Storage Management, Protection and Security

Module 7

Dr. Naveenkumar J

Associate Professor,

PRP- 217 - 4

System Threats and Security

❑Operating System (OS) security refers to the **measures** taken to **protect** a computer system's **resources**—such as the **CPU, memory, disks, and data**—from **unauthorized access**, **malicious attacks**, and **interference**

❑ Security systems aim to ensure **confidentiality, integrity, and availability** of system resources against external threats.

System Threats and Security

- ❑ **System Threats**

  - ❑ A system threat is **any program or event** that has the **potential to cause damage to a computer system**.

  - ❑ These threats can be **intentional or accidental** and are typically launched by **non-users or unauthorized programs** from outside the operating system.

System Threats and Security

- ❑A **virus** is a self-replicating piece of code that attaches itself to legitimate programs. When the host program is executed, the virus also runs, potentially corrupting or destroying files and spreading to other programs.

- ❑A **Trojan horse** is a malicious program disguised as useful software. It performs a hidden, destructive function while appearing to do something benign. Unlike viruses, Trojans do not replicate themselves.

- ❑**Trap Door (or Backdoor)** is a secret entry point into a program that allows someone to gain access while bypassing normal security measures. It may be left by the original developer for testing or intentionally placed for malicious purposes

System Threats and Security

- ❑**Logic Bomb** This is a piece of code intentionally inserted into a software system that will trigger a malicious function when specified conditions are met. The trigger could be a specific date, time, or the absence of a particular file

- ❑**Buffer Overflow** This occurs when a program attempts to write more data into a fixed-length memory block (a buffer) than it can hold. The excess data overwrites adjacent memory, which can corrupt data, crash the program, or, in a targeted attack, execute malicious code with the privileges of the compromised application.

- ❑**Privilege Escalation** This attack involves a user with limited permissions exploiting a vulnerability to gain elevated access (e.g., administrator or root privileges).

Policy vs. Mechanism

- In the context of OS security, policy and mechanism are distinct concepts that work together to enforce security rules

Policy	Mechanism
A security policy defines what needs to be secured. It is a set of high-level rules and objectives that specify the desired security goals for a system. Policies are about the goals, not the implementation.	A security mechanism is the implementation or tool used to enforce a policy. It provides the how—the specific methods, functions, and hardware used to achieve the security goals defined by the policy.
A company policy might state, "Only employees from the Finance department are allowed to view and modify payroll files." This rule does not specify how to enforce this restriction.	To enforce the policy, the operating system could use an Access Control List (ACL) as a mechanism. The ACL for the payroll files would be configured to grant read and write permissions only to users who are members of the "Finance" group.

Access vs. Authentication

- ❑ Authentication and access (via authorization) are sequential steps in a secure process. Authentication confirms identity, while access control determines what an identified user can do.

Authentication	Authorization (Governs Access)
To verify a user's identity and ensure they are who they claim to be . It answers the question, **"Who are you?"**	To determine the permissions and access rights of an authenticated user . It answers the question, **"What are you allowed to do?"**
Occurs before authorization . The user provides credentials that the system validates.	**Occurs after successful authentication** . The system checks the user's permissions against the requested resource.
Requires information like a username/password, biometric data (fingerprint, retina scan), or a security token/key .	Relies on policies, roles, or permissions (e.g., read, write, execute) assigned to the user's identity
Users can typically manage their own authentication credentials, such as changing a password .	Permissions are granted by a system administrator or resource owner and cannot be changed by the user .

- ❑ Access Control is the practical application of authorization. Once a user is authenticated and their authorization level is determined, the access control mechanism enforces this policy, either granting or denying the user's request to access a resource.

System Protection

- ❑System protection is concerned with **controlling access to resources** within a computer system.

- ❑ The goal is to ensure that **processes, users, and programs can only access the objects (like files, devices, or memory segments) for which they have been granted authorization, and only in the manner specified**.

- ❑Two fundamental models for this are the
  - ❑Access Matrix and
  - ❑Capability-Based Systems.

System Protection : Access Matrix

- ❑ An Access Matrix is a conceptual model used to define the access rights of subjects to objects.
  - ❑**Subjects**: These are the active entities that request access, such as **users or processes**. The rows of the matrix represent subjects (often grouped into domains of execution).
  - ❑**Objects**: These are the passive resources that need protection, such as **files, printers, or other devices**. The columns of the matrix represent objects.
  - ❑**Access Rights**: The cells of the matrix, **Access(i, j)**, contain the **set of operations that a subject in domain i can perform on object j**. Examples of rights include read, write, execute, and own.

System Protection : Access Matrix

- Imagine a small company with three users (acting as subjects) and four resources (objects):
  - **Subjects/Domains:**
    - D1: Alice (a project manager)
    - D2: Bob (a finance analyst)
    - D3: Charlie (a developer)
  - **Objects:**
    - File1: Project_Plan.docx
    - File2: Financials.xlsx
    - File3: Source_Code.c
  - **Object1: Printer**

System Protection : Access Matrix

❑The security policy can be represented by the following Access Matrix:

Domain / Object	File1 (Project Plan)	File2 (Financials)	File3 (Source Code)	Object1 (Printer)
D1: Alice	read,write	read		print
D2: Bob	read	read,write		print
D3: Charlie	read		read,write	

❑ The Access Matrix itself is an abstract model. In a real OS, it is too large and sparse to be stored as a simple table. Instead, it is implemented in one of two primary ways: Access Control Lists or Capability Lists

System Protection : Access Matrix

- ❑ **Access Control Lists (ACLs)** - A Column-based View: The matrix is broken down by columns (objects). Each object has a list (an ACL) attached to it that specifies which subjects have what rights.

- ❑Example: The Financials.xlsx file would have an ACL like this:

  (D1:Alice, {read}), (D2:Bob, {read, write}).

- ❑When Alice tries to write to the file, the system checks this list, sees she only has read permission, and denies the operation.

System Protection : Capability Lists

- ❑**Capability Lists** - **A Row-based View**: The matrix is broken down by rows (subjects).
- ❑ Each subject has a list of "capabilities," where each capability specifies an object and the access rights for it. This forms the basis for capability-based systems.
- ❑Example: Alice would possess a capability list:

(File1, {read, write}), (File2, {read}), (Object1, {print}).

Capability-Based Systems

- ❑A capability-based system takes the concept of a **capability list** and makes it the **central security mechanism**. In this model, a capability is like a key that gives the holder specific rights to an object.

- ❑ A capability is a data structure that contains two key pieces of information:
  - ❑ A unique pointer or identifier to an object.
  - ❑ The set of access rights for that object (e.g., read, write).

- ❑The operating system kernel ensures that capabilities cannot be forged or modified by user-level processes. The core principle is: possession of a capability is proof of the right to access the object.