CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTEOF ENGINEERING

Bachelor of Engineering (Computer Science & Engineering)

Operating System (20CST/ITT-313)

Subject Coordinator: Er. Puneet kaur(E6913)

System Protection and Security

Goals of Protection

- In a protection model, computer consists of a collection of objects, hardware or software
- Each object has a unique name and can be accessed through a well-defined set of operations
- Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so

Principles of Protection

- **Guiding Principle – Principle of Least Privilege**
  - Programs, users and systems should be given just enough **privileges** to perform their tasks
  - Privilege Limits are damaged if entity has a bug.
  - Privilege Can be static (during life of system, during life of process)
  - Privilege Can be dynamic (changed by process as needed) – **domain switching**, **privilege escalation**

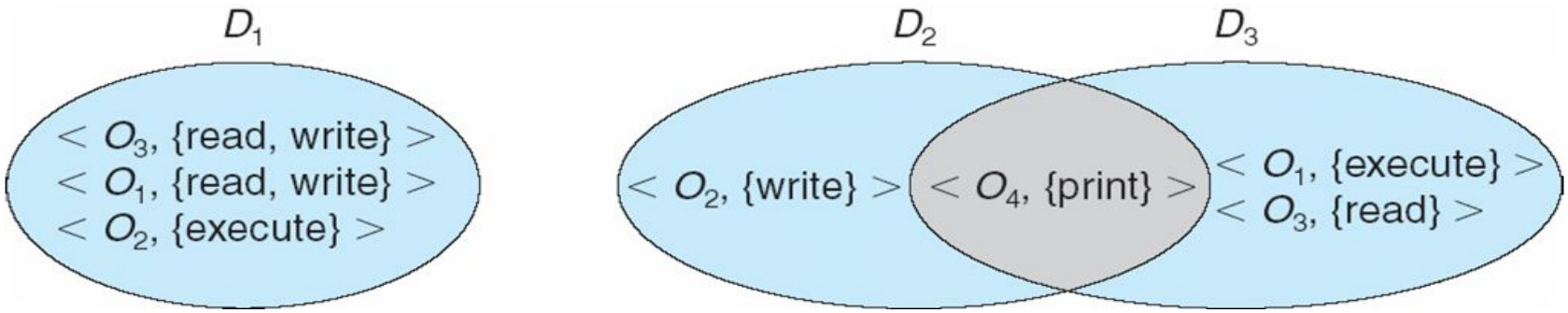
- Must consider **"grain"** aspect
  - **Rough**-grained privilege management is easier, simpler, but least privilege is better
  - **Fine**-grained management is more complex, more overhead, but more protective

- Domain can be user, process, procedure

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



Access Matrix

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

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	

Use of Access Matrix

- If a process in Domain D_i tries to do "op" on object O_j, then "op" must be in the access matrix
- User who creates object can define access column for that object
- Can be expanded to dynamic protection
  - Operations to add, delete access rights
  - Special access rights:
    - *owner of* O_i
    - *copy op from* O_i *to* O_j *(denoted by "*")*
    - *control –* D_i *can modify* D_j *access rights*
    - *transfer – switch from domain* D_i *to* D_j
  - *Copy* and *Owner* applicable to an object
  - *Control* applicable to domain object

Use of Access Matrix (Cont.)

- **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 the rules are strictly enforced
  - Policy
    - User dictates policy
    - Who can access what object and in what mode
- But doesn't solve the general confinement problem

Access Matrix with Domains as Objects

object / domain	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			

Access Matrix with *Copy* Rights

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	

(b)

Access Matrix With *Owner* Rights

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

(a)

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

(b)

Modified Access Matrix

object / domain	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 control
D_3		read	execute					
D_4	write		write		switch			

Implementation of Access Matrix

- Generally, a sparse matrix
- Option 1 – **Global table**
  - Store ordered triples < *domain, object, rights-set* > in table
  - A requested operation M on object O_j within domain D_i -> search table for < D_i, O_j, R_k >
    - with M ∈ R_k
  - But table could be large -> won't fit in main memory
  - Difficult to group objects (consider an object that all domains can read)

- Option 2 – **Access lists for objects**
  - Each column implemented as an access list for one object
  - Resulting per-object list consists of ordered pairs < *domain, rights-set* > defining all domains with non-empty set of access rights for the object
  - Easily extended to contain default set -> If M ∈ default set, also allow access

Implementation of Access Matrix (Cont.)

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

  Domain 1 = Read, Write
  Domain 2 = Read
  Domain 3 = Read

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

  Object F1 – Read
  Object F4 – Read, Write, Execute
  Object F5 – Read, Write, Delete, Copy

Implementation of Access Matrix (Cont.)

Option 3 – **Capability list for domains**

- Instead of object-based, list is domain based
- **Capability list** for domain is list of objects together with operations allows on them
- Object represented by its name or address, called a **capability**
- Execute operation M on object O_j, process requests operation and specifies capability as parameter
  - Possession of capability means access is allowed
- Capability list associated with domain but never directly accessible by domain
  - Rather, protected object, maintained by OS and accessed indirectly
  - Like a "secure pointer"
  - Idea can be extended up to applications

Option 4 – **Lock-key**

- Compromise between access lists and capability lists
- Each object has list of unique bit patterns, called **locks**
- Each domain as list of unique bit patterns called **keys**
- Process in a domain can only access object if domain has key that matches one of the locks

Video Links

https://www.youtube.com/watch?v=gr29JiWlTH8

https://www.youtube.com/watch?v=2YIhzk7tJI8

References

- https://www.unf.edu/public/cop4610/ree/Notes/PPT/PPT8E/CH15-OS8e.pdf
- https://www.tutorialspoint.com/operating_system/os_security.htm
- https://www.coursehero.com/file/19323929/Operating-System-Threats-and-Vulnerabilities/
- https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/15_Security.html
- https://devqa.io/security-threats-attack-vectors/
- https://www.geeksforgeeks.org/system-security/
- https://www.javatpoint.com/os-security-management