Revocation for ACL / capability

- **Reacquisition.** Periodically, capabilities are deleted from each domain. If a process wants to use a capability, it may find that that capability has been deleted. The process may then try to reacquire the capability. If access has been revoked, the process will not be able to reacquire the capability.
- **Back-pointers**. A list of pointers is maintained with each object, pointing to all capabilities associated with that object. When revocation is required, we can follow these pointers, changing the capabilities as necessary. It is quite general, but its implementation is costly.
- **Indirection.** The capabilities point indirectly, not directly, to the objects. Each capability points to a unique entry in a global table, which in turn points to the object. We implement revocation by searching the global table for the desired entry and deleting it. Then, when an access is attempted, the capability is found to point to an illegal table entry. Table entries can be reused for other capabilities without difficulty, since both the capability and the table entry contain the unique name of the object.

Revocation for ACL / capability

- The object for a capability and its table entry must match. This scheme was adopted in the CAL system. It does not allow selective revocation.

- **Keys.** A key is a unique bit pattern that can be associated with a capability.
- This key is defined when the capability is created, and it can be neither modified nor inspected by the process that owns the capability. A Master key is associated with each object; it can be defined or replaced with the set-key operation. When a capability is created, the current value of the master key is associated with the capability. When the capability is exercised, its key is compared with the master key. If the keys match, the operation is allowed to continue; otherwise, an exception condition is raised. Revocation replaces the master key with a new value via the set-key operation, invalidating all previous capabilities for this object.
- This scheme does not allow selective revocation, since only one master key is associated with each object. If we associate a list of keys with each object, then selective revocation can be implemented. Finally, we can group all keys into one global table of keys. A capability is valid only if its key matches some key in the global table. We implement revocation by removing the matching key from the table. With this scheme, a key can be Associated with several objects, and several keys can be associated with each object, providing maximum flexibility.

Security

Security

- The Security Problem
- Program Threats
- System and Network Threats
- Cryptography as a Security Tool
- User Authentication
- Implementing Security Defenses
- Firewalling to Protect Systems and Networks
- Computer-Security Classifications
- An Example: Windows XP

Protection and Security

Protection is strictly an *internal* problem:
How do we provide controlled access to programs and data stored in a computer system? Security on the other hand, requires not only an adequate protection system but also consideration of the *external* environment within which the system operates. A protection system is ineffective if user authentication is compromised or a program is run by an unauthorized user.
Computer resources must be guarded against unauthorized access, malicious destruction or alteration, and accidental introduction of inconsistency.
These resources include information stored in the system (both data and code), as well as the CPU, memory, disks, tapes, and networking that are the computer.

Protection and Security

PROTECTION	SECURITY
A method used in operating systems that manages threats within the system to maintain the proper functioning of the system	A method used in operating systems that handles the threats from outside of the system to maintain the proper functioning of the system
Focuses on internal threats of the system	Focuses on external threats to the system
Provides a mechanism for controlling the access to programs, processes, and user resources	Provides a mechanism to safeguard the system resources and user resources from external users
Involves mechanisms such as setting or changing protection information of a resource and checking whether that resource is accessible by a user	Involves mechanisms such as adding, deleting users, verifying whether a specific user is authorized, using anti-malware software, etc.

Objectives

- To discuss security threats and attacks
- To explain the fundamentals of encryption, authentication, and hashing
- To examine the uses of cryptography in computing
- To describe the various countermeasures to security attacks

The Security Problem

- Security must consider **external environment** of the system, and protect the system resources
- **Threat** is potential **security violation**
- **Attack** is attempt to breach security
- Attack can be accidental or malicious
- Easier to protect against accidental than malicious misuse

Security Violations: Threat

- **Categories**
  - **Breach of confidentiality:**
  - **This type of violation involves unauthorized reading of data (theft of information)**
  - **Ex: stealing credit-card info or identity info for identity theft**
  - **Breach of integrity:**
  - **This type of violation involves unauthorized modification of data**
  - **Can result in modification of commercial application**
  - **Breach of availability: This type of violation involves unauthorized destruction of data**
  - **Ex: defacing of a web page**

Security Violations: Threat

- **Categories**
  - **Theft of service: This type of violation involves unauthorized use of system and resources**
  - **using someone's computer for misuse**
  - **Denial of service (DOS): prevent legitimate use of service.**
  - A Denial-of-Service (DoS) attack **aims to disrupt the normal functioning of a network or server by overwhelming it with excessive traffic, making it unavailable to legitimate users.**

Security Violations: Threat

- **Methods to break security:**
  - **Masquerading (breach authentication)**
    - One participant in a communication pretends to be someone else
    - Could be another host or person.
    - Goal is to gain access that they would not normally be allowed
      Or could try to escalate their privileges
  - **Replay attack**
    - Replay a captured exchange of data
    - Sometimes the replay is the attack: ex: repeat of a request to transfer money
    - Message modification: replace some data in the replay to obtain access for unauthorized user

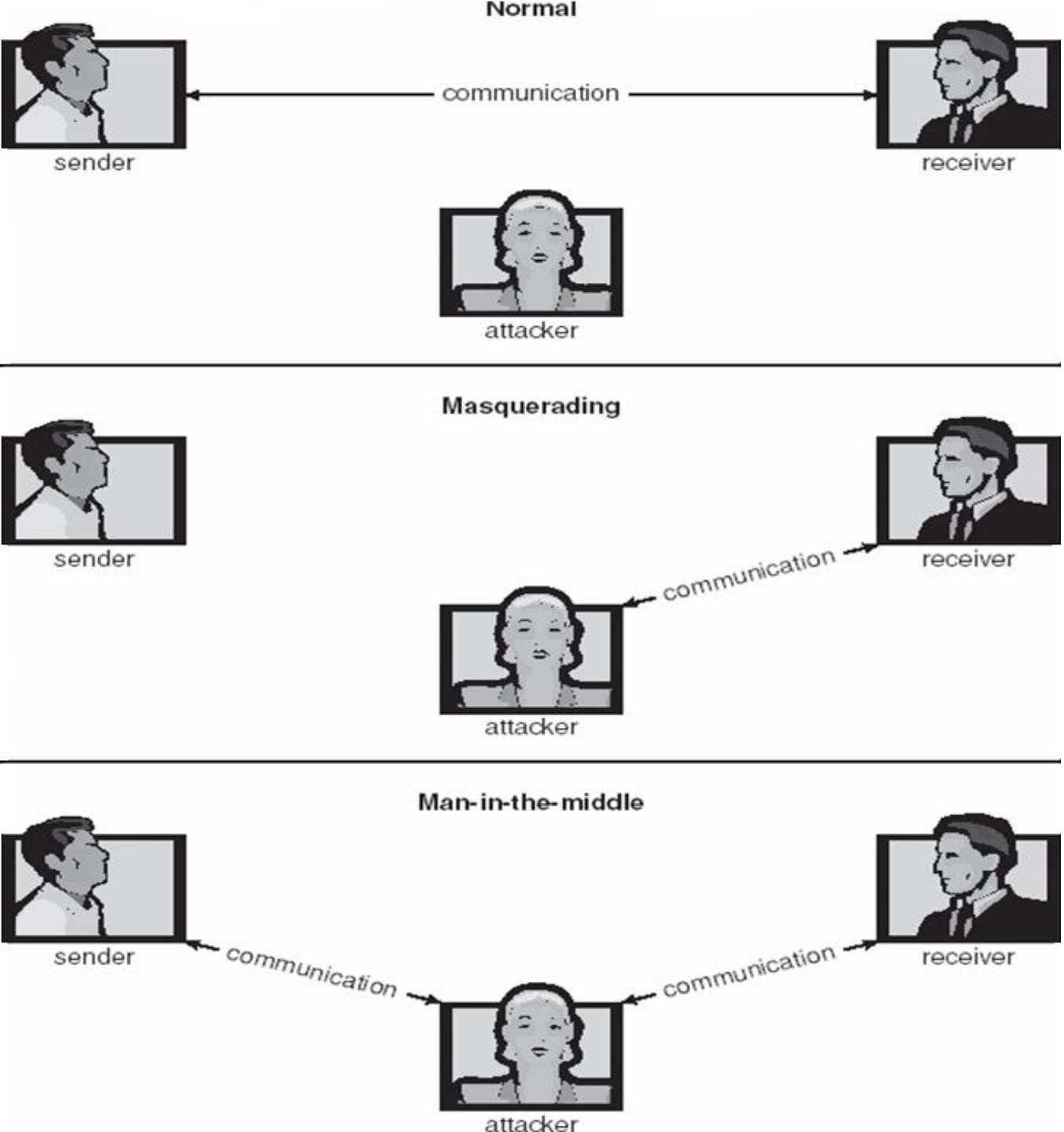
Security Violations: Threat

Methods to break security:

- A **Man-in-the-Middle attack** occurs when a third party (the "attacker") intercepts and possibly alters the communication between two parties who believe they are directly communicating with each other.
- This type of attack is dangerous because the attacker can steal sensitive information, inject malicious data.

Session Hijacking: The attacker steals an active session (for example, the session token used in web applications) to impersonate the user. Or An active communication session is intercepted.

Standard Security Attacks



Threat Types

- Program Threats
  - Trojan horse
  - Trap door
  - Logic bomb
  - viruses
- System and Network threats
  - Worms
  - Port scanning
  - Denial of service attack

Trojan Horse

- Many systems have mechanisms for allowing programs written by users to be executed by other users.
- If these programs are executed in a domain that provides the access rights of the executing user, the other users may misuse these rights.
- A text-editor program, for example, may include code to search the file to be edited for certain keywords. If any are found, the entire file may be copied to a special area accessible to the creator of the text editor. A code segment that misuses its environment is called a Trojan horse.
- The search path lists the set of directories to search when an ambiguous program name is given. The path is searched for a file of that name, and the file is executed. All the directories in such a search path must be secure, or a Trojan horse could be slipped into the user's path and executed accidentally.

Trojan Horse

- For instance, consider the use of the "." character in a search path. The "." tells the shell to include the current directory in the search.
- Thus, if a user has "." in her search path, has set her current directory to a friend's directory, and enters the name of a normal system command, the command may be executed from the friend's directory instead.
- The program would run within the user's domain, allowing the program to do anything that the user is allowed to do, including deleting the user's files, for instance.
- A variation of the Trojan horse is a program that emulates a login program. An unsuspecting user starts to log in at a terminal and notices that he has apparently mistyped his password. He tries again and is successful. What has happened is that his authentication key and password have been stolen by the login emulator, which was left running on the terminal by the thief.

Trojan Horse

- Another variation on the Trojan horse is spyware.
- Spyware sometimes accompanies a program that the user has chosen to install.
- Most frequently, it comes along with freeware or shareware programs, but sometimes it is included with commercial software.
- The goal of spyware is to download ads to display on the user's system, create pop-up browser windows when certain sites are visited, or capture information from the user's system and return it to a central site. This latter mode is an example of a general category of attacks known as covert channels, in which surreptitious communication occurs.

Examples of Trojan Horse

Backdoor Trojans

As the name implies, this Trojan can create a "backdoor" on a computer to gain access to it without the user's knowledge. A backdoor Trojan allows an attacker to acquire remote access to control a computer, typically uploading, downloading, or executing data at will. These are one of the most basic yet potentially harmful varieties of Trojans. They are primarily used to install new malware, spy on you, and steal your data.

Downloader Trojans

A Trojan downloader is a sort of Trojan that installs itself on the system and then connects to a remote server or website to download more applications (typically malware) onto the affected machine. This Trojan is designed to infect a computer even more than it already is. It downloads and installs new versions of malicious applications. Trojans and adware are examples of such threats. Trojan downloaders are also often distributed in disguised file attachments in spam emails.

Examples of Trojan Horse

Mailfinder Trojans

A mailfinder Trojan seeks to harvest and steal email addresses saved on a computer and sends them to the criminal users via email, the web, file transfer protocol (FTP), or other methods. Cybercriminals then utilize stolen addresses to send out large, bulk-based mailings of malware and spam.

Rootkit Trojans

Rootkits are meant to conceal specific activities or items in the system. Their primary goal is to prevent the detection of malicious tasks to extend the time the programs can function on the machine, resulting in maximum damage.

Trojan Horse

How it spreads:

- Email attachments
- Sending files in chat rooms
- Infected computer can attack other computer

Consequences to user:

- Loss of identity
- Permanent damage to hardware
- Corruption of files in an operating system
- Financial loss to corporations or user
- Reduction of system performance

Trojan Horse

Ways to avoid this attack:

- Don’t download from unknown source
- Don’t type a command or go to web site told by a stranger
- Remove unnecessary services and file shares
- Anti-virus/ Anti-Trojan software
- Monitor the sites and files you often visit and use.
- Be aware of what you upload and download to your operating system.

Trap Door

- A trap door is kind of a secret entry point into a program that allows anyone gain access to any system without going through the usual security access procedures.
- Also says that it is a method of bypassing normal authentication methods.
- Also known as Back Door.
- It is created by the system programmer just to bypass some normal check.
- A clever trap door could be included in a compiler. The compiler could generate standard object code as well as a trap door, regardless of the source code being compiled.
- This activity is particularly nefarious, since a search of the source code of the program will not reveal any problems. Only the source code of the compiler would contain the information.
- Trap Door pose a difficult problem because, to detect them, we have to analyze all the source code for all components of a system.
- Software system consist of millions of lines of code, this analysis is not done frequently and frequently it is not done at all!

Logic Bomb

- A Logic Bomb is a piece of code inserted into an operating system or software program that implements a malicious function after a certain time limit or specific conditions are met. Logic bombs are often used with viruses, worms and trojan horses to time to do maximum damage before being noticed.

- They perform actions like corrupting or altering data, reformatting a hard drive, and deleting important files.

Logic Bomb

Example:

A story about a programmer at a large corporation who engineered this type of attack. Apparently, the programmer had been having some trouble at the company at which he worked and was on probation. Fearing that he might be fired, he added a subroutine to another program. The subroutine was added to a program that ran once a month and was designed to scan the company's human resources employee database to determine if a termination date had been loaded for his employee record. If the subroutine found that a termination date had been loaded, then it was designed to wipe out the entire system by deleting all files on the disk drives. The program ran every month and so long as his employee record did not have a termination date then nothing would happen. In other words, if he were not fired the program would do no damage.

Sure enough, this stellar employee was fired, and the next time the logic bomb that he created ran it found a termination date in his employee record and wiped out the system. This is an example of how simple it can be, for one with privileged access to a system, to set up this type of attack.

Logic Bomb

Prevention from Logic Bomb:

- Use a strong antivirus and update it regularly.
- Don’t download pirated software.
- If you need a freeware, make sure you’re downloading it from a reputable source.
- Keep your operating system up to date.
- Practice good internet behavior – don’t click on suspicious links or email attachments.
- If you run a company, make sure you protect all computers individually.

Viruses

- Another form of program threat is a virus.
- Viruses are self-replicating and are designed to "infect" other programs. They can wreak havoc(create destruction) in a system by modifying or destroying files and causing system crashes and program malfunctions.
- A virus is a fragment of code embedded in a legitimate program. As with most penetration attacks, viruses are very specific to architectures, operating systems, and applications. Viruses are a particular problem for users of pcs. UNIX and other multiuser operating systems generally are not susceptible to viruses because the executable programs are protected from writing by the operating system. Even if a virus does infect such a program, its powers usually are limited because other aspects of the system are protected.

- Viruses are usually borne via email, with spam the most common vector. They can also spread when users download viral programs from Internet file-sharing services or exchange infected disks.

Viruses

- Another common form of virus transmission uses Microsoft Office files, such as Microsoft Word documents.
- These documents can contain macros (or Visual Basic programs) that programs in the Office suite (Word, PowerPoint, and Excel) will execute automatically.
- Because these programs run under the user's own account, the macros can run largely unconstrained (for example, deleting user files at will).
- Commonly, the virus will also e-mail itself to others in the user's contact list.
- **How do viruses work?**
- Once a virus reaches a target machine, a program known as a virus dropper inserts the virus onto the system.
- The virus dropper is usually a Trojan horse, executed for other reasons but installing the virus as its core activity.
- Once installed, the virus may do anyone of a number of things.
- There are literally thousands of viruses, but they fall into several main categories.
- Note that many viruses belong to more than one category.
-

Viruses

• File. A standard file virus infects a system by appending itself to a file. It changes the start of the program so that execution jumps to its code. After it executes, it returns control to the program so that its execution is not noticed. File viruses are sometimes known as parasitic viruses, as they leave no full files behind and leave the host program still functional.

• Boot. A boot virus infects the boot sector of the system, executing every time the system is booted and before the operating system is loaded. It watches for other bootable media (that is, floppy disks) and infects them. These viruses are also known as memory viruses, because they do not appear in the file system. Figure 15.5 shows how a boot virus works.

• Macro. Most viruses are written in a low-level language, such as assembly or C. Macro viruses are written in a high-level language, such as Visual Basic. These viruses are triggered when a program capable of executing the macro is run. For example, a macro virus could be contained in a spreadsheet file.

Viruses

Source code. A source code virus looks for source code and modifies it to include the virus and to help spread the virus.

- Polymorphic. This virus changes each time it is installed to avoid detection by antivirus software. The changes do not affect the virus's functionality but rather change the virus's signature. A virus signature is a pattern that can be used to identify a virus, typically a series of bytes that make up the virus code.

- Encrypted. An encrypted virus includes decryption code along with the encrypted virus, again to avoid detection. The virus first decrypts and then executes.

- Stealth. This tricky virus attempts to avoid detection by modifying parts of the system that could be used to detect it. For example, it could modify the read system call so that if the file it has modified is read, the original form of the code is returned rather than the infected code.

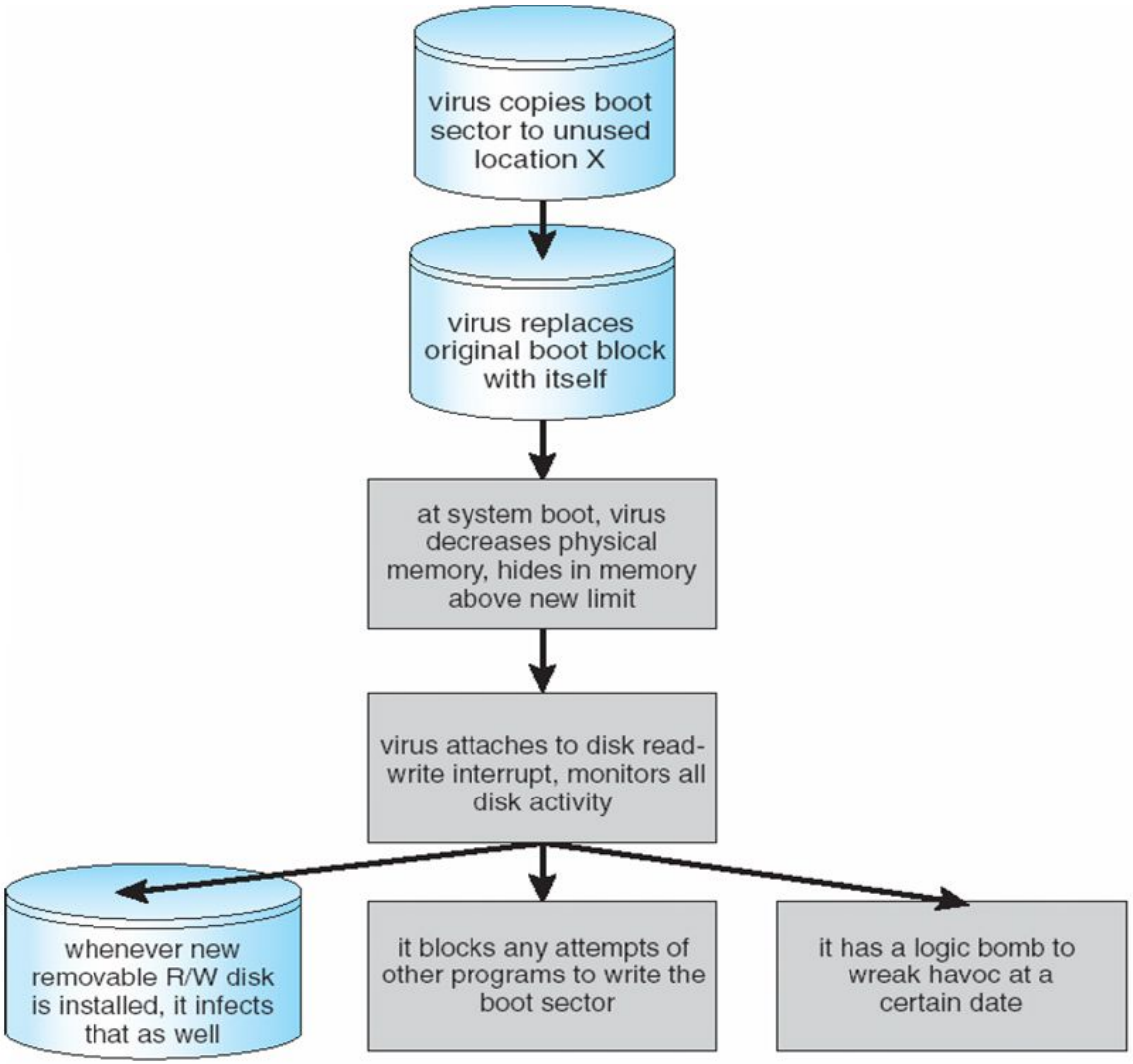
Viruses

- Tunneling. This virus attempts to bypass detection by an antivirus scanner by installing itself in the interrupt-handler chain. Similar viruses install themselves in device drivers.

- Multipartite. A virus of this type is able to infect multiple parts of a system, including boot sectors, memory, and files. This makes it difficult to detect and contain.

- Armored. An armored virus is coded to make itself hard for antivirus researchers to unravel and understand. It can also be compressed to avoid detection and disinfection. In addition, virus droppers and other full files that are part of a virus infestation are frequently hidden via file attributes or unviewable file names.

A Boot-sector Computer Virus



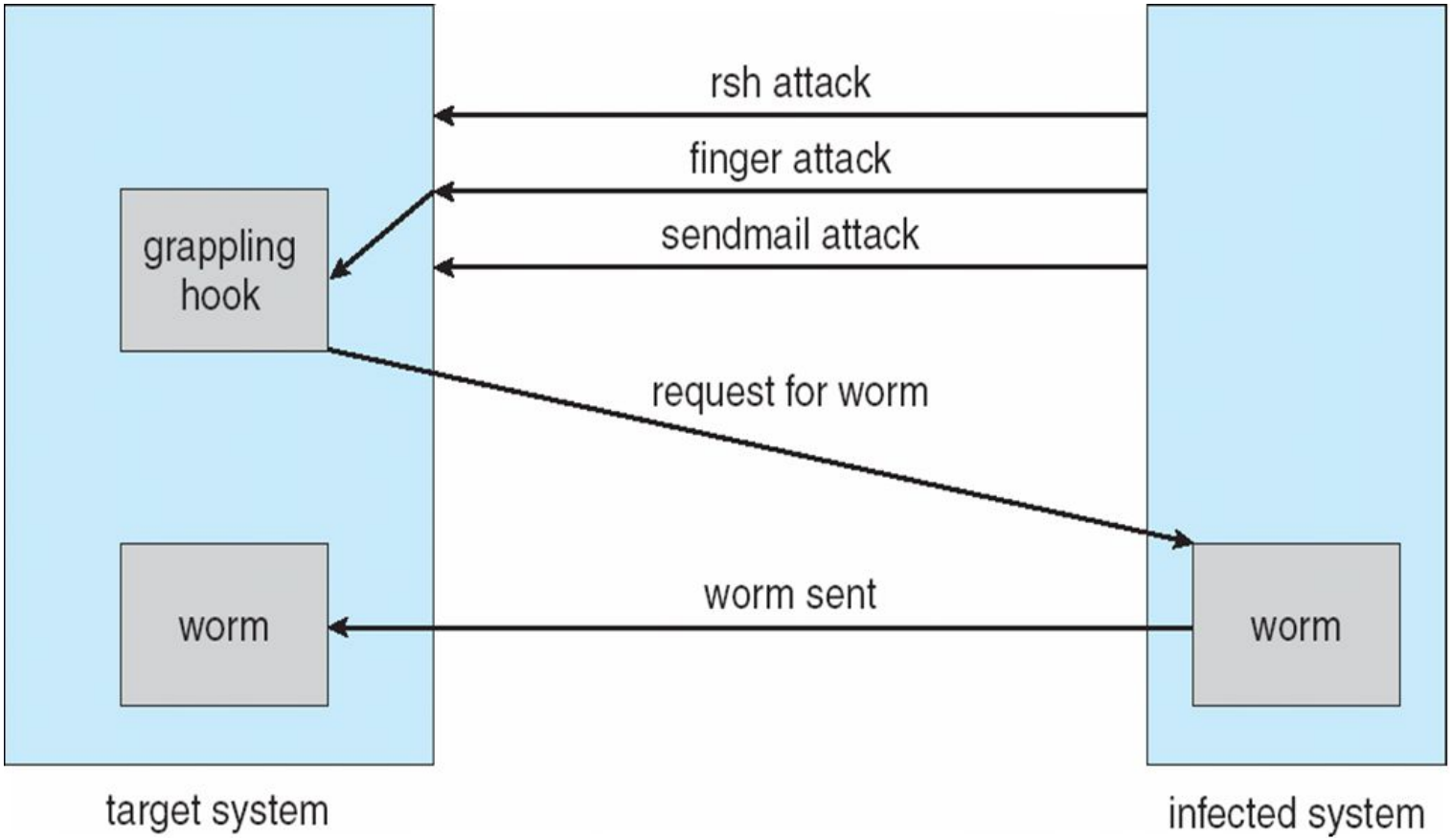
System and Network Threats

- **Worms** – standalone program
  - use **spawn (reproduction)** mechanism, use system resources and lock out all other processes to damage system;
  - Mainly attack on network thus shut down an entire network.
  - Robert Morris used this attack

- **Attack on remote access feature and bugs in *finger* and *sendmail* programs.**

- **It is made up of two programs**
  - **Grappling hook (also called bootstrap or vector)**
  - **Main program (/1.c)**

The Morris Internet Worm



The Morris Internet Worm

- A ***worm*** is a process that uses the fork/spawn process to make copies of itself in order to wreak havoc on a system. Worms consume system resources, often blocking out other, legitimate processes. Worms that propagate over networks can be especially problematic, as they can tie up vast amounts of network resources and bring down large-scale systems.
- One of the most well-known worms was launched by Robert Morris, a graduate student at Cornell, in November 1988. Targeting Sun and VAX computers running BSD UNIX version 4, the worm spanned the Internet in a matter of a few hours, and consumed enough resources to bring down many systems.
- This worm consisted of two parts:
    - A small program called a ***grappling hook,*** which was deposited on the target system through one of three vulnerabilities, and
    - The main worm program, which was transferred onto the target system and launched by the grappling hook program.

The Morris Internet Worm

The three vulnerabilities exploited by the Morris Internet worm were as follows:

- **rsh (remote shell)** is a utility that was in common use at that time for accessing remote systems without having to provide a password. If a user had an account on two different computers (with the same account name on both systems), then the system could be configured to allow that user to remotely connect from one system to the other without having to provide a password. Many systems were configured so that ***any*** user (except root) on system A could access the same account on system B without providing a password.
- **finger** is a utility that allows one to remotely query a user database, to find the true name and other information for a given account name on a given system. For example "finger joeUser@somemachine.edu" would access the finger daemon at somemachine.edu and return information regarding joeUser. Unfortunately the finger daemon (which ran with system privileges) had the buffer overflow problem, so by sending a special 536-character user name the worm was able to fork a shell on the remote system running with root privileges.
- **sendmail** is a routine for sending and forwarding mail that also included a debugging option for verifying and testing the system. The debug feature was convenient for administrators, and was often left turned on. The Morris worm exploited the debugger to mail and execute a copy of the grappling hook program on the remote system.

- Once in place, the worm undertook systematic attacks to discover user passwords:
  - First it would check for accounts for which the account name and the password were the same, such as "guest", "guest".
  - Then it would try an internal dictionary of 432 favorite password choices. (I'm sure "password", "pass", and blank passwords were all on the list.)
  - Finally it would try every word in the standard UNIX on-line dictionary to try and break into user accounts.
- Once it had got access to one or more user accounts, then it would attempt to use those accounts to rsh to other systems, and continue the process.
- With each new access the worm would check for already running copies of itself, and 6 out of 7 times if it found one it would stop. (The seventh was to prevent the worm from being stopped by fake copies.)
- Fortunately the same rapid network connectivity that allowed the worm to propagate so quickly also quickly led to its demise - Within 24 hours remedies for stopping the worm propagated through the Internet from administrator to administrator, and the worm was quickly shut down.
- There is some debate about whether Mr. Morris's actions were a harmless prank or research project that got out of hand or a deliberate and malicious attack on the Internet. However the court system convicted him, and penalized him heavy fines and court costs.

- **rsh** : user can omit entering a password each time they access remote account by configuring some special files. Worm search these files for sites that allow remote execution without password.
- **Finger** : users can provide their personal information. Attacker uses buffer overflow attack on finger.
- **Sendmail** : worm discover user passwords.

Port scanning

- ***Port Scanning*** is technically not an attack, but rather a search for vulnerabilities to attack. The basic idea is to systematically attempt to connect to every known (or common or possible) network port on some remote machine, and to attempt to make contact. Once it is determined that a particular computer is listening to a particular port, then the next step is to determine what daemon is listening, and whether or not it is a version containing a known security flaw that can be exploited.
- Because port scanning is easily detected and traced, it is usually launched from ***zombie systems,*** i.e. previously hacked systems that are being used without the knowledge or permission of their rightful owner.

Denial of Service

- Denial-of-service attacks are aimed not at gaining information or stealing resources but rather at disrupting the legitimate use of a system or facility.
- Most such attacks involve systems that the attacker has not penetrated. Indeed, launching an attack that prevents legitimate use is frequently easier than breaking into a machine or facility.
- Denial-of-service attacks are generally network based. They fall into two categories. Attacks in the first category use so many facility resources that, in essence, no useful work can be done. For example, a Web-site click could download a Java applet that proceeds to use all available CPU time or to pop up windows infinitely.
- The second category involves disrupting the network of the facility. There have been several successful denial-of-service attacks of this kind against major Web sites.
- These attacks result from abuse of some of the fundamental functionality of TCP */IP. For instance, if the attacker sends the part* of the protocol that says "I want to start a TCP connection," but never follows with the standard "The connection is now complete," the

Denial of Service

- If enough of these sessions are launched, they can eat up all the network resources of the system, disabling any further legitimate TCP connections. Such attacks, which can last hours or days, have caused the partial or full failure of attempts to use the target facility. The attacks are usually stopped at the network level until the operating systems can be updated to reduce their vulnerability.
- Generally, it is impossible to prevent denial-of-service attacks. The attacks
- use the same mechanisms as normal operation. There are other interesting aspects of DOS attacks. For example, if an authentication algorithm locks an account for a period of time after several incorrect attempts to access the account, then an attacker could cause all authentication to be blocked by purposely making incorrect attempts to access all accounts.

User Authentication

- Checking validity of user
- Can be done using three things
  - User's possession of something (key / card)
  - Users knowledge of something (id / password)
  - User's attribute (finger print / face / retina pattern / signature)

User Authentication : Passwords

- User identity most often established through *passwords.*
- *Password can be also given to each resource. E.f. file*
- *Different passords may be associated with different access rights. (read / write / read-write)*
- Password Vulnerabilities
  - Password guessing
    - User selected password
      - Personal information
      - Trying enumerations of characters (use long password)
  - Shoulder surfing
  - Monitoring network (n/w sniffing)
  - Password exposure
    - System selected password is hard to guess.
  - Sharing user id

- Passwords must be kept secret

Password

- Some systems accept only **strong password**
- Some systems forces to **change password regularly** / at the end of each session.
- Users can toggle password. **System records N passwords and do not allow user to use that password again.**

Password Vulnerabilities

- Passwords are extremely common because they are easy to understand and use.
- Unfortunately, passwords can often be guessed, accidentally exposed, sniffed,or illegally transferred from an authorized user to an unauthorized one, as we show next.

- **There are two common ways to guess a password**.
- One way is for the intruder (either human or program) to know the user or to have information about the user.
- All too frequently, people use obvious information (such as the names of their cats or spouses) as their passwords.
- The other way is to use brute force(trial and error), trying enumeration (counting)or all possible combinations of valid password characters (letters, numbers, and punctuation on some systems)-until the password is found.

- Short passwords are especially vulnerable to this method.
- For example, a four-character password provides only 10,000 variations.
- On average, guessing 5,000 times would produce a correct hit.

Password Vulnerabilities

- A program that could try a password every millisecond would take only about 5 seconds to guess a four-character password.
- Enumeration is less successful where systems allow longer passwords that include both uppercase and lowercase letters, along with numbers and all punctuation characters.
- Of course, users must take advantage of the large password space and must not use only lowercase letters.
- In addition to being guessed, passwords can be exposed as a result of visual or electronic monitoring.
- An intruder can look over the shoulder of a user when the user is logging in and can learn the password easily by watching the keyboard.
- Alternatively, anyone with access to the network on which a computer resides can seamlessly add a network monitor, allowing him to watch all data being transferred on the network including user IDs and passwords. Encrypting the data stream containing the password solves this problem. Even such a system could have passwords stolen, however. For example, if a file is used to contain the passwords, it could be copied for off-system analysis.
- Exposure is a particularly severe problem if the password is written down where it can be read or lost. As we shall see, some systems force users to select hard-to-remember or long passwords, which may cause a user to record the password or to reuse it. As a result, such systems provide much less security than systems that allow users to select easy passwords!

Password Vulnerabilities

- The final type of password compromise, illegal transfer, is the result of human nature. Most computer installations have a rule that forbids users to share accounts. This rule is
- sometimes implemented for accounting reasons but is often aimed at improving security. For instance, suppose one user ID is shared by several users, and a security breach occurs from that user ID.
- It is impossible to know who was using the ID at the time the break occurred or even whether the user was an authorized one. With one user per user ID, any user can be questioned directly about use of the account; in addition, the user might notice something different about the account and detect the break-in.
- Sometimes, users break account-sharing rules to help friends or to circumvent accounting, and this behavior can result in a system's being accessed by unauthorized users -possibly harmful ones.

Password Vulnerabilities

- **Passwords can be either generated by the system or selected by a user.**
- System-generated passwords may be difficult to remember, and thus users may write them down. As mentioned, however, user-selected passwords are often easy to guess (the user's name or favorite car, for example). Some systems will check a proposed password for ease of guessing or cracking before accepting it. At some sites, administrators occasionally check user passwords and notify a user if his password is easy to guess. Some systems also *age passwords, forcing* users to change their passwords at regular intervals (every three months, for instance).
- This method is not foolproof either, because users can easily toggle between two passwords. The solution, as implemented on some systems, is to record a password history for each user. For instance, the system could record the last *N passwords and not allow their reuse.* Several variants on these simple password schemes can be used. For example,the password can be changed more frequently. In the extren:1e, the password is changed from session to session. A new password is selected (either by the system or by the user) at the end of *each session, and that password* must be used for the next session. In such a case, even if a password is misused, it can be used only once. When the legitimate user tries to use a now-invalid password at the next session, he discovers the security violation. Steps can then be taken to repair the breached security.