

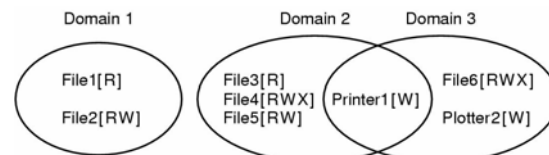
Protection

CS 256/456

Dept. of Computer Science, University of Rochester

OS Protection Mechanism

- Operating system consists of a collection of objects, hardware or software (e.g., files, printers)
- Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so.
- Access-right = $\langle \text{object-name}, \text{rights-set} \rangle$
- Domain = group of processes having the same set of access-rights



Protection Matrix

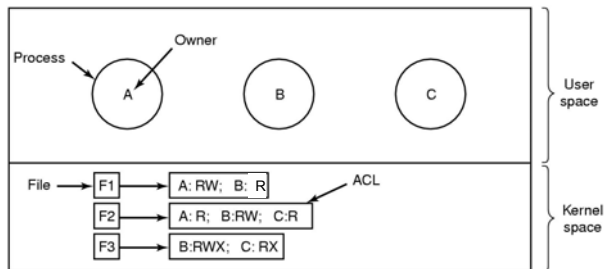
- View protection as a matrix (*protection matrix*)
 - Rows represent domains
 - Columns represent objects
 - $Access(i, j)$ is the set of operations that a process executing in $Domain_i$ can invoke on $Object_j$
- Dynamic protection
 - Operations to add, delete access rights

		Object							
		File1	File2	File3	File4	File5	File6	Printer1	Plotter2
Domain	1	Read	Read Write						
	2			Read	Read Write Execute	Read Write		Write	
	3						Read Write Execute	Write	Write

Implementation of Protection Matrix

- The complete protection matrix consumes too much space and frankly it is very sparse.
 - two ways to condense it
- 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 for each domain
Define what operations allowed on what objects.
 - Object 1 - Read
 - Object 4 - Read, Write, Execute
 - Object 5 - Read, Write, Delete, Copy

Access Control Lists

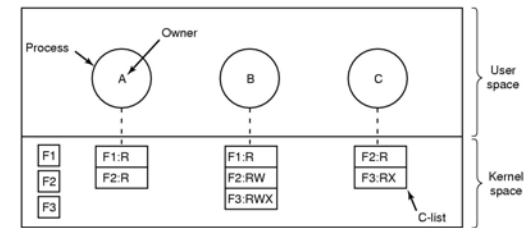


Use of access control lists to manage file access

Revocation of access rights?

Capabilities

- Tagged architecture
 - memory words containing capabilities are tagged
 - user programs can only read those words; only kernel programs can change those words
- Kernel-space capability list
 - user programs use handles (e.g., file descriptors) to refer to them



Passing Capabilities at User-level

- Cryptographically-protected capability
 - a random number (called "check") is generated for each file at creation time and maintained in secrecy
 - capability is formed cryptographically
- | | | | |
|--------|--------|--------|--------------------------|
| Server | Object | Rights | f(Object, Rights, Check) |
|--------|--------|--------|--------------------------|
- Function f() is a one-way function that:
 - it is computationally infeasible to guess what "check" is even when object, rights, and f(object, rights, check) is known.
 - Cryptographically-protected capability can be passed around but not tampered

Revocation of Access Rights for Capabilities

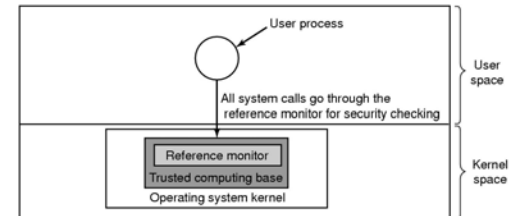
- Revocation:
 - *total*: revoking rights to all processes
 - *selective*: revoking rights to a selected group of processes (e.g., all processes belonging to a particular user)
- *Capability List* - Scheme required to locate capability in the system before capability can be revoked.
 - Indirection
 - Timeout & Reacquisition
 - Back-pointers

ACL vs. Capabilities

- Compare them on:
 - determining access rights for a process
 - determining access rights to a object
 - efficiency of access rights checking

Trusted Systems

- Trusted Computing Base
 - hardware and software for enforcing security rules
- If the TCB works according to specification, the system security cannot be compromised
- A reference monitor-based TCB:



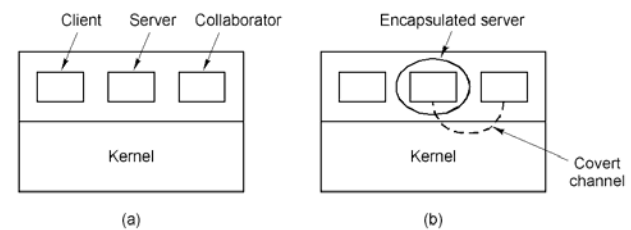
Formal Models of Secure Systems

	Objects				Objects		
	Compiler	Mailbox 7	Secret		Compiler	Mailbox 7	Secret
Eric	Read Execute			Eric	Read Execute		
Henry	Read Execute	Read Write		Henry	Read Execute	Read Write	
Robert	Read Execute		Read Write	Robert	Read Execute	Read	Read Write

(a) (b)

- Protection commands: operations that can change protection matrix
- TCB must ensure that protection commands do not move the protection matrix from an authorized state to an unauthorized state

Covert Channels



(a) Client, server and collaborator processes

(b) Encapsulated server can still leak to collaborator via covert channels

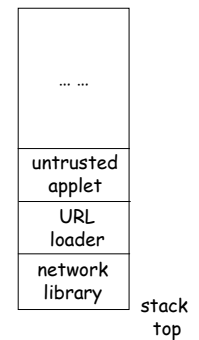
- Covert channels
 - Modulating CPU usage
 - Locking/unlocking files

Protection in UNIX

- Protection domains: users
- Access matrix for files:
 - a simplified access control list
- Protection commands for files:
 - each user can change protection on files it owns
 - superuser can do everything

Protection in Java

- A Java class can be loaded remotely; therefore can be dangerous
 - 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 for access right violation.



Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).