Operating System

Assignment-5

Q.1. What are the main goal of protection in an operating system?

⇒ Protection in an operating system refers to the mechanism that control the access of processes to system system resources. Its main goal are:

(i) Prevent Unauthorized Access:
The OS ensures that only valid users and processes can access certain files, memory areas, or devices. This protects sensitive information and prevents malicious actions.

(ii) Ensure Controlled Sharing:
Multiple users may need to share resources like printers, files or databases, protection ensures sharing happens safely without violating privacy or integrity.

(iii) Maintain System Integrity:
protection mechanism prevent user or programs from accidentally (bugs) or intentionally altering import system data, OS code, or hardware settings.

(iv) Guarantee User privacy:
Every users data must remain isolated from others unless permission is given. Protection ensures confidentiallity.

(v) Support System Reliability and Stability:
If processes could execute harmful operations freely, the system could crash. Protection keep resources safe, ensuring the system runs reliably.

(vi) Enforce Access policies:
organizations often have security policies. OS protection implement them consistently for all users.

Q.2. Define the Domain of protection with a suitable example.

⇒ A Domain of Protection defines the set of resources (objects) and operations (rights) a processor or user is allowed to perform on those resources.

It essentially specifies "who can do what" in a system.

Characteristics of a Domain:

- A domain contains objects like files, memory segments, printers, devices, etc.
- Each object has access rights like read, write, execute, print, delete.
- A processes runs with a domain, and its permissions depend on that domain.

Example:

Suppose a user process runs in a Domain D1. The domain may allow:

Object	Allowed Rights
file F1	Read, write
file F2	Read only
printer P1	print
program P2	Execute

Here, the collection of all these rights is called the domain of protection for that process.

Domains can also change dynamically. For example, in Linux, using the sudo command temporarily switches a user to a domain with higher privileges.

Q3. What is an Access Matrix in operating systems?

The **Access Matrix** is a conceptual model used to describe and manage protection in a computer system. It shows **which subjects** (users/processes) have **what access rights** to which **objects** (files, printers, memory, devices).

Components:

- **Rows** → Subjects (users, processes)
- **Columns** → Objects (files, devices)
- **Each cell** → Contains the allowed rights (e.g., read, write, execute)

Example of Access Matrix

	File1	File2	Printer	Memory
User1	R, W	R	Print	—
User2	R	—	—	Exec

The access matrix is not implemented directly but helps design the protection model.

Q4. How can the Access Matrix be implemented in practice?

The Access Matrix is only theoretical. Real systems use these practical implementations:

---

1. Access Control Lists (ACLs)

- Stored **per object**.
- Each object has a list of subjects and their access rights.
- Example: NTFS permissions in Windows.

Advantages: Easy to know “who can access this object?”

---

2. Capability Lists (C-Lists)

- Stored **per subject**.
- Each subject has a list of objects it can access and what rights it has.

Advantages: Easy to know “what objects can this subject access?”

---

3. Lock–Key Mechanism

- Each object has a **lock** (a list of required rights).
- Each domain or process has a **key** (a list of allowed rights).
- A process can access an object only if its key matches the lock.

---

4. Hybrid Implementations

Practical systems often use combinations of ACLs and capabilities (e.g., modern OS and cloud systems).

Q5. What is Revocation of Access Rights, and why is it necessary?

Revocation means **withdrawing or removing access rights** previously granted to a user, process, or domain.

Why Revocation is Necessary:

1. **Security:** When a user leaves an organization, their access must be removed.
2. **Temporary Permissions:** Sometimes rights are granted for specific tasks only.

3. **Prevent Misuse:** If a user is found misusing rights, they must be revoked.
4. **Dynamic Environments:** In distributed and cloud systems, rights must be updated frequently.
5. **Error Correction:** Sometimes rights were granted accidentally.

Types of Revocation:

- **Immediate vs. Delayed** – Rights removed instantly or after a time.
- **Selective vs. General** – Remove rights from specific users or from all users.

Q6. Explain Language-based Protection in operating systems.

Language-based protection uses **programming languages and their features** to enforce security rules and prevent unauthorized access to resources.

Instead of relying only on OS mechanisms, the language itself restricts what code can do.

Key Techniques:

1. Type Checking

Prevents operations on incompatible data types (e.g., Java, C#).
This avoids memory corruption.

2. Encapsulation

Classes and objects hide internal data.
Access is allowed only through safe methods.

3. Memory Safety

Languages like Java, Python, and Rust prevent direct memory access, reducing vulnerabilities.

4. Sandboxing

Code runs in a restricted environment.
Example: Web browser JavaScript executes in a sandbox.

5. Runtime Checks

Languages can enforce rules during execution (e.g., array bounds checking).

Benefit

Reduces the chances of buffer overflows, illegal memory access, and other vulnerabilities.

Q7. What are Capability-based Systems? Give one real-world example.

A **Capability-based System** is a protection model where access to objects is managed using **capabilities**.

What is a Capability?

A capability is a **token or key** that specifies:

- The object reference
- The allowed operations (read/write/execute)

If a process holds a capability, it can access the object.

Properties:

- Cannot be forged
- Must be untransferable unless allowed
- Stored securely by the system

Real-world Examples:

1. **UNIX File Descriptors**
   A process can read/write a file only if it holds its file descriptor — this is a capability.
2. **AWS IAM Access Tokens**
   Cloud systems issue tokens specifying what a user or service can access.

Q8. What is the Security Problem in computer systems?

The **Security Problem** refers to protecting computer systems from:

- Unauthorized access
- Data theft
- Misuse
- Destruction of data
- Disruption of services

Three major goals (CIA Triad):

1. Confidentiality

Ensures information is accessible only to authorized users.

2. Integrity

Ensures data is accurate and not tampered with.

3. Availability

Ensures services and data are available whenever needed.

Security threats include:

- Malware
- Hackers
- Insider threats
- Phishing attacks
- System vulnerabilities

The main challenge is to design a system that is functional while ensuring security.

Q9. Explain the role of Intrusion Detection in cybersecurity.

Intrusion Detection refers to the process of monitoring system and network activities to detect suspicious behavior or unauthorized access.

Roles and Functions:

1. Identifies Attacks

Detects malicious activities such as port scans, malware, or unauthorized login attempts.

2. Monitors System and Network Traffic

Analyses logs, packets, and system calls to find anomalies.

3. Alerts Administrators

Sends notifications when threats are detected.

4. Helps in Incident Response

Assists security teams in reacting quickly to minimize damage.

Types of Intrusion Detection Systems (IDS):

1. **Host-Based IDS (HIDS)**
    - Works on individual computers
    - Monitors logs, file changes, system calls
2. **Network-Based IDS (NIDS)**
    - Monitors network traffic
    - Can detect DOS attacks, malware propagation, etc.

IDS is a critical part of modern cybersecurity systems.

Q10. What is Cryptography? Differentiate between Symmetric and Asymmetric encryption.

Cryptography

Cryptography is the science of securing information by converting it into unreadable form, ensuring privacy, authentication, and data integrity.

It protects data while:

- Storing
- Processing
- Transmitting

Difference Between Symmetric and Asymmetric Encryption

Feature	Symmetric Encryption	Asymmetric Encryption
Number of Keys	Uses **one shared key**	Uses **two keys**: Public key (encrypt), Private key (decrypt)
Speed	Very fast	Slower due to heavy mathematical operations
Security	Key distribution is difficult	More secure key exchange
Best Use Case	Bulk data encryption (files, databases)	Digital signatures, secure key exchange
Examples	AES, DES, Blowfish	RSA, ECC

How They Work:

- **Symmetric:**
  Sender encrypts with key K → Receiver decrypts with same key K.
- **Asymmetric:**
  Sender encrypts with **receiver's public key** → Only receiver's **private key** can decrypt.