Protection:

System protection:

→ protection refers to a mechanism for controlling the access of progra processes, or users to the resources defined by a computer system.

→ protection ensures that the resources of the computer are used in a proper way.

→ It ensure that each object accessed correctly and only by those processes that are allowed to do.

→ OS designer faces challenge of creating a protection scheme that cannot be by passed by any software that may be created in the future.

Goals of protection:

→ We need to provide protection for several reasons. The most obvious is the need to prevent the bad, intentional violation of an access restriction by user.

→ an unprotected resource cannot defined against misuse by c unauthorized 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 implementation of the policies governi resource use.

→ These policies can be established in a variety of ways. Some are fixed in the design of the system. while other are formulated by the management of a system. still others are defined by the individual users to protect their own files and programs.

Principle of protection:

→ The 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.

Access control:

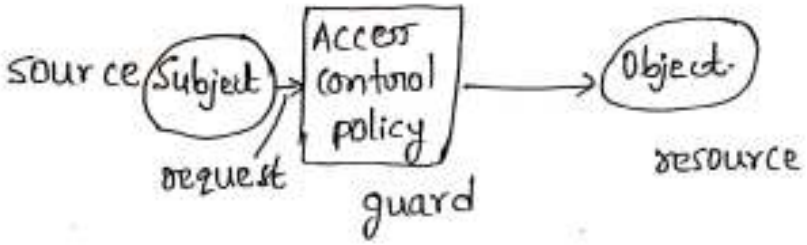
→ It allows the activities of valid users, mediating every attempt by a user to access a resource in the system.

→ Object: An entity that contains or receives information access to an object potentially implies access to the information it contains.

Eg:- file, programs, printer, disk etc.

→ Access rights: The permission granted to a user to perform an operation.

Eg:- Read, write, execute etc.



Domain of protection:

→ Domain is a collection of objects and a set of access rights for each of the objects

→ A process operates within a protection domain that specifies the resources that the process may access.

→ Each domain defines a set of objects and the types of operations that may be invoked on each object.
→ The ability to execute an operation on an object is an access right.
→ The system will consists of such multiple domains each having certain predefined access right on different object.
→ During execution of the process it can change the domain this is called domain switching.
→ A domain can be realized in a variety of ways

Each user may be a domain, in this case the set of objects that can be accessed depends on the identity of the user.

Each process may be a domain, in this case, the set of objects that can be accessed depends on the identity of the process.

Access Control Matrix:

→ protection model can be viewed abstractly as a matrix, called an access matrix.
→ rows represent domains, columns represent objects.
→ Each entry in the matrix consists of a set of access rights.
→ The entry access (i, j) defines the set of operations that a process executing in Domain, can invoke on object.

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	

→ If a process in domain D_i tries to do operation on object O_j then operation must be in the access matrix.

use of Access matrix:

Access matrix design separates mechanism from policy mechanism: operating system provides access matrix + rules it ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced.

policy: user dictates policy. who can access what object and in what mode.

object / domain	F_1	f_2	f_3	laserprinter	P_1	P_2	P_3	P_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Access matrix with domains as objects

object / domain	f_1	f_2	f_3
D_1	execute		write
D_2	execute	read*	execute
D_3	execute		

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

object / domain	f_1	f_2	f_3
D_1	owner execute		write
D_2		read owner	read* owner write
D_3	execute		

object / domain	f_1	f_2	f_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

Access matrix with owner rights

Implementation of Access Matrix:

Each column = Access control list for one object

Domain D_1 = Read, write
Domain D_2 = Read
Domain D_3 = Read

Each Row = capability List
for each domain, what operations allowed on what objects.

Object 1 – Read
Object 4 – Read, Write, Execute
Object 5 – Read, Write, Delete, copy

There are several methods to implement access matrix.

Global table:

→ The simplest implementation of the access matrix
→ matrix table consists of set of ordered triples $\langle$ Domain, Object, right set $\rangle$.
→ whenever an operation m is executed on object O_j with domain D_i then a global table is searched for triple $\langle D_i, O_j, R_k \rangle$. if triple is found, operation is allowed to continue otherwise it deny access.

Disadvantages:

- usually large thus cannot be kept in main memory
  Additional I/O is needed.
- It must have separate entry in domain

Access Lists for Objects:

Access control list (ACL) focus on the object

ACL $\equiv$ column of the access control matrix

$O_j \langle D_i, R_k \rangle$

→ ACL defines all domain with non empty set of access rights for that object.
→ Access rights are often defined for groups of users because individual subjects may create a huge list.
→ ACL is stored in the directory entry of the file

Capability List:

Capability list focus on the object
capability list = rows of the access control matrix.

→ capability is pointer to the object, contain address of the object.
→ Each domain has its capability list which contain list of capability together with operation allowed.
→ capability list is itself a protected object
- maintained by operating system
- Accessed by user only indirectly.

Capability Based system:

Hydra:

fixed set of access rights known to and interpreted by the system. Analysis of user defined rights performed only by user's program. System provides access protection for use of these rights.

Cambridge CAP system:

Data capability provides standard read, write, e of individual storage segments associated with object so capability interpretation left to the subsystem, through its protected procedures.

Language Based protection:

Specification of protection in a programming Lang allows the high-level description of policies for the allocation and use of resources.

Language implementation can provide software for protection enforcement when automatic hardware supported checking is unavailable.

Revocation of access rights:

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

- Immediate versus delayed:
- Selective versus general
- partial versus total
- Temporary versus permanent

Revocation capabilities include following

- Reacquisition
- Back pointers
- indirection
- keys