Security II

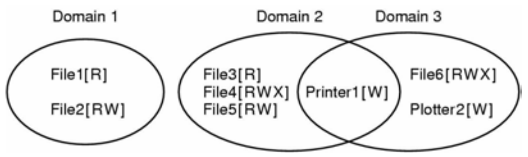
Security Policy & Mechanisms

- Policy decides what kinds of entities can perform operations on what kinds of objects
  - Deals with users, processes, students, files, printers, managers
    - Example: Students can't use the colour printer
- *Protection mechanisms* are used to represent and enforce security policy
  - Example: reference monitor looks up a table representing a policy and decided yes/no.

Protection Mechanisms

- Protection system deals with
  - *Objects*
    - Set of 'things' in the system that can be operated on
      - Files, devices, sockets, etc…
  - *Rights*
    - The permission to perform one of the operations possible on an object
      - Example: Possessing permission to read an object is termed possessing a *read right* to the object.
  - *Domains*
    - A set of (object, right) pairs which together represent the set of possible operations on objects.
    - Each process has a domain associated with it.

Protection Domains

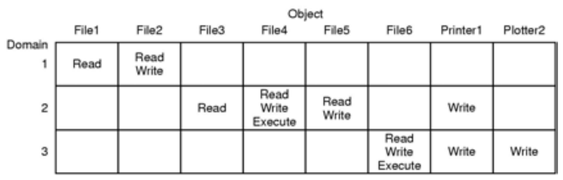


Examples of three protection domains

Protection Domain Example

- UNIX
  - The UID and GID of a process determines the *domain* the process executes within
    - Determines exactly what rights the process has to objects (files) in the system
  - Another process with the same UID, GID lies with the same domain
    - Has exactly the same set of access rights to objects
  - Process can change domains to gain access rights via SETUID or SETGUID

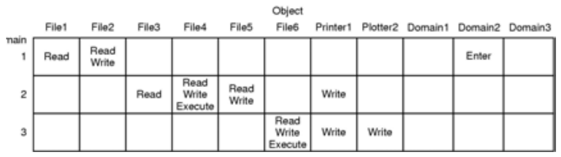
Representing Protection Domains



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

Represent access rights using a protection matrix

Protection Domains

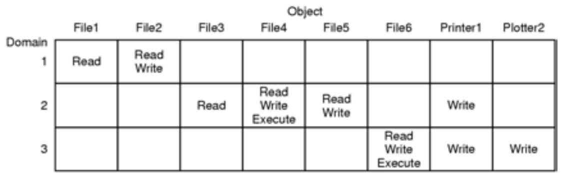


main	File1	File2	File3	File4	File5	Object File6	Printer1	Plotter2	Domain1	Domain2	Domain3
1	Read	Read Write								Enter	
2			Read	Read Write Execute	Read Write		Write				
3						Read Write Execute	Write	Write			

A protection matrix with domains as objects

Access Matrix Issue

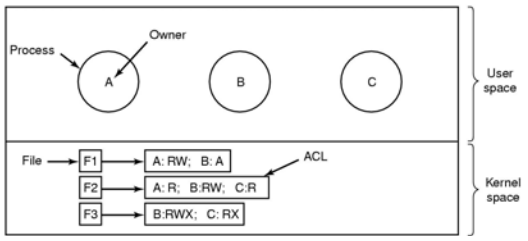
- Most domains have access to a subset of all objects in the system
  - ⇒Matrix is sparsely populated
  - ⇒Wastes space
- Idea
  - Store populated entries by column (object)
    - List of domains and operation that can operate on the object
  - Store populated entries by rows (domain)
    - List of objects and operations domain can perform
    - Note: Domains are sometimes termed *subject*, or *principal*.



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

- Columns: *Access Control Lists*
- Rows: *Capabilities*

Access Control Lists

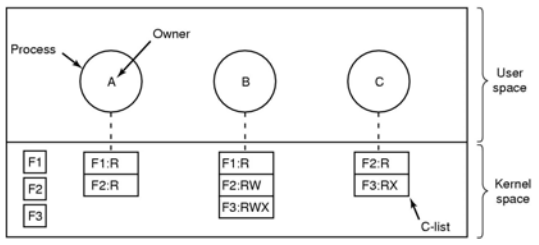


Use of access control lists of manage file access

Access Control Lists

- List stored with meta-data of object
  - Example: stored in the inode of the file
- Easy to revoke access to the object
- Easy to determine who has (direct) rights to the file
  - *'direct'* meaning ignoring transitive rights changes
    - Example: A writes B, B writes C ⇒ A writes C

Capabilities



Each process has a capability list

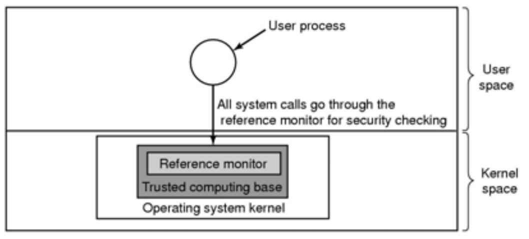
Capabilities

- Capability list stored with the subject (e.g. the process)
- Set of capabilities forms the protection of domain of the subject
  – Easy to determine the protection domain of the process
- Hard to determine who has (direct) access to a particular object
  – Capabilities can be stored many places (with each process, each user, etc..)
  – Have to examine them all for one referring to the object
- Revocation is hard
  – Have to remove all capabilities to an object

Building Secure Systems

- Sometimes called *Trusted Systems*
- Consist on users/processes running on *Trusted Computing Base* (TCB)
- Idea
  – TCB has a small, understandable, verifiable, security model
  – Enables statements/reasoning about security properties
    - "Bob can never read file X"
    - "Alice can only run the word processor"
    - "The program can only modify file Z"
  – All operations are authorised via the TCB.

Trusted Systems
Trusted Computing Base



A reference monitor

Formal Models of Secure Systems

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

(a)

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

(b)

(a) An authorized state

(b) An unauthorized state (Robert can read Henry's mailbox)

Given a set of authorized and unauthorized states, and the TCB's security model, can we prove that starting at (a), (b) can never happen??

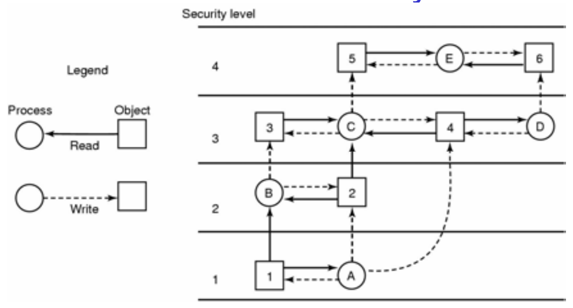
Access Control Policy

- *Discretionary Access Control*
  – Allow users to determine who can read and write their files
  – Policy not enough to control information flow
  – Example: UNIX
- *Mandatory Access Control*
  – System determines (and enforced) who can read and write individual files
  – Example policies: Bell-La Padula and Biba

Bell-La Padula Multilevel Security

- Designed to keep secrets
  – Simple security property
    - A process at level *k* read objects at it's level or lower
      – Lieutenant can read sergeants files, but not vice versa
      – Can *read down*
  – The * property
    - A process can write files to it's level or above
      – Sergeants can write information to Lieutenants, who can write to Generals.
      – Can *write up*
- Issue
  – Generals can't write to Lieutenants, etc.
    - Can't *write down*
  – Generals can't give orders!!!
  – Privates can write to generals potentially false information

Multilevel Security



The Bell-La Padula multilevel security model

Multilevel Security
The Biba Model

- Principles to guarantee integrity of data

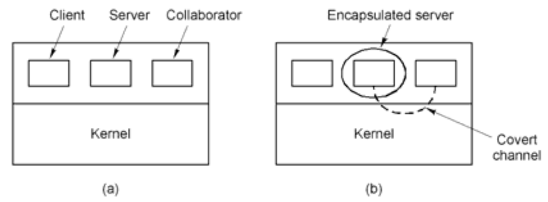
1. Simple integrity principle
   - process can write only objects at its security level or lower
2. The integrity * property
   - process can read only objects at its security level or higher

Multilevel Security
The Biba Model

- Managers can write the files of employees
- Employees cannot write the files of managers
- Employees read (trust) files of managers
- Managers cannot read (trust) the files of employees

- Note: Biba and Bell-La Padula are in direct conflict with each other
  - Developing sensible security policy is hard

Covert Channels



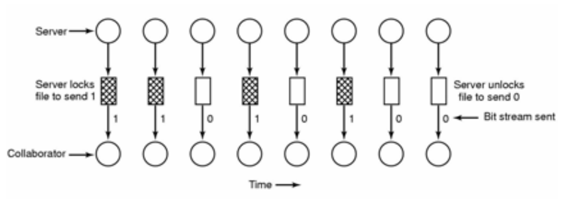
Client, server and collaborator processes

We'd like to confine the server so as to not pass on client's info

Encapsulated server can still leak to collaborator via covert channels

Example: CPU modulation

Covert Channels



A covert channel using file locking
(Assuming a common read-only file)

Covert Channels

- Can be created using a any shared resource whose behaviour can be monitored
  - Network Bandwidth
  - CPU time
  - Disk Response time
  - Disk Bandwidth