Guideline for Protection module: To know Access matrix you have to get an idea of principles of protection, domain of protection, access matrix with object and domains etc.

- **What are the strengths and weaknesses of implementing an access matrix using access lists that are associated with objects?**
- **Ans:** The strength of storing an access list with each object is the control that comes from storing the access privileges along with each object, thereby allowing the object to revoke or expand the access privileges in a localized manner. The weakness with associating access lists is the overhead of checking whether the requesting domain appears on the access list. This check would be expensive and needs to be performed every time the object is accessed.

Objectives

- Discuss the **goals and principles** of **protection** in a modern computer system
- Explain how protection domains combined with an **access matrix** are used to specify the resources a process may access
- Examine **capability-based** protection systems

Goals of Protection

- In one common protection model, a 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**
  - Static
  - Dynamic - **domain switching**, **privilege escalation**
  - "Need to know" a similar concept regarding access to data
  - "Containment of failure"

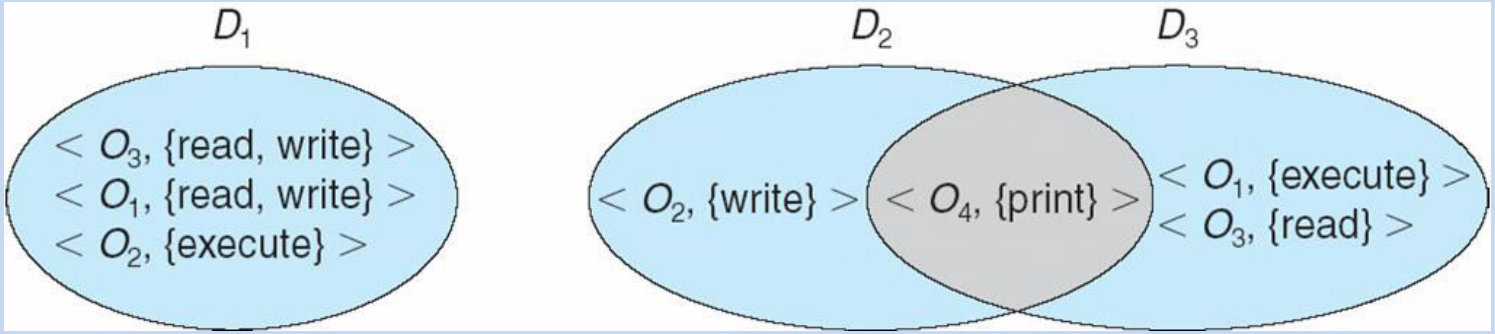
- Must consider "grain" aspect
  - Rough-grained
  - Fine-grained

- 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



Domain Implementation (UNIX)

- Domain = user-id
- Domain switch accomplished via file system
  - Each file has associated with it a domain bit (setuid bit)
  - When file is executed and setuid = on, then user-id is set to owner of the file being executed (similary "setgid")
  - When execution completes user-id is reset
- Domain switch accomplished via passwords
  - `su` command temporarily switches to another user's domain when other domain's password provided
- Domain switching via commands
  - `sudo` command prefix executes specified command in another domain (if original domain has privilege or password given)

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

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	

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* D_i *to* D_j *(denoted by "*")*
    - *control* – D_i *can modify* D_j *access rights*
    - *transfer (switch)* – *switch from domain* D_i *to* D_j
  - *Copy* and *Owner* applicable to an object
  - *Control* applicable to domain

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 that rules are strictly enforced
  - Policy
    - User dictates policy
    - Who can access what object and in what mode
    - Good policy supported by good **default values**

Access Matrix Example

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 Example

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 Example

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)

Access Matrix Example

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			