BILKENT UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
CS342 OPERATING SYSTEMS

LECTURE 14
PROTECTION
(CHAPTER 14)

Dr. İbrahim Körpeoğlu
http://www.cs.bilkent.edu.tr/~korpe

References

- *The slides here are adapted/modified from the textbook and its slides: Operating System Concepts, Silberschatz et al., 7th & 8th editions, Wiley.*

REFERENCES

- Operating System Concepts, 7th and 8th editions, Silberschatz et al. Wiley.
- Modern Operating Systems, Andrew S. Tanenbaum, 3rd edition, 2009.

Outline

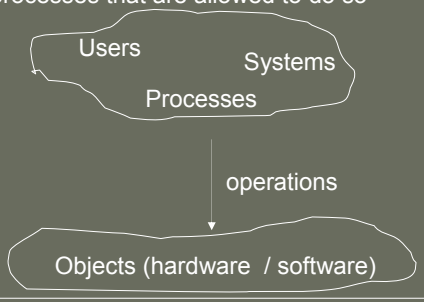
- Goals of Protection
- Principles of Protection
- Domain of Protection
- Access Matrix
- Implementation of Access Matrix
- Access Control
- Revocation of Access Rights
- Capability-Based Systems
- Language-Based Protection

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 and language-based protection systems

Goals of Protection

- Operating system 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



Access rights?

Principles of Protection

- Guiding principle – principle of least privilege
  - Programs, users and systems should be given *just enough privileges* to perform their tasks

Domain of Protection

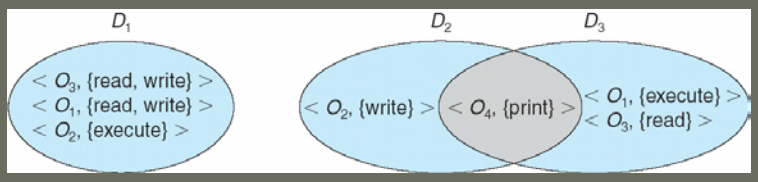
- Protection requirement:
  - 1. a process should be allowed to access only those resources for which it has authorization
  - 2. a process should be able to access only those resources that it currently requires to complete its task (*need to know* principle)
- To do these, a process operates within a protection domain.
- A protection domain specifies which resources/objects can be accessed in which way by the processes operating in that domain.

Domain of Protection

- We need
  - Associating processes with domains
    - Static association
    - Dynamic association
  - Defining domains and changing their content.
    - No change is allowed (static domain)
    - Can be changed (dynamic domain)

Domain Structure

- Access-right = <*object-name*, *a_set_of_rights*>
  where *a_set_of_rights* is a subset of all valid operations that can be performed on the object.
- Domain = set of access-rights



Domain Implementation (Unix)

- System consists of 2 domains:
  - User
  - Supervisor
- UNIX
  - Domain = user-id (i.e. domain determined from user-id)
  - The domain of a process is defined by its UID and GID.
  - Files (objects) also have associated UID and GUI.
  - Then: we can make a list of files (includes file corresponding to devices) that can be accesses (an how) by the process.

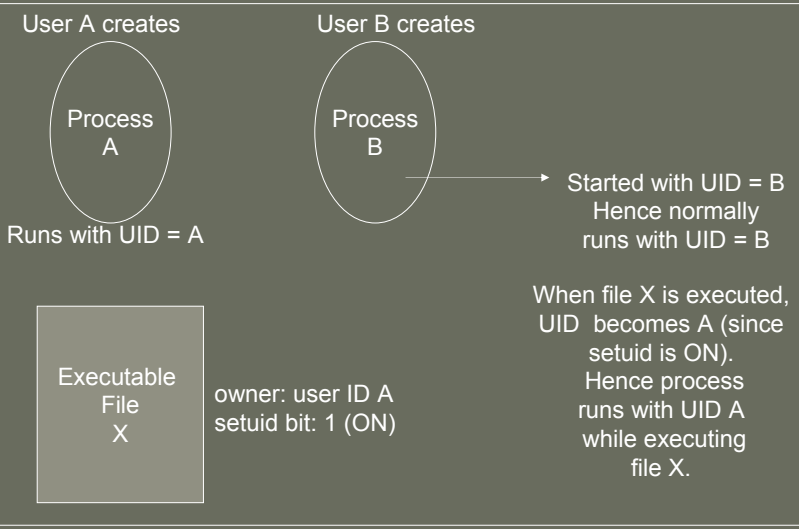
Domain Implementation (Unix)

- A user loges in. All processes created by that user operate in the same dome.
  - What should happen when a process tries to execute a file which is created by another user?
    - File has user ID as well.
    - Which user ID should be used by the running process?
      - » The created of the process?
      - » The owner of the executable file?

- 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 (domain) is set to owner of the file being executed. When execution completes user-id is reset

owner: a user ID
setuid bit: 0 or 1

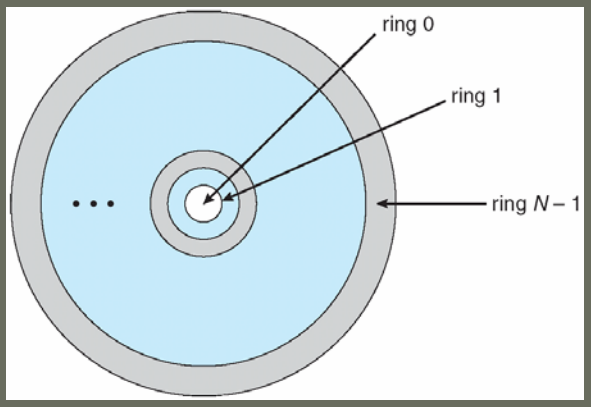
Domain Implementation (Unix)



Domain Implementation (MULTICS)

- Let D_i and D_j be any two domain rings
- If $j < i \Rightarrow D_i \subset D_j$ (Dj has more privileges than Di)

Consider domains as rings; Hierarchical organization



Access Matrix

- Modeling protection
- View protection as a matrix (*access matrix*): protection rules are expressed using a matrix
  - Rows represent domains
  - Columns represent objects
  - *Access(i, j)* contains the set of operations that a process executing in $Domain_i$ can invoke on $Object_j$
- A general method and mechanism
- Can be implemented in various ways.

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
- Can be expanded to dynamic protection
  - Operations to add, delete access rights to/from the Matrix
  - Special access rights for a domain:
    - *owner of* O_i
    - *copy op from* O_i *to* O_j
    - *control –* D_i *can modify* D_j *access rights*
    - *transfer – switch from domain* D_i *to* D_j

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

Access Matrix With Domains as Objects

domain \ object	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			

Switch is applicable to only domain objects. A process can switch to that domain.

Access Matrix with *Copy* Rights

Has copy right for object F2

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

(a)

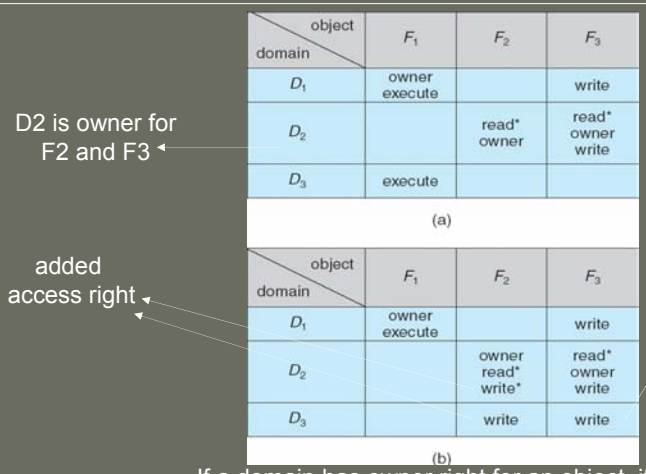
domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

copied access right

If a domain has a copy right, then it can copy the right to another domain.

Access Matrix With *Owner* Rights



If a domain has owner right for an object, it can add / remove access right entries for the object

Modified Access Matrix

domain \ object	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			

Control is applicable to only domain objects. Di can control Dj (any access right can be removed/added from/to Dj by Di)

Implementation of Access Matrix

- Each Column = Access-Control List for one object
  Defines who can perform what operation.

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

Access Control List associated with an object

Implementation of Access Matrix

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

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

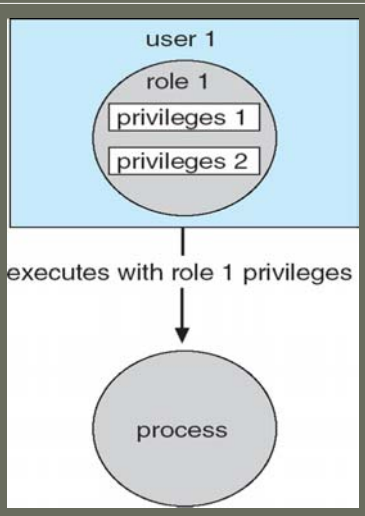
Access Rights of a Domain (i.e. capabilities of a domain: what a domain can do)

Access Control

- Protection can be applied to non-file resources
- Solaris 10 provides role-based access control (RBAC) to implement least privilege
  - Privilege is right to execute system call or use an option within a system call
  - Can be assigned to processes
  - Users assigned roles granting access to privileges and programs

Role Based Access Control

Users can take roles if they know the corresponding passwords



Revocation of Access Rights

- Assume want to remove (revoke) an access right from an object
- How we will revoke and whether it is easy or difficult to revoke depends on the implementation of Access Matrix.

- Access List – Delete access rights from access list
  - Simple
  - Immediate

- Capability List – Scheme required to locate capability in the system before capability can be revoked
  - Reacquisition
  - Back-pointers
  - Indirection
  - Keys

Capability Based Systems

- Hydra
  - Fixed set of access rights known to and interpreted by the system
  - Interpretation of user-defined rights performed solely by user's program; system provides access protection for use of these rights

- Cambridge CAP System
  - Data capability - provides standard read, write, execute of individual storage segments associated with object
  - Software capability -interpretation left to the subsystem, through its protected procedures

Language Based Protection

- Specification of protection in a programming language 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
- Interpret protection specifications to generate calls on whatever protection system is provided by the hardware and the operating system

Protection in Java 2

- Protection is handled by the Java Virtual Machine (JVM)
- A class is assigned a protection domain when it is loaded by the JVM
- The protection domain indicates what operations the class can (and cannot) perform
- If a library method is invoked that performs a privileged operation, the stack is inspected to ensure the operation can be performed by the library

Stack Inspection

protection domain:	untrusted applet	URL loader	networking
socket permission:	none	*.lucent.com:80, connect	any

class:	gui:
	...
	get(url);
	open(addr);
	...
		get(URL u):
		...
		doPrivileged {
		open('proxy.lucent.com:80');
		}
		<request u from proxy>
		...
			open(Addr a):
			...
			checkPermission (a, connect);
			connect (a);
			...

End of Lecture