OPERATING SYSTEMS

Input - Output Management and Security - 5,6,7,8,9,10

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 5

Unit 5 : I/O Management and Security

I/O Hardware, polling and interrupts, DMA, Kernel I/O Subsystem and Transforming I/O Requests to Hardware Operations - Device interaction, device driver, buffering. System Protection: Goals, Principles and Domain of Protection, Access Matrix, Access control, Access rights. System Security: The Security Problem,Program Threats, System Threats and Network Threats. Case Study: Windows 7/Windows 10

OPERATING SYSTEMS

Course Outline

- I/O Hardware, polling and interr...
- Kernel I/O Subsystem
- Transforming I/O Requests to H...
- System Protection: Goals, Princi...
- Domain of Protection: Unix, MU...
- Access Matrix
- Implementation of Access Matri...
- System Security: The Security Pr...
- Program Threats.
- System Threats and Network Th...

- Implementation of Access Matrix
- Access Control
- Access Rights

Implementation of Access Matrix

- In general, the matrix will be sparse; that is, most of the entries will be empty.

- Although data structure techniques are available for representing sparse matrices, they are not particularly useful for this application, because of the way in which the protection facility is used.

Implementation of Access Matrix

- **Global Table:**
  - The simplest implementation of the access matrix is a global table consisting of a set of ordered triples < domain, object, rights-set >.
  - Whenever an operation M is executed on an object Oj within domain Di , the global table is searched for a triple <Di , Oj , Rk >, with M ∈ Rk .
  - If this triple is found, the operation is allowed to continue; otherwise, an exception (or error) condition is raised.

Implementation of Access Matrix

- **Global Table:**
  - This implementation suffers from several drawbacks. The table is usually large and thus cannot be kept in main memory, so additional I/O is needed.
  - Virtual memory techniques are often used for managing this table.
  - In addition, it is difficult to take advantage of special groupings of objects or domains.

Implementation of Access Matrix

- **Access Lists for Objects:**
  - List for each object consists of ordered pairs <domain, rights-set >, which define all domains with a nonempty set of access rights for that object.
  - This approach can be extended easily to define a list plus a default set of access rights.
  - When an operation M on an object Oj is attempted in domain Di , we search the access list for object Oj , looking for an entry <Di , Rk > with $M \in Rk$.
  - If the entry is found, we allow the operation; if it is not, we check the default set.
  - If M is in the default set, we allow the access. Otherwise, access is denied, and an exception condition occurs.

Implementation of Access Matrix

- **Capability Lists for Domains:**
  - A capability list for a domain is a list of objects together with the operations allowed on those objects.
  - An object is often represented by its physical name or address, called a **capability**.
  - To execute operation M on object Oj , the process executes the operation M, specifying the capability (or pointer) for object Oj as a parameter.
  - Simple possession of the capability means that access is allowed.
  - The capability list is associated with a domain, but it is never directly accessible to a process executing in that domain

- An ability is the power, skill, means and opportunity to do something.
- A capability is the ability, aptitude or fitness to do something.

Access Control

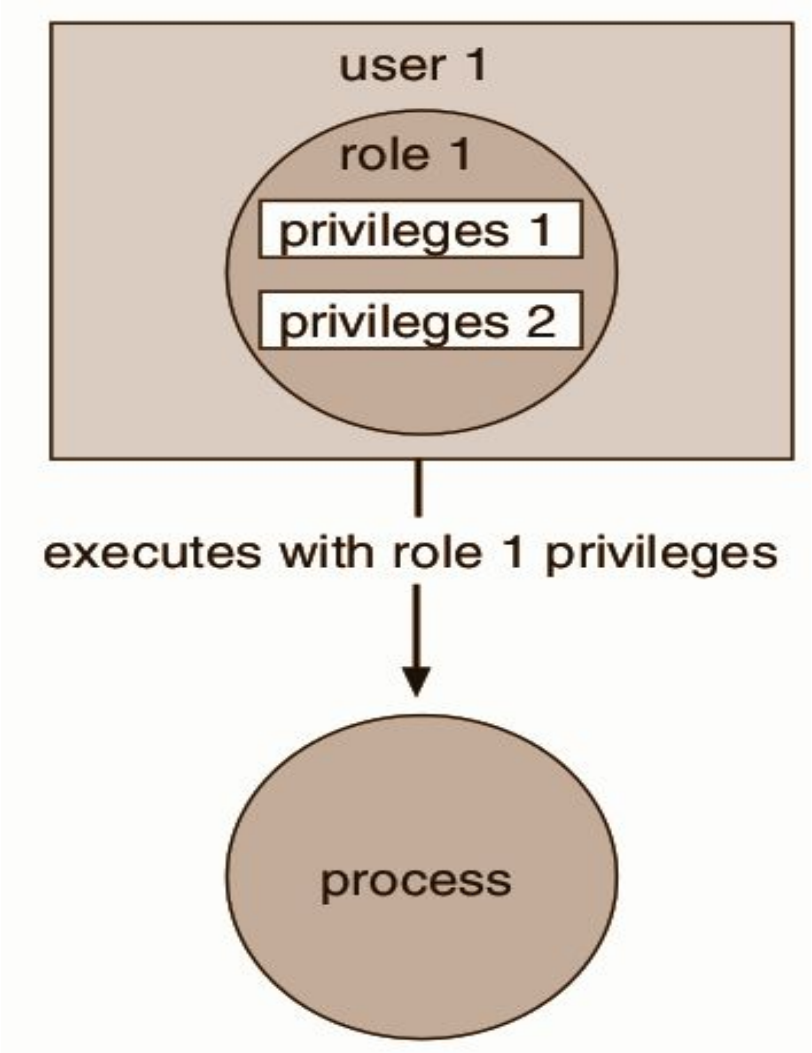
- Each file and directory is assigned an owner, a group, or possibly a list of users, and for each of those entities, access-control information is assigned.
- A similar function can be added to other aspects of a computer system.
- Solaris 10 advances the protection available in the operating system by explicitly adding the principle of least privilege via role-based access control (RBAC).
- This facility revolves around privileges.

Access Control

- A **Privilege** is the right to execute a system call or to use an option within that system call (such as opening a file with write access).
- Privileges can be assigned to processes, limiting them to exactly the access they need to perform their work.
- Privileges and programs can also be assigned to roles.
- Users are assigned roles or can take roles based on passwords to the roles.

Access Control

- Notice that this facility is similar to the access matrix



Role-based access control in Solaris 10.

Access Rights

- In a dynamic protection system, we may sometimes need to **revoke** access rights to objects shared by different users

- Various questions about revocation may arise:
  - **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 ?

Access Rights: Reacquisition, Back-pointers

- Schemes that implement **revocation** for capabilities include the following
  - **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.
    - This scheme was adopted in the MULTICS system.
    - It is quite general, but its implementation is costly.

- An object is often represented by its physical name or address, called a **capability**.

- **An ability is the power, skill, means and opportunity to do something.**
- **A capability is the ability, aptitude or fitness to do something.**

Access Rights: Indirection

- Schemes that implement revocation for capabilities include the following
  - **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.
    - The object for a **capability** and its table entry must match.

- An object is often represented by its physical name or address, called a **capability**.

- **An ability is the power, skill, means and opportunity to do something.**
- **A capability is the ability, aptitude or fitness to do something.**

Access Rights

- Schemes that implement revocation for capabilities include the following
  - **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.

- An object is often represented by its physical name or address, called a **capability**.

- **An ability is the power, skill, means and opportunity to do something.**
- **A capability is the ability, aptitude or fitness to do something.**

Access Rights : Key

- 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.
- In key-based schemes, the operations of defining keys, inserting them into lists, and deleting them from lists should not be available to all users.

- An object is often represented by its physical name or address, called a **capability**.

- **An ability is the power, skill, means and opportunity to do something.**
- **A capability is the ability, aptitude or fitness to do something.**

Access Rights

- In particular, it would be reasonable to allow only the owner of an object to set the keys for that object.

- This choice, however, is a policy decision that the protection system can implement but should not define.

Topics Uncovered in this Session

- Implementation of Access Matrix
- Access Control
- Access Rights

- The Security Problem

The Security Problem

- **Protection**, as discussed earlier is strictly an internal problem deals with answering the question
  - How do we provide controlled access to programs and data stored in a computer system ?

The Security Problem

- **Security**, 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.

The Security Problem

- Large commercial systems containing payroll or other financial data are inviting targets to thieves.
- Systems that contain data pertaining to corporate operations may be of interest to unscrupulous competitors.
- Furthermore, loss of such data, whether by accident or fraud, can seriously impair the ability of the corporation to function.
- One can say that a system is **Secure** if its **resources** are **used** and **accessed** as **intended** under all circumstances.

The Security Problem

- Total security **cannot** be achieved.
- Mechanisms to make security breaches a rare occurrence, rather than the norm.
- Security violations or misuse of the system can be categorized as **intentional** or **malicious** or **accidental**.
- Protection mechanisms are the core of protection from accidents.

The Security Problem

- Several forms of accidental and malicious security violations typically occur
- **Intruder** and **Cracker** for those attempting to breach security.
- A **Threat** is the potential for a security violation, such as the discovery of a vulnerability
- An **Attack** is the attempt to break security

Accidental and Malicious Security Violations

- **Breach of Confidentiality:**
  - This type of violation involves unauthorized reading of data (or theft of information).
  - Typically, a breach of confidentiality is the goal of an intruder. Capturing secret data from a system or a data stream, such as credit-card information or identity information for identity theft, can result directly in money for the intruder.

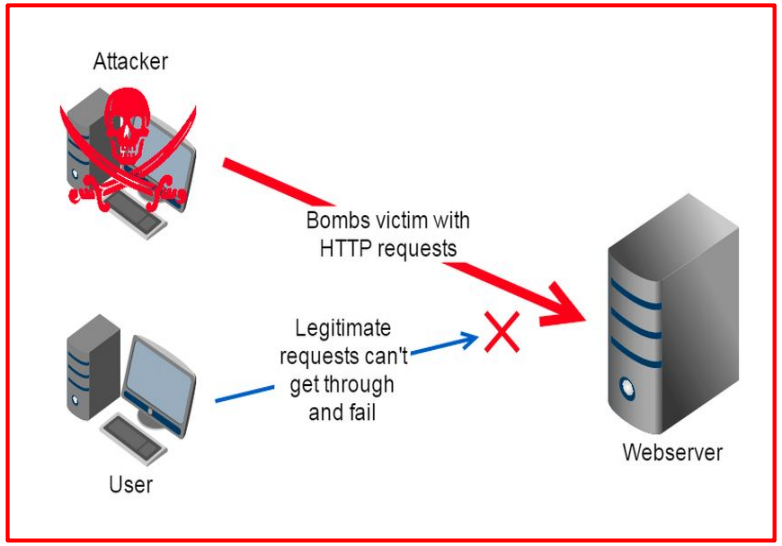
Accidental and Malicious Security Violations

- **Breach of integrity:**
  - This violation involves unauthorized modification of data.
  - Such attacks can, for example, result in passing of liability to an innocent party or modification of the source code of an important commercial application.

Oxymoron

Accidental and Malicious Security Violations

- **Breach of availability:**
  - This violation involves unauthorized destruction of data.
  - Some crackers would rather wreak havoc and gain status or bragging rights than gain financially.
  - Website defacement is a common example of this type of security breach

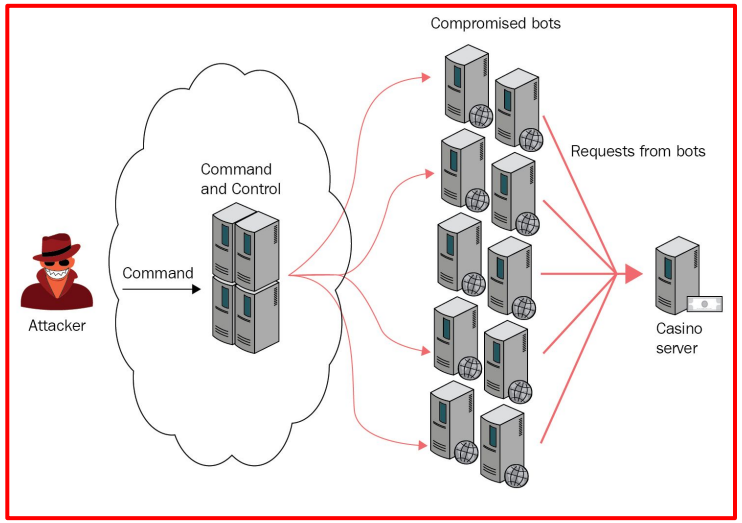


Accidental and Malicious Security Violations

- **Theft of service:**
  - This violation involves unauthorized use of resources.
  - For example, an intruder (or intrusion program) may install a daemon on a system that acts as a file server

Accidental and Malicious Security Violations

- **Denial of service:**
  - This violation involves preventing legitimate use of the system.
  - Denial-of-service (DOS) attacks are sometimes accidental.
  - The original Internet worm turned into a DOS attack when a bug failed to delay its rapid spread.



The Security Problem

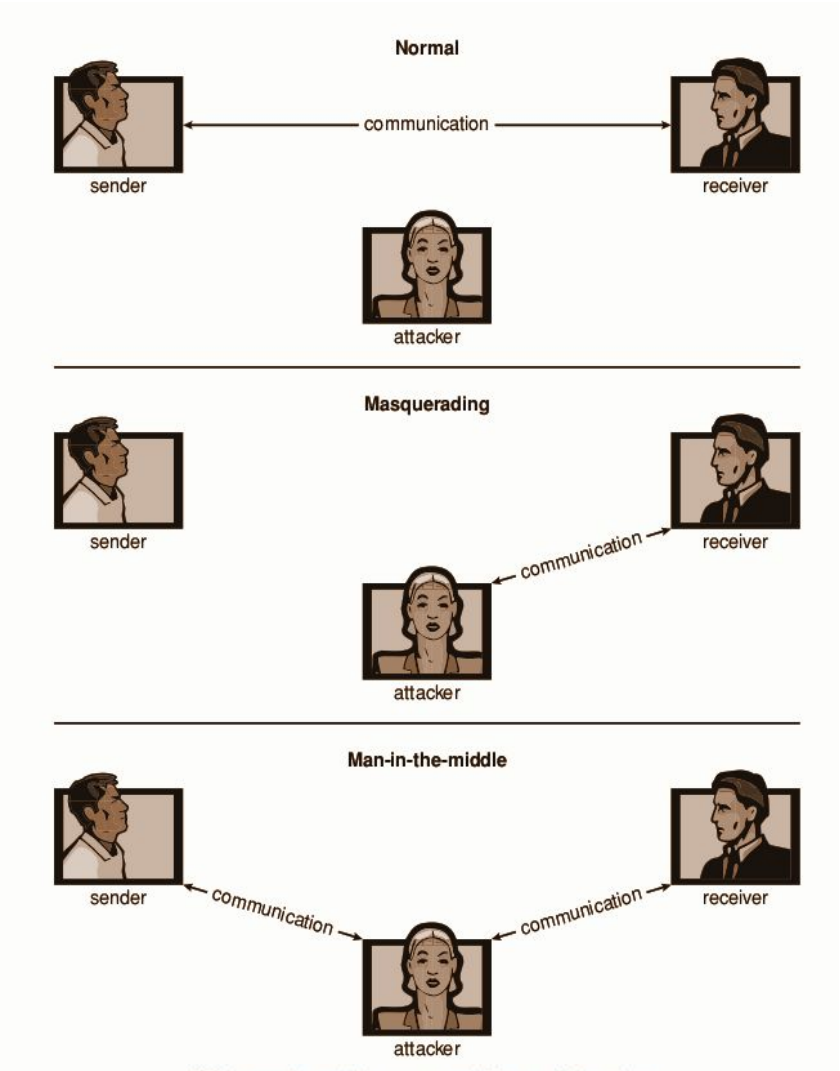
- Attackers use several standard methods in their attempts to breach security.

- **Masquerading:**

  - Here one participant in a communication pretends to be someone else another host or another person.

  - By masquerading, attackers breach authentication, the correctness of identification

  - They can then gain access that they would not normally be allowed or escalate their privileges and obtain privileges to which they would not normally be entitled.



Standard security attacks.

The Security Problem

- Attackers use several standard methods in their attempts to breach security.

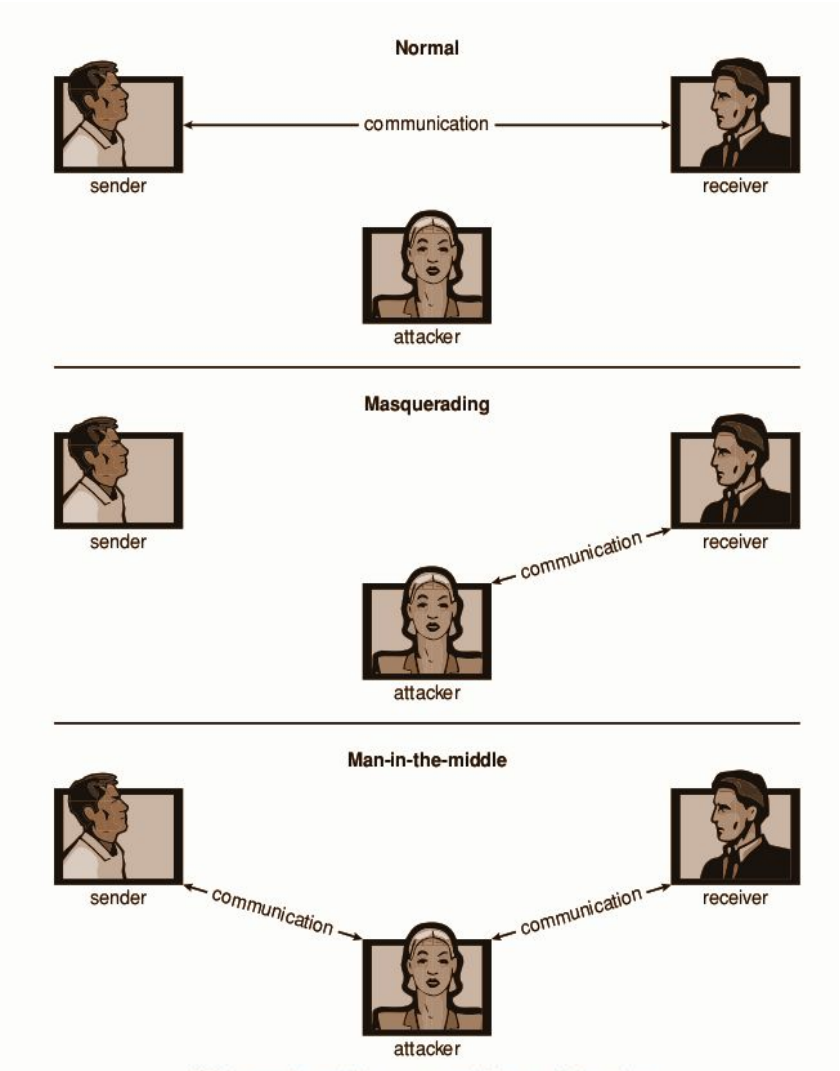
- **Replay Attack:**

  - Consists of the malicious or fraudulent repeat of a valid data transmission.

  - Sometimes the replay comprises the entire attack

  - As an example, in a repeat of a request to transfer money.

  - But frequently it is done along with message modification, again to escalate privileges.



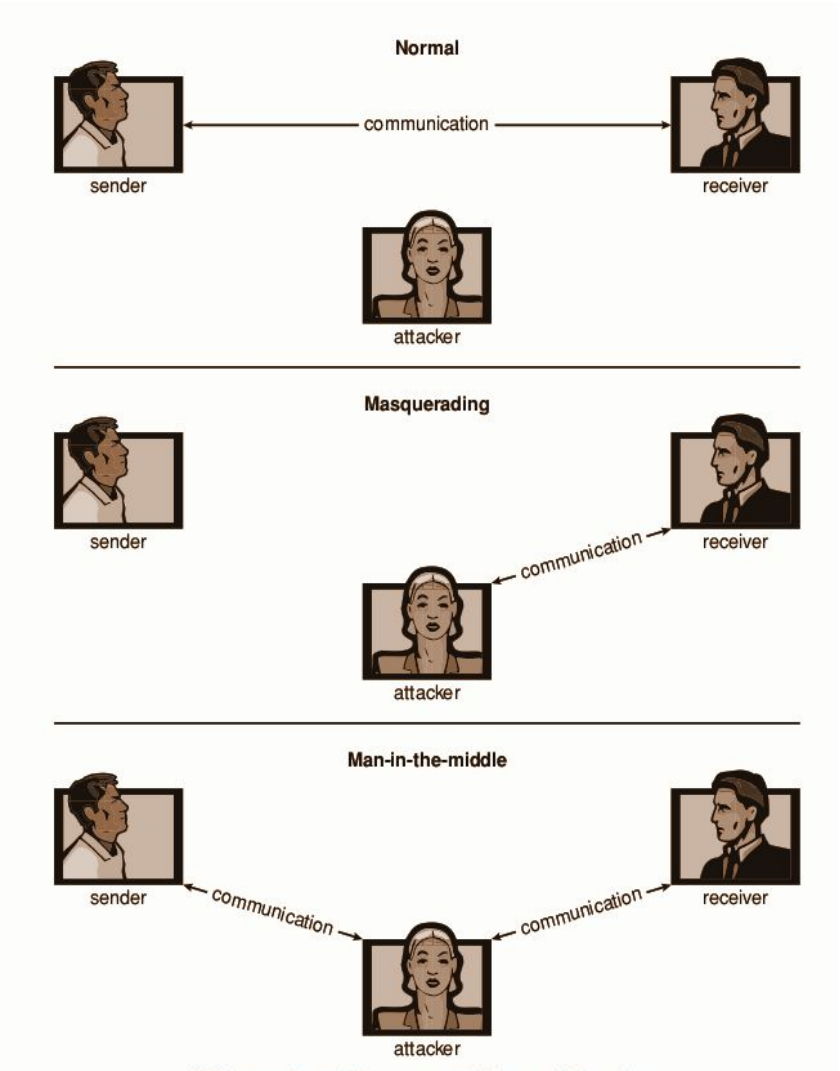
Standard security attacks.

The Security Problem

- Attackers use several standard methods in their attempts to breach security.

- **Man-in-the-middle attack:**

  - An attacker sits in the data flow of a communication, masquerading as the sender to the receiver, and vice versa.

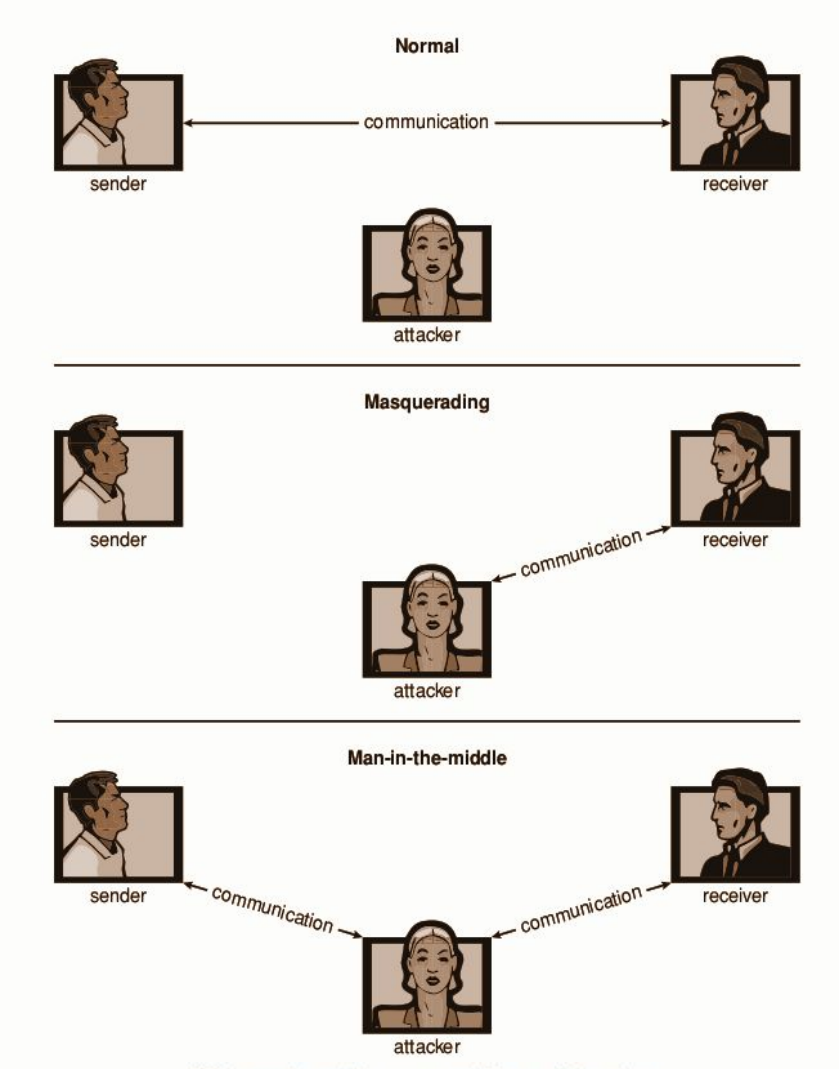


Standard security attacks.

The Security Problem

- Attackers use several standard methods in their attempts to breach security.

- **Session Hijacking:**
  - An active communication session is intercepted.



Standard security attacks.

The Security Problem

- To protect a system, one must take security measures at four levels

1. Physical:

- The site or sites containing the computer systems must be physically secured against armed or surreptitious entry by intruders.

- Both the machine rooms and the terminals or workstations that have access to the machines must be secured.

The Security Problem

- To protect a system, one must take security measures at four levels

2. Human:

- Authorization must be done carefully to assure that only appropriate users have access to the system.
- Even authorized users, however, may be “encouraged” to let others use their access (in exchange for a bribe, for example).
- They may also be tricked into allowing access via social engineering.

The Security Problem

- To protect a system, one must take security measures at four levels

2. Human:

- One type of social-engineering attack is **Phishing**.
- Here, a legitimate-looking email or web page misleads a user into entering confidential information.
- Another technique is **dumpster diving**, a general term for attempting to gather information in order to gain unauthorized access to the computer by looking through trash, finding phone books, or finding notes containing passwords, as an example.
- These security problems are management and personnel issues, not problems pertaining to operating systems.

The Security Problem

- To protect a system, one must take security measures at four levels

3. Operating system:

- The system must protect itself from accidental or purposeful security breaches.
- A runaway process could constitute an accidental denial-of-service attack.
- A query to a service could reveal passwords.
- A stack overflow could allow the launching of an unauthorized process.
- The list of possible breaches is almost endless.

The Security Problem

- To protect a system, one must take security measures at four levels

4. Network:

- Much computer data in modern systems travels over private leased lines, shared lines like the Internet, wireless connections, or dial-up lines.

- Intercepting these data could be just as harmful as breaking into a computer, and interruption of communications could constitute a remote denial-of-service attack, diminishing users' use of and trust in the system

The Security Problem

- Security at the first two levels must be maintained if Operating System security is to be ensured.
- A weakness at a high level of security (physical or human) allows circumvention of strict low-level (operating-system) security measures
- A chain is only as strong as its weakest link is especially true of system security.
- All of these aspects must be addressed for security to be maintained.
- Without the ability to authorize users and processes, to control their access, and to log their activities, it would be impossible for an operating system to implement security measures or to run securely
- Hardware protection features are needed to support an overall protection scheme.

The Security Problem

- Security within the operating system and between operating systems is implemented in several ways, ranging from passwords for authentication through guarding against viruses to detecting intrusions.

- The Security Problem

- The Program Threats

The Program Threats

- Processes, along with the kernel, are the only means of accomplishing some appropriate task on a computer.

- Writing a program that creates a breach of security, or causing a normal process to change its behavior and create a breach, is a common goal of crackers.

- While it is useful to log in to a system without authorization, it is quite a lot more useful to leave behind a **back-door** daemon that provides information or allows easy access even if the original exploit is blocked.

The Program Threats: Trojan Horse

- A code segment that misuses its environment is called a **Trojan horse**.
- Long search paths, such as are common on UNIX systems, **exacerbate** the Trojan horse problem.
- 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.
- If a user has "." in his / her search path, has set his / 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.
- The program will 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

The Program Threats: Trojan Horse

- 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 has apparently mistyped his password and tries again and is successful.
- Thus authentication key and password have been stolen by the login emulator, which was left running on the terminal by the thief.
- The emulator stored away the password, printed out a login error message, and exited; the user was then provided with a genuine login prompt.

The Program Threats: Trojan Horse

- 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 practice is an example of a general category of attacks known as covert channels, in which surreptitious communication occurs.

The Program Threats: Trojan Horse

- The installation of an innocuous seeming program on a Windows system could result in the loading of a spyware daemon.
- The spyware could contact a central site, be given a message and a list of recipient addresses, and deliver a spam message to those users from the Windows machine.
- This process continues until the user discovers the spyware.
- Frequently, the spyware is not discovered.
- In **2010**, it was estimated that **90 percent** of **spam** was being delivered by this method.
- This theft of service is not even considered a crime in most countries

The Program Threats: Trojan Horse

- **Spyware** is a micro example of a macro problem: violation of the principle of least privilege.
- Under most circumstances, a user of an operating system does not need to install network daemons.
- Such daemons are installed via two mistakes.
    - First, a user may run with more privileges than necessary for example, as the administrator, allowing programs that he / she runs to have more access to the system than is necessary. This is a case of human error—a common security weakness.
    - Second, an operating system may allow by default more privileges than a normal user needs. This is a case of poor operating-system design decisions.

The Program Threats: Trap Door

- The designer of a program or system might leave a hole in the software that only he / she is capable of using.
- This type of security breach (or **trap door**) was shown in the movie **War Games**.
- Programmers have been arrested for embezzling from banks by including rounding errors in their code and having the occasional half-cent credited to their accounts.
- This account crediting can add up to a large amount of money, considering the number of transactions that a large bank executes
- 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.
- Trap doors pose a difficult problem because, to detect them, we have to analyze all the source code for all components of a system.
- Given that software systems may consist of millions of lines of code, this analysis is not done frequently, and frequently it is not done at all!

The Program Threats: War Games Movie Trailer

- **WarGames** is a **1983** American Cold War science fiction techno-thriller film written by Lawrence Lasker and Walter F. Parkes and directed by John Badham.
- The film stars Matthew Broderick, Dabney Coleman, John Wood, and Ally Sheedy.
- The film was a box-office success, **costing $12 million** and grossing **$79 million**, after five months, in the United States and Canada

The Program Threats: Logic Bomb

- Consider a program that initiates a security incident only under certain circumstances.
- It would be hard to detect because under normal operations, there would be no security hole.
- However, when a predefined set of parameters was met, the security hole would be created. This scenario is known as a **logic bomb**.
- A programmer, for example, might write code to detect whether he/she was still employed; if that check failed, a daemon could be spawned to allow remote access, or code could be launched to cause damage to the site.

The Program Threats: Under Siege 2 Movie Trailer

- **Under Siege 2: Dark Territory** is a **1995** American action thriller film directed by Geoff Murphy, starring Steven Seagal as the ex-Navy SEAL, Casey Ryback

- The title refers to the railroading term that the subject train was travelling through dark territory, a section of railroad track that has no train signals and in which communications between train dispatchers and the railroad engineers were impossible

The Program Threats: Stack and Buffer Overflow

- The attack exploits a bug in a program.
- The bug can be a simple case of poor programming, in which the programmer neglected to code bounds checking on an input field.
- In this case, the attacker sends more data than the program was expecting.
- By using trial and error, or by examining the source code of the attacked program if it is available, the attacker determines the vulnerability and writes a program to do the following
    - Overflow an input field, command-line argument, or input buffer—for example, on a network daemon—until it writes into the stack.
    - Overwrite the current return address on the stack with the address of the exploit code loaded in step 3.
    - Write a simple set of code for the next space in the stack that includes the commands that the attacker wishes to execute —for instance, spawn a shell.

The Program Threats: Stack and Buffer Overflow

- The result of this attack program's execution will be a root shell or other privileged command execution.

- For instance, if a web-page form expects a username to be entered into a field, the attacker could send the user name, plus extra characters to overflow the buffer and reach the stack, plus a new return address to load onto the stack, plus the code the attacker wants to run.

- When the buffer-reading subroutine returns from execution, the return address is the exploit code, and the code is run.

The Program Threats: Viruses

- This vast variety of viruses has continued to grow.
- For example, in 2004 a new and widespread virus was detected.
- It exploited three separate bugs for its operation.
    - This virus started by infecting hundreds of Windows servers (including many trusted sites) running Microsoft Internet Information Server (IIS).
    - Any vulnerable Microsoft Explorer web browser visiting those sites received a browser virus with any download.
    - The browser virus installed several back-door programs, including a **keystroke logger**, which records everything entered on the keyboard (including passwords and credit-card numbers).
    - It also installed a daemon to allow unlimited remote access by an intruder and another that allowed an intruder to route spam through the infected desktop computer.

The Program Threats: Viruses

- A **Virus** is a fragment of code embedded in a legitimate program.
- Viruses are self-replicating and are designed to “infect” other programs.
- They can wreak havoc in a system by modifying or destroying files and causing system crashes and program malfunctions.
- 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.

The Program Threats: Viruses

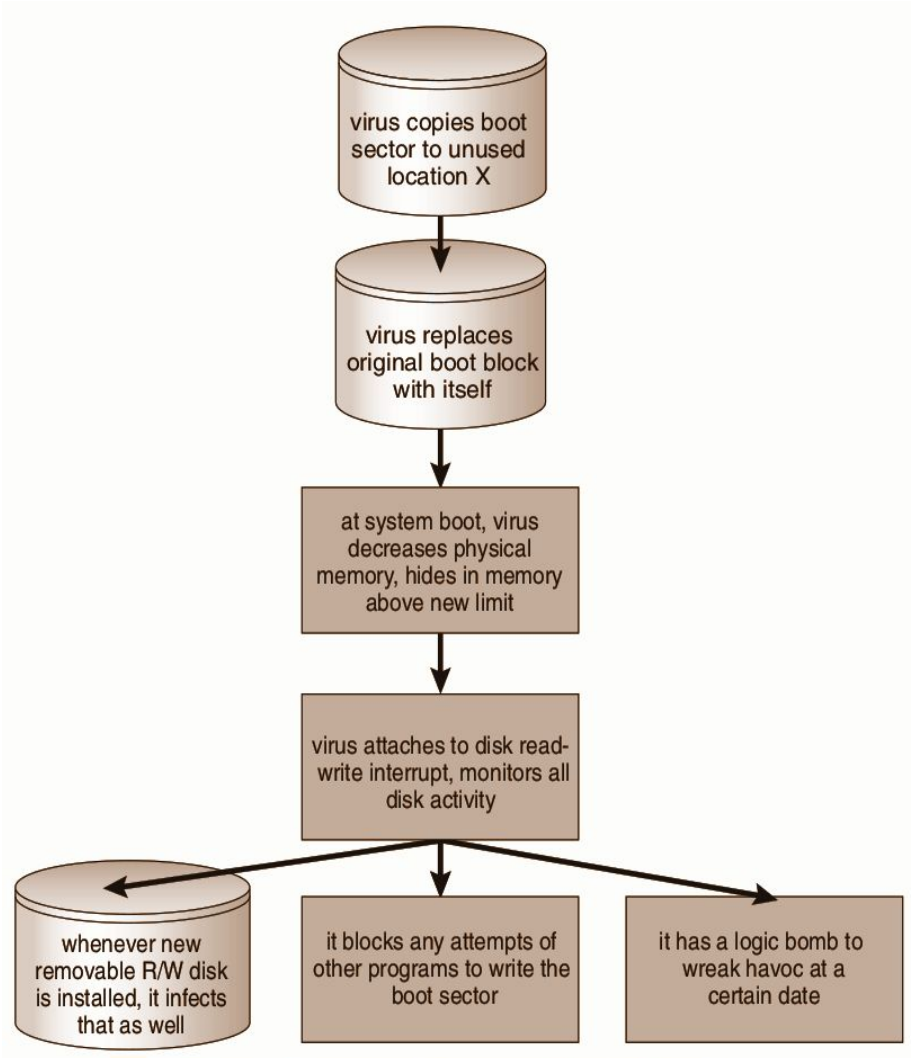
- Viruses are usually borne via email, with spam the most common vector.
- They can also spread when users download viral programs from Internet file-sharing services or exchange infected disks.
- Once a virus reaches a target machine, a program known as a **Virus Dropper** inserts the virus into the system.
- The virus dropper is usually a Trojan horse, executed for other reasons but installing the virus as its core activity.
- There are literally thousands of viruses, but they fall into several main categories.
- Note that many viruses belong to more than one category.

The Program Threats: 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.

The Program Threats: Viruses

- **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 and infects them.
  - These viruses are also known as **memory viruses**, because they do not appear in the file system.



A boot-sector computer virus

The Program Threats: Viruses

- **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.

The Program Threats: 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**:
  - A **polymorphic 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.

The Program Threats: Viruses

- **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.
- **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.

The Program Threats: Viruses

- **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 it 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.

The Program Threats: Viruses

- Viruses are the most disruptive security attacks, and because they are effective, they will continue to be written and to spread.
- An active security-related debate within the computing community concerns the existence of a **monoculture**, in which many systems run the same hardware, operating system, and application software.
- This monoculture supposedly consists of Microsoft products.
- One question is whether such a monoculture even exists today.
- Another question is whether, if it does, it increases the threat of and damage caused by viruses and other security intrusions.

The Program Threats: Viruses - Additional Input

- Antivirus and firewall software market is worth over $37 billion
- Over 350,000 pieces of **Malware** are detected every day
- **Malware**, short for malicious software, is a blanket term for viruses, worms, trojans and other harmful computer programs hackers use to wreak destruction and gain access to sensitive information.
- There were a record-breaking **10.52 billion** malware attacks in **2018**
- **E**ssential **S**ecurity against **E**volving **T**hreats - **ESET** holds the biggest market share among Windows antivirus solutions

The Program Threats: Viruses - Additional Input

- The average cost of antivirus software is around **$30 per year per user**
- Every day, at least **350,000** new malicious programs are detected
- There are more than **970 million pieces of malware** circulating the **internet right now**
- More **desktop users** have antivirus software installed than **laptop users**
- Only **half** of mobile devices in the US and half of that in the world are protected by an antivirus
- On a monthly basis, **only 4%** of Windows Defender users encounter a piece of malware
- Statistics show that **55% of PC software** around the world is **outdated**

The Program Threats: Viruses - Additional Input

- By **2024**, the mobile security market will be valued at **$42.18 billion**
- The latest smartphone antivirus statistics show there are **1.3 billion** mobile devices with antivirus software installed.
- **RiskTool** is the most common mobile malware, accounting for **54.06%** of all infected devices
- **Kaspersky** protected nearly **10 million** mobile users from viruses and malware in 2018
- More than **3.6 million** mobile infections were registered in **2016**
- As of September 2019, there have been more than **335 million installations** of **harmful apps** via **Google Play**
- Nearly **half** of popular Android antivirus programs fail at their job

The Program Threats: Viruses - Additional Input

- In 2018, there were a record-breaking **10.52 billion** malware attacks
- On average, antivirus software is only **25% successful** at detecting malware.
- Half of all malware attacks target the **United States, India Next ?**
- Every **fifth internet user** has been a victim of a malware attack at least once in their life
- **70%** of all infections come from **Microsoft Office** vulnerabilities
- **Trojans** are the most common malware

The Program Threats: Viruses - Additional Input

- **Ransomware** is used in just **0.3%** of all internet attacks
- **Ransomware** is malicious software that infects your computer and displays messages demanding a fee to be paid in order for your system to work again.
- Nearly half of all computers in **China** are **infected** with malware
- **Sweden** has the **lowest** ratio of infected computers: just **20%**
- The **highest ratio** of infected **mobile devices** is in **Pakistan**
- There were **116.5 million** malware attacks directed at mobile devices in **2018**
- The most widespread computer virus of all time caused **$10 billion** worth of damage in just a few days

The Program Threats: Viruses - Additional Input

- **Avast** and **AVG** are among the most popular and reliable free antivirus programs currently on the market.

- It's also worth looking into the free versions of otherwise premium antivirus programs like **Kaspersky, Bitdefender, and ESET**, although the latest antivirus statistics show that paid solutions are much better at detecting malware than the free ones.

- Credits: https://dataprot.net/statistics/antivirus-statistics/

- The Program Threats

Topic Outline

- The System Threats
- The Network Threats

The System and Network Threats

- Program threats typically use a breakdown in the protection mechanisms of a system to attack other programs.
- In contrast, **System** and **Network Threats** involve the abuse of services and network connections.
- System and network threats create a situation in which operating-system resources and user files are misused
- A system and network attack is used to launch a program attack, and vice versa
- Operating Systems strive to be secure by default

The System and Network Threats

- Changes in policy and mechanisms reduce the system's attack surface the set of ways in which an attacker can try to break into the system.
- It is important to note that masquerading and replay attacks are also commonly launched over networks between systems.
- These attacks are more effective and harder to counter when multiple systems are involved. For example, within a computer, the operating system usually can determine the sender and receiver of a message.
- Even if the sender changes to the ID of someone else, there may be a record of that ID change.
- When multiple systems are involved, especially systems controlled by attackers, then such tracing is much more difficult

The System and Network Threats

- Sharing secrets to prove identity and as keys to encryption is required for authentication and encryption, and sharing secrets is easier in environments such as a single operating system in which secure sharing methods exist.

- These methods include shared memory and interprocess communications.

The System and Network Threats: Worms

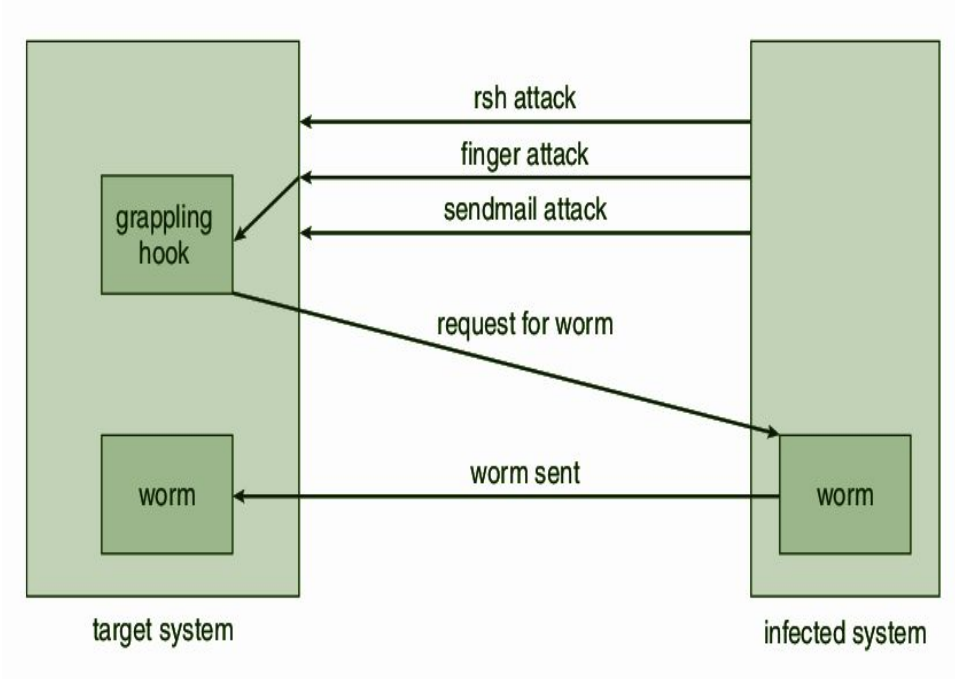
- A **worm** is a process that uses the spawn mechanism to duplicate itself.
- The worm spawns copies of itself, using up system resources and perhaps locking out all other processes.
- An event occurred in 1988 to UNIX systems on the Internet, causing the loss of system and system-administrator time worth millions of dollars
- At the close of the workday on November 2, 1988, Robert Tappan Morris, Jr., a first-year Cornell graduate student, unleashed a worm program on one or more hosts connected to the Internet.
- Targeting Sun Microsystems' Sun 3 workstations and VAX computers running variants of Version 4 BSD UNIX , the worm quickly spread over great distances.
- Within a few hours of its release, it had consumed system resources to the point of bringing down the infected machines.

The System and Network Threats: Worms

- The **worm** was made up of two programs, a **grappling hook** also called a **bootstrap** or **vector program** and the **main program.**

- Once established on the computer system under attack, the grappling hook connected to the machine where it originated and uploaded a copy of the main worm onto the hooked system

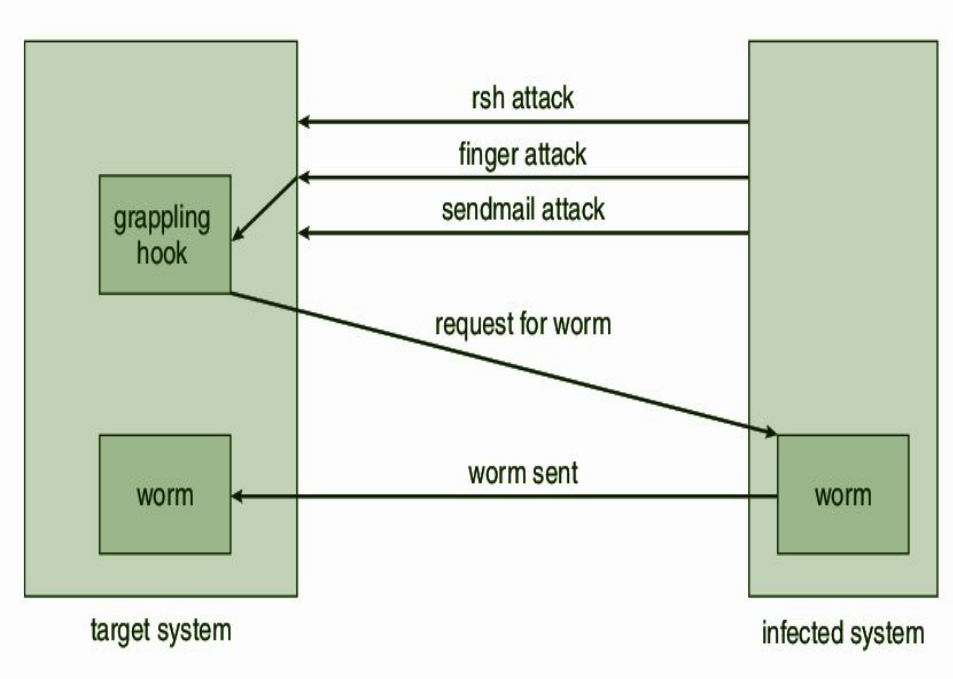
- The main program proceeded to search for other machines to which the newly infected system could connect



The Morris Internet worm.

The System and Network Threats: Worms

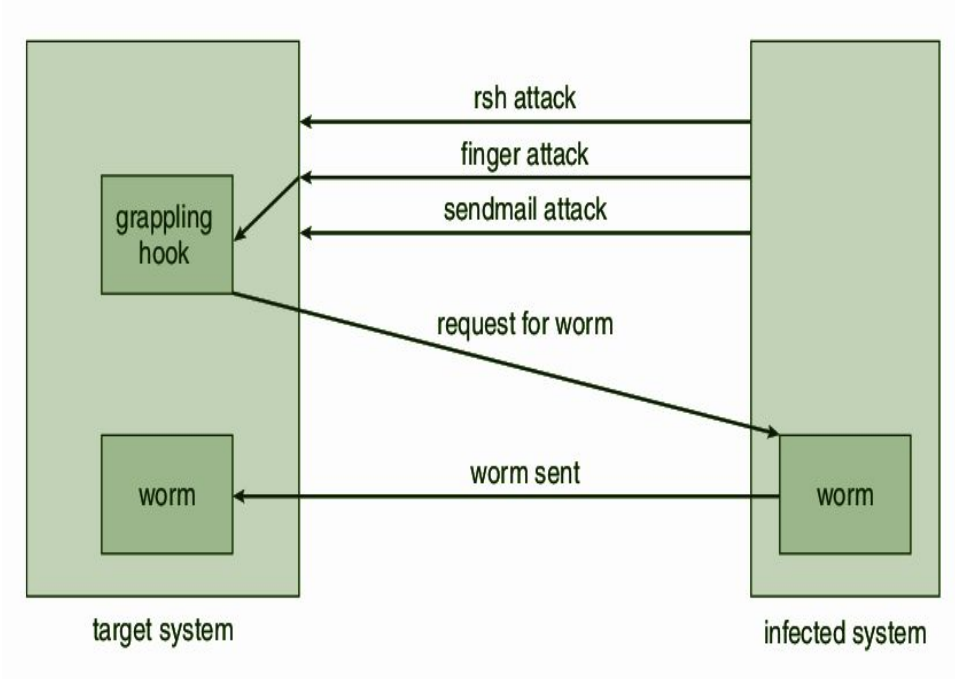
- In these actions, Morris exploited the UNIX networking utility rsh for easy remote task execution.
- By setting up special files that list host–login name pairs, users can omit entering a password each time they access a remote account on the paired list
- The worm searched these special files for site names that would allow remote execution without a password.
- Where remote shells were established, the worm program was uploaded and began executing as new.



The Morris Internet worm.

The System and Network Threats: Worms

- The **Finger utility** functions as an electronic telephone directory.
- The command finger user-name@hostname returns a person's real and login names along with other information that the user may have provided, such as office and home address and telephone number, research plan, or clever quotation.
- Finger runs as a background process (or daemon) at each BSD site and responds to queries throughout the Internet.

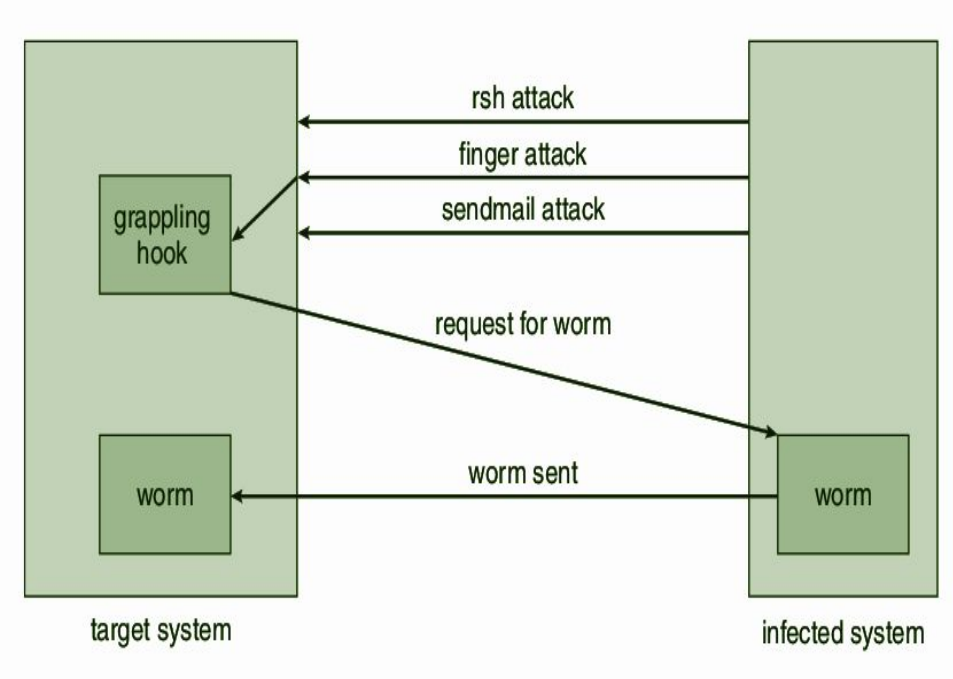


The Morris Internet worm.

The System and Network Threats: Finger Utility

The System and Network Threats: Worms

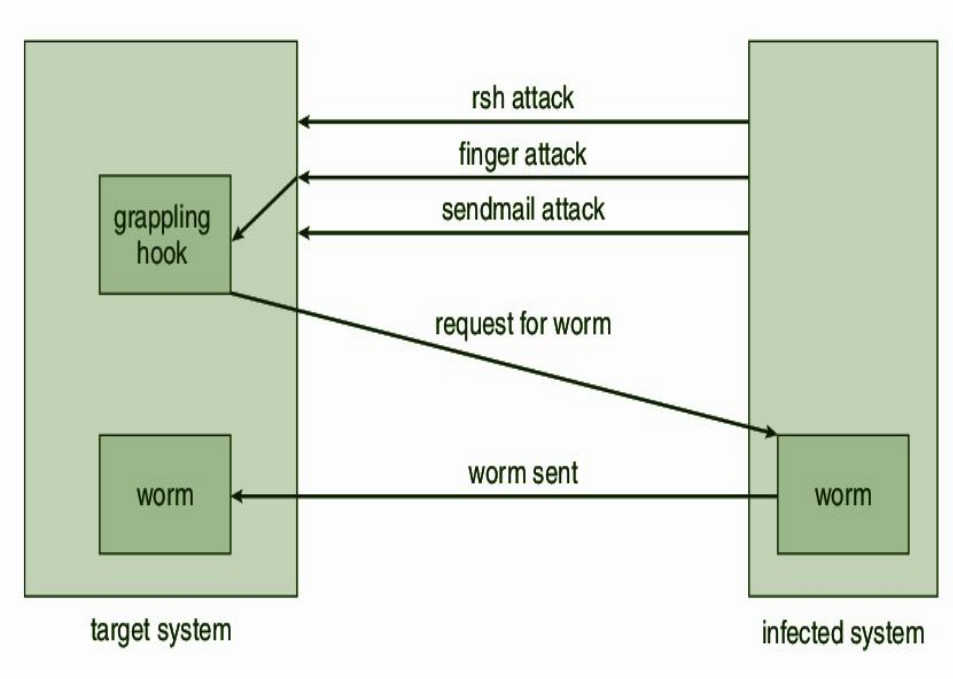
- The program queried finger with a 536-byte string crafted to exceed the buffer allocated for input and to overwrite the stack frame.
- Instead of returning to the main routine where it resided before Morris's call, the finger daemon was routed to a procedure within the invading 536-byte string now residing on the stack.
- The new procedure executed /bin/sh , which, if successful, gave the worm a remote shell on the machine under attack
- The bug exploited in sendmail also involved using a daemon process for malicious entry. sendmail sends, receives, and routes electronic mail.



The Morris Internet worm.

The System and Network Threats: Worms

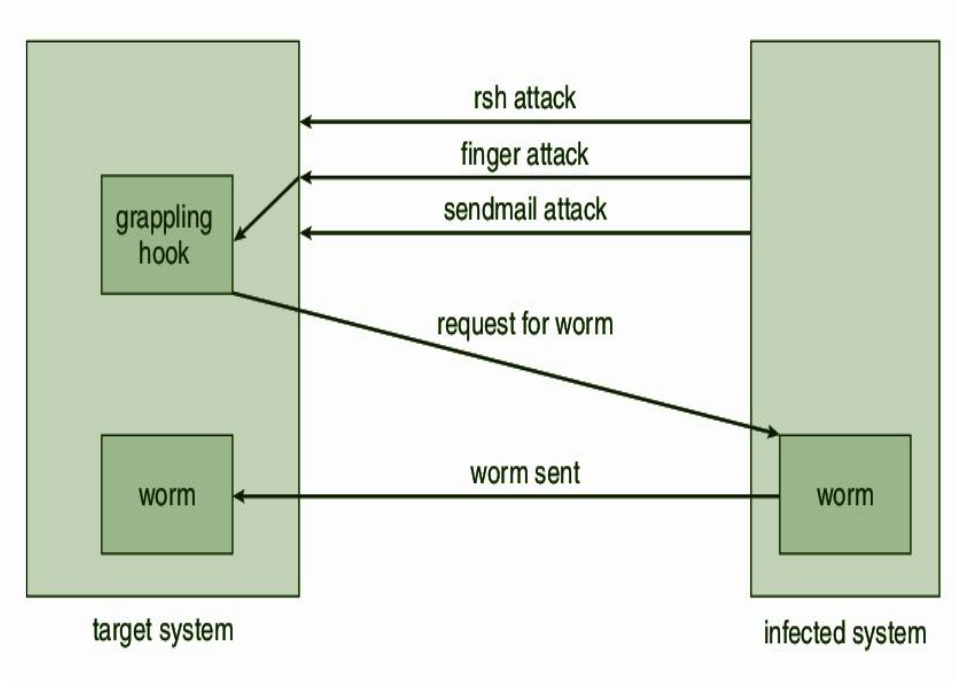
- Once in place, the main worm systematically attempted to discover user passwords.
- It began by trying simple cases of no password or passwords constructed of account–username combinations, then used comparisons with an internal dictionary of 432 favorite password choices, and then went to the final stage of trying each word in the standard UNIX online dictionary as a possible password.
- The very features of the UNIX network environment that assisted in the worm's propagation also helped to stop its advance.



The Morris Internet worm.

The System and Network Threats: Worms

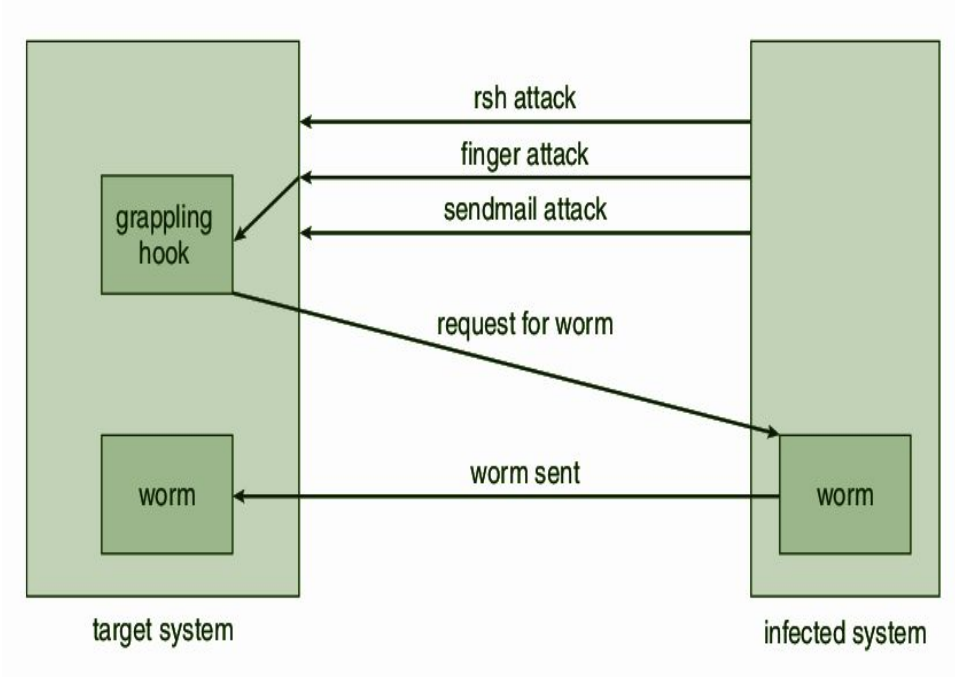
- By the evening of the next day, November 3, methods of halting the invading program were circulated to system administrators via the Internet.
- Within days, specific software patches for the exploited security flaws were available.
- The action has been characterized as both a harmless prank gone awry and a serious criminal offense.
- Based on the complexity of the attack, it is unlikely that the worm's release or the scope of its spread was unintentional.
- The worm program took elaborate steps to cover its tracks and to repel efforts to stop its spread.



The Morris Internet worm.

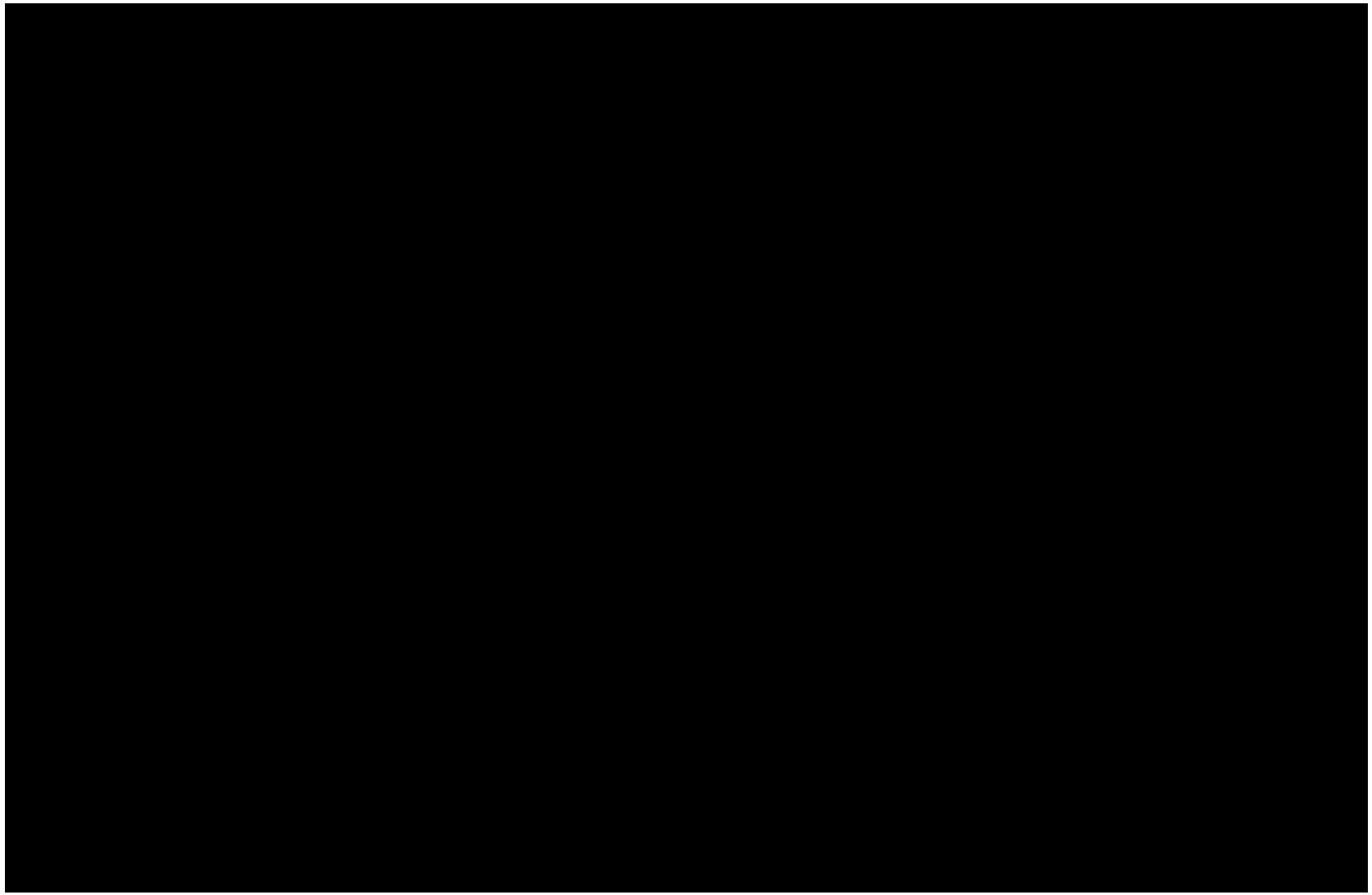
The System and Network Threats: Worms

- What is not open to speculation, however, is the legal outcome: a federal court convicted Morris and handed down a sentence of three years' probation, 400 hours of community service, and a **$10,000 fine**. Morris's legal costs probably exceeded **$100,000**



The Morris Internet worm.

The World's First Cyber Crime: The Morris Worm [KERNEL PANIC]



The System and Network Threats: Port Scanning

- **Port scanning** is **not** an **attack** but rather a means for a cracker to detect a system's vulnerabilities to attack.
- **Port scanning** typically is **automated**, involving a tool that attempts to create a TCP/IP connection to a specific port or a range of ports.
- A cracker could launch a port scanner to try to connect, say, to port 25 of a particular system or to a range of systems.
- If the connection was successful, the cracker (or tool) could attempt to communicate with the answering service to determine if the service was indeed sendmail and, if so,if it was the version with the bug
- A tool in which each bug of every service of every operating system was encoded.
- The tool could attempt to connect to every port of one or more systems.

The System and Network Threats: Port Scanning

- Frequently, the bugs are buffer overflows, allowing the creation of a privileged command shell on the system.
- From there, of course, the cracker could install Trojan horses, back-door programs, and so on
- Because port scans are detectable, they frequently are launched from **zombie systems**.
- **Zombies** make crackers particularly difficult to prosecute because determining the source of the attack and the person that launched it is challenging

The System and Network Threats: Denial of Service

- **Denial-of-service** attacks are aimed not at gaining information or stealing resources but rather at disrupting legitimate use of a system or facility.
- 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, which typically fall into two categories
- Attacks in the **first category** use so many facility resources that, in essence, no useful work can be done.
- The **second category** involves disrupting the network of the facility.

The System and Network Threats: Denial of Service

- 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.
- Even more difficult to prevent and resolve are **distributed denial-of-service (DDOS)** attacks
- These attacks are launched from multiple sites at once, toward a common target, typically by zombies.
- **DDOS** attacks have become more common and are sometimes associated with blackmail attempts.
- A site comes under attack, and the attackers offer to halt the attack in exchange for money.

The System and Network Threats: Denial of Service

- 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.
- Similarly, a firewall that automatically blocks certain kinds of traffic could be induced to block that traffic when it should not.
- These examples suggest that programmers and systems managers need to fully understand the algorithms and technologies they are deploying.
- Finally, computer science classes are notorious sources of accidental system DOS attacks.
- Consider the first programming exercises in which students learn to create sub processes or threads.
- A common bug involves spawning subprocesses infinitely.
- The system's free memory and CPU resources don't stand a chance.

Topic Outline

- The System Threats
- The Network Threats

- Case study : Windows File System - 2

Case study : Windows File System - Recovery

- In many simple file systems, a power failure at the wrong time can damage the file-system data structures so severely that the entire volume is scrambled.

- Many UNIX file systems, including UFS but not ZFS , store redundant metadata on the disk, and they recover from crashes by using the fsck program to check all the file-system data structures and restore them **forcibly** to a **consistent state**.

- Restoring them often involves deleting damaged files and freeing data clusters that had been written with user data but not properly recorded in the file system's metadata structures.

Case study : Windows File System - Recovery

- **NTFS** takes a different approach to file-system robustness.
- In NTFS , all file system data-structure updates are performed inside transactions.
- Before a data structure is altered, the transaction writes a log record that contains redo and undo information.
- After the data structure has been changed, the transaction writes a commit record to the log to signify that the transaction succeeded
- After a crash, the system can restore the file-system data structures to a consistent state by processing the log records, first redoing the operations for committed transactions and then undoing the operations for transactions

Case study : Windows File System - Recovery

- Periodically usually every 5 seconds, a checkpoint record is written to the log.
- The first time after system startup that an NTFS volume is accessed, NTFS automatically performs file-system recovery.
- Does not guarantee that all the user-file contents are correct after a crash.
- It ensures only that the file-system data structures (the metadata files) are undamaged and reflect some consistent state that existed prior to the crash.
- Current Versions of Windows does ensure the user files as well

Case study : Windows File System - Recovery

- The log has two sections
  - **Logging Area** which is a circular queue of log records,
  - **Restart Area** which holds context information, such as the position in the logging area where NTFS should start reading during a recovery.
    - The **Restart Area** holds two copies of its information, so recovery is still possible if one copy is damaged during the crash.
- The logging functionality is provided by the **log-file service**.
- Log-file service keeps track of the free space in the log file. If the free space gets too low, the log-file service queues pending transactions, and NTFS halts all new I/O operations
- After the in-progress operations complete, NTFS calls the cache manager to flush all data and then resets the log file and performs the queued transactions.

Case study : Windows File System - Security

- The security of an NTFS volume is derived from the Windows object model
- Each NTFS file references a security descriptor, which specifies the owner of the file, and an access-control list, which contains the access permissions granted or denied to each user or group listed.
- In normal operation, NTFS does not enforce permissions on traversal of directories in file path names.
- For compatibility with POSIX , these checks can be enabled.
- Prefix matching is an algorithm that looks up strings in a cache and finds the entry with the longest match—for example, an entry for \ home\sridatta would be a match for \home \sridatta \Desktop \Dynamic_AER.jpg
- The prefix-matching cache allows path-name traversal to begin much deeper in the tree, saving many steps. Enforcing traversal checks means that the user's access must be checked at each directory level.

Case study : Windows File System - Volume Management and Fault Tolerance

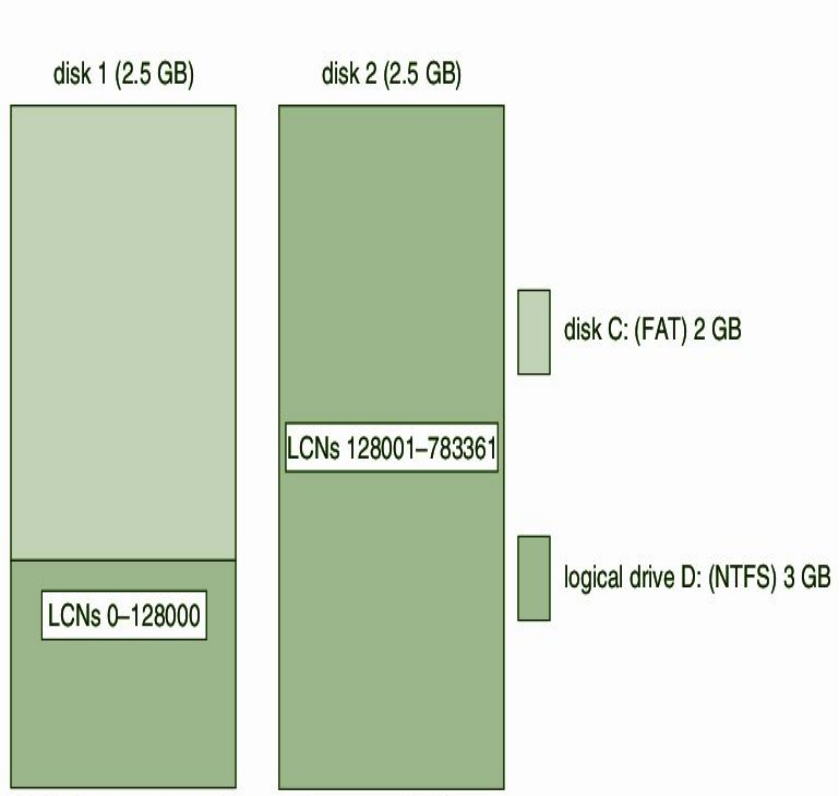
- **FtDisk** is the fault-tolerant disk driver for Windows.

- When installed, it provides several ways to combine multiple disk drives into one logical volume so as to improve **Performance**, **Capacity**, or **Reliability**

Case study : Windows File System - Volume Management and Fault Tolerance

Volume Sets and RAID Sets

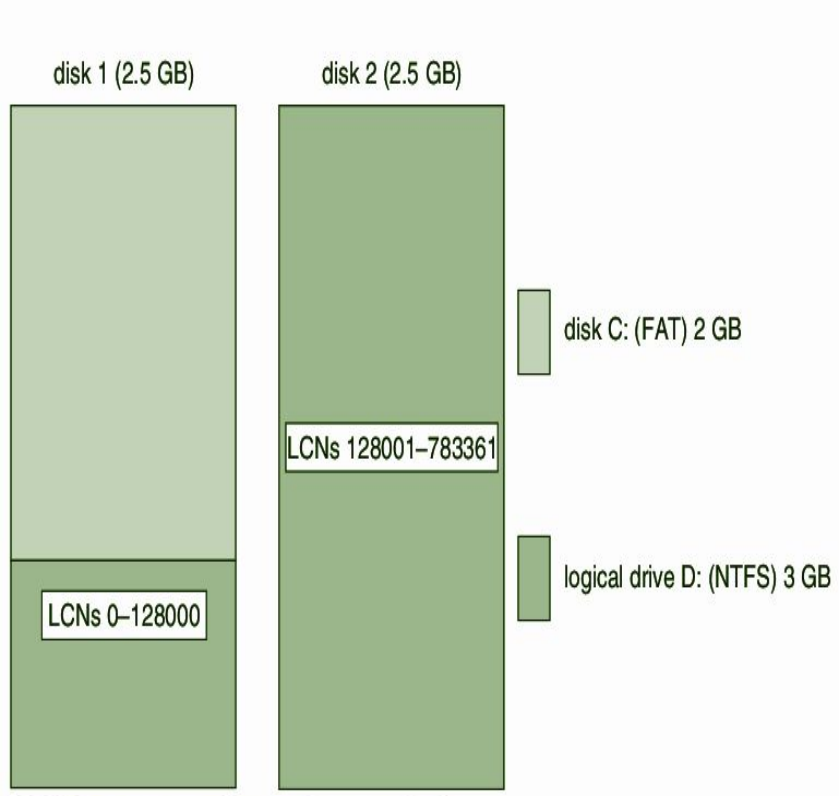
- One way to combine multiple disks is to concatenate them logically to form a large logical volume, as shown in Figure
- In Windows, this logical volume, called a **volume set**, can consist of up to 32 physical partitions.
- A volume set that contains an NTFS volume can be extended without disturbance of the data already stored in the file system.
- The bitmap metadata on the NTFS volume are simply extended to cover the newly added space.



Volume set on two drives

Volume Sets and RAID Sets

- Second way to combine multiple physical partitions is to interleave their blocks in round-robin fashion to form a stripe set.
- This scheme is also called RAID level 0, or **disk striping**.
- Windows also supports RAID level 5, stripe set with parity, and RAID level 1, mirroring.



Volume set on two drives

Sector Sparing and Cluster Remapping

- To deal with disk sectors that go bad, FtDisk uses a hardware technique called **Sector Sparing**, and NTFS uses a software technique called **Cluster Remapping**.
- **Sector sparing** is a hardware capability provided by many disk drives.
- When a disk drive is formatted, it creates a map from logical block numbers to good sectors on the disk.
- **Cluster Remapping** is a software technique performed by the file system.
- If a disk block goes bad, NTFS substitutes a different, unallocated block by changing any affected pointers in the MFT.
- NTFS also makes a note that the bad block should never be allocated to any file.

Sector Sparing and Cluster Remapping

- When a disk block goes bad, the usual outcome is a data loss.
- But sector sparing or cluster remapping can be combined with fault-tolerant volumes to mask the failure of a disk block.
- If a read fails, the system reconstructs the missing data by reading the mirror or by calculating the exclusive or parity in a stripe set with parity.
- The reconstructed data are stored in a new location that is obtained by sector sparing or cluster remapping.

Case study : Windows File System - Compression

- NTFS can perform data compression on individual files or on all data files in a directory.
- To compress a file, NTFS divides the file's data into compression units, which are blocks of 16 contiguous clusters.
- When a compression unit is written, a data-compression algorithm is applied.
- If the result fits into fewer than 16 clusters, the compressed version is stored.
- For sparse files or files that contain mostly zeros, NTFS uses another technique to save space. Clusters that contain only zeros because they have never been written are not actually allocated or stored on disk.
- Instead, gaps are left in the sequence of virtual-cluster numbers stored in the MFT entry for the file.
- When reading a file, if NTFS finds a gap in the virtual-cluster numbers, it just zero-fills that portion of the caller's buffer. This technique is also used by UNIX .

Case study : Windows File System - Compression

Mount Points, Symbolic Links, and Hard Links

- Mount points are a form of symbolic link specific to directories on NTFS that were introduced in Windows 2000.
- They provide a mechanism for organizing disk volumes that is more flexible than the use of global names like drive letters.
- A mount point is implemented as a **symbolic link** with associated data that contains the true volume name
- The links can be absolute or relative, can point to objects that do not exist, and can point to both files and directories even across volumes.
- NTFS also supports **hard links**, where a single file has an entry in more than one directory of the same volume.

Case study : Windows File System - Compression

Mount Points, Symbolic Links, and Hard Links

- A **hard link** acts as a copy (mirrored) of the selected file.
    - It accesses the data available in the original file.
    - If the earlier selected file is deleted, the hard link to the file will still contain the data of that file.
- A **Soft Link** also known as **Symbolic link** acts as a pointer or a reference to the file name.
    - It does not access the data available in the original file.
    - If the earlier file is deleted, the soft link will be pointing to a file that does not exist anymore.

Case study : Windows File System - Change Journal

- NTFS keeps a journal describing all changes that have been made to the file system.
- User-mode services can receive notifications of changes to the journal and then identify what files have changed by reading from the journal.
- The search indexer service uses the change journal to identify files that need to be re-indexed.
- The file-replication service uses it to identify files that need to be replicated across the network.

Case study : Windows File System - Volume Shadow Copies

- Windows implements the capability of bringing a volume to a known state and then creating a shadow copy that can be used to backup a consistent view of the volume.
- This technique is known as **Snapshots** in some other file systems.
- Making a shadow copy of a volume is a form of copy-on-write, where blocks modified after the shadow copy is created are stored in their original form in the copy.
- The server version of Windows uses shadow copies to efficiently maintain old versions of files stored on file servers

- Case study : Windows File System - 2

Course Syllabus Completed

for all the

Units (Unit 1, Unit 2, Unit 3, Unit 4, Unit 5)
(Next Week Revision of all 5 Units)

Thank you

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com