IF2230 - 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

Security & Protection

- Security: jaminan bahwa integritas dan keamanan sistem tidak terlanggar
- Protection: mekanisme untuk mengendalikan akses ke objek
- Protection mechanism enforces security policies

Goals of Protection

- Operating system terdiri atas kumpulan objek, hardware atau software
- Setiap objek memiliki nama unik dan dapat diakses melalui operasi yang terdefinisi
- Protection problem – memastikan setiap objek diakses dengan benar dan hanya oleh mereka yang dibolehkan mengakses

Principles of Protection

- Guiding principle – principle of least privilege
  - Programs, users and systems should be given just enough privileges to perform their tasks
  - tujuan: agar jika terjadi pelanggaran, kerusakan yang terjadi dapat dibatasi
  - contoh:
    - tidak bisa meng-kill proses user lain
    - tidak dapat membuka file /etc/hosts untuk ditulis
  - pelanggaran
    - print daemon dapat membuat seseorang menambahkan user
    - proses dapat menulis ke sebuah file yang tidak diperlukan
    - user account biasa mendapat privilege admin
  - masalah: least privilege sulit untuk didefinisikan

Principles of Protection

- Privilege separation
- program dipecah menjadi beberapa bagian: ada yang memiliki privilege tinggi, ada yang rendah
- contoh: pada UNIX, setiap proses memiliki real dan effective user ID
- privileges dievaluasi berdasarkan effective user id
  - umumnya uid == euid
- executable file dapat ditandai dengan setuid bit
  - saat dijalankan, euid adalah owner id

Security Goals

- Authentication
  - menjamin users, programs, mesin dan resources teridentifikasi dengan benar
- Confidentiality
  - menghindari akses yang tidak authorized
- Integrity
  - menjamin integritas data (dihapus, diubah, ditambahkan)
- Availability
  - menjamin sistem dapat digunakan

Operating Systems

- OS menyediakan untuk proses akses ke resources:
  - Processor
  - Memory
  - Peripheral devices
  - logical persistent data
  - communication networks

- akses ke resources selalu melalui OS
- OS menentukan apakah akses dibolehkan
  - aturan yang mengatur akses: policy

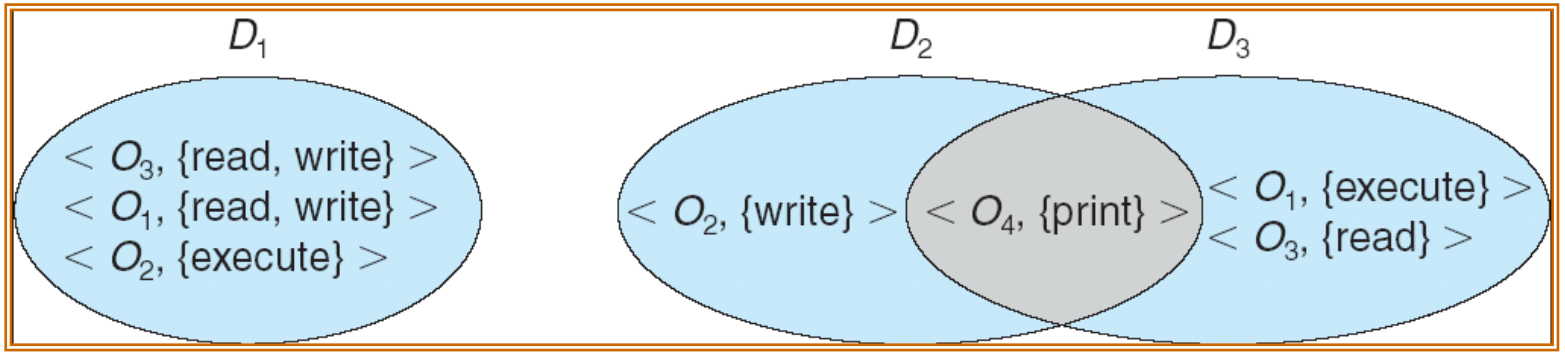
Domains of Protection

- Proses berinteraksi dengan objek
- objek mencakup:
  - hardware (CPU, memory, I/O devices)
  - software (files, processes, semaphore, messages, signals)
- proses hanya boleh mengakses objek yang dia memiliki otorisasi terhadap objek tersebut
  - proses berjalan pada protection domain
  - protection domain mendefinisikan objek mana saja yang dapat diakses oleh proses

Domain Structure

- Access-right = <*object-name*, *rights-set*>
  dimana *rights-set* adalah subset dari semua valid operations yang dapat dilakukan terhadap object.

- Domain = set of access-rights



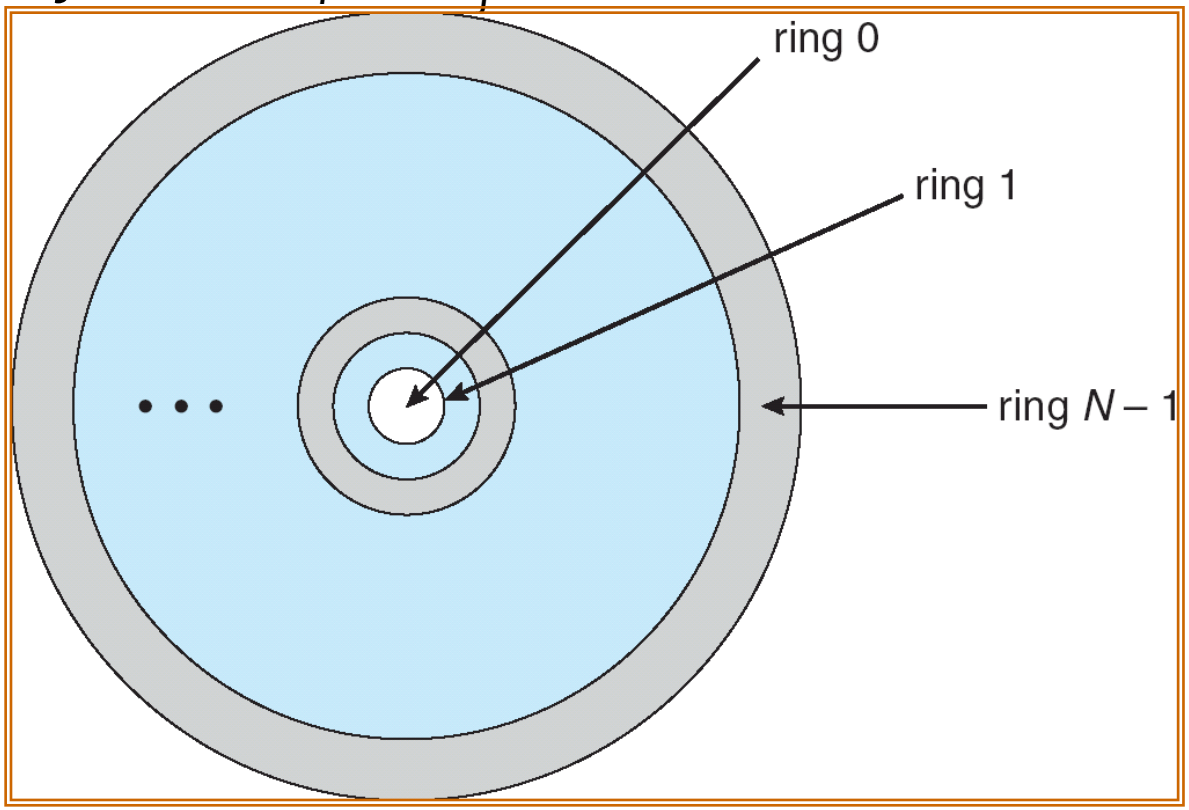
Domain Implementation (UNIX)

- System consists of 2 domains:
  - User
  - Supervisor

- UNIX
  - Domain = user-id
  - Domain switch dilakukan melalui file system.
    - Setiap file memiliki atribut domain bit (setuid bit).
    - Saat file di-eksekusi dan setuid = on, maka user-id di set ke owner of the file yang dieksekusi. Saat eksekusi selesai, user-id di-reset.

Domain Implementation (Multics)

- Let D_i and D_j be any two domain rings.
- If $j < i \Rightarrow D_i \subseteq D_j$



Multics Rings

Pemodelan Protection Domain dengan 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	

Figure A

Use of Access Matrix

- Jika sebuah proses di Domain D_i mencoba melakukan operasi "op" pada object O_j, maka "op" harus terdapat pada access matrix.

- Dapat dikembangkan untuk dynamic protection.
  - Operasi untuk menambah dan menghapus access rights.
  - Special access rights:
    - *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 (Cont.)

- desain Access matrix memisahkan mekanisme dari policy.
  - Mechanism
    - Operating system menyediakan access-matrix + rules.
    - OS memastikan bahwa matrix hanya dimanipulasi oleh authorized agents dan that rules are strictly enforced.
  - Policy
    - User dictates policy.
    - Who can access what object and in what mode.

Access Matrix of Figure A 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			

Figure B

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 of Figure B

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			

Implementasi Access Matrix

- Access Matrix jika disimpan sebagai tabel, tidak praktikal => umumnya sparse matrix

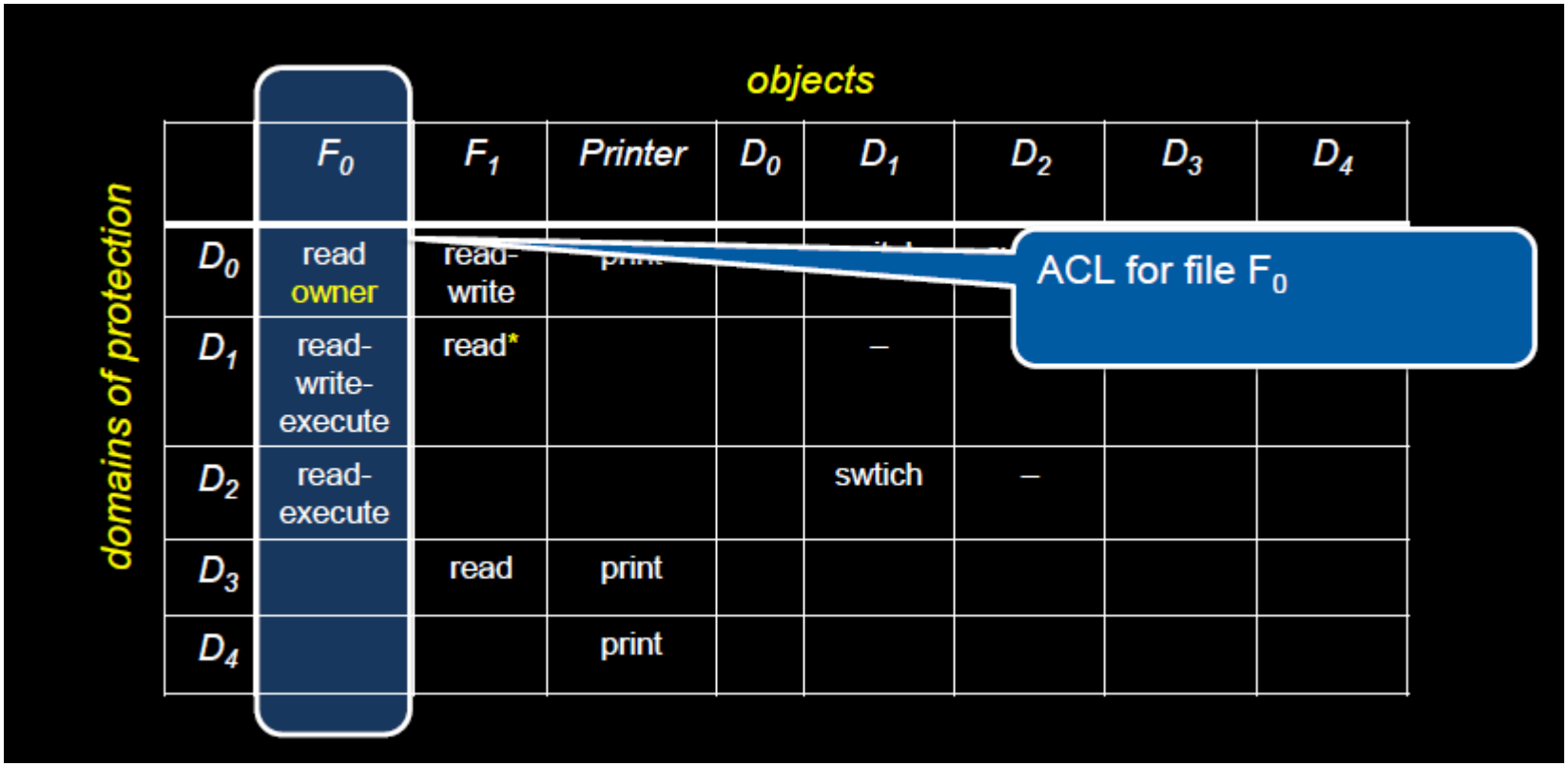
- Each column = Access-control list for one object
  Defines who can perform what operation.

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

  ⋮

- 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


objects
domains of protection
F_0
F_1
Printer
D_0
D_1
D_2
D_3
D_4
D_0
read
owner
read-
write
D_1
read-
write-
execute
read*
–
D_2
read-
execute
swtich
–
D_3
read
print
D_4
print
ACL for file F_0

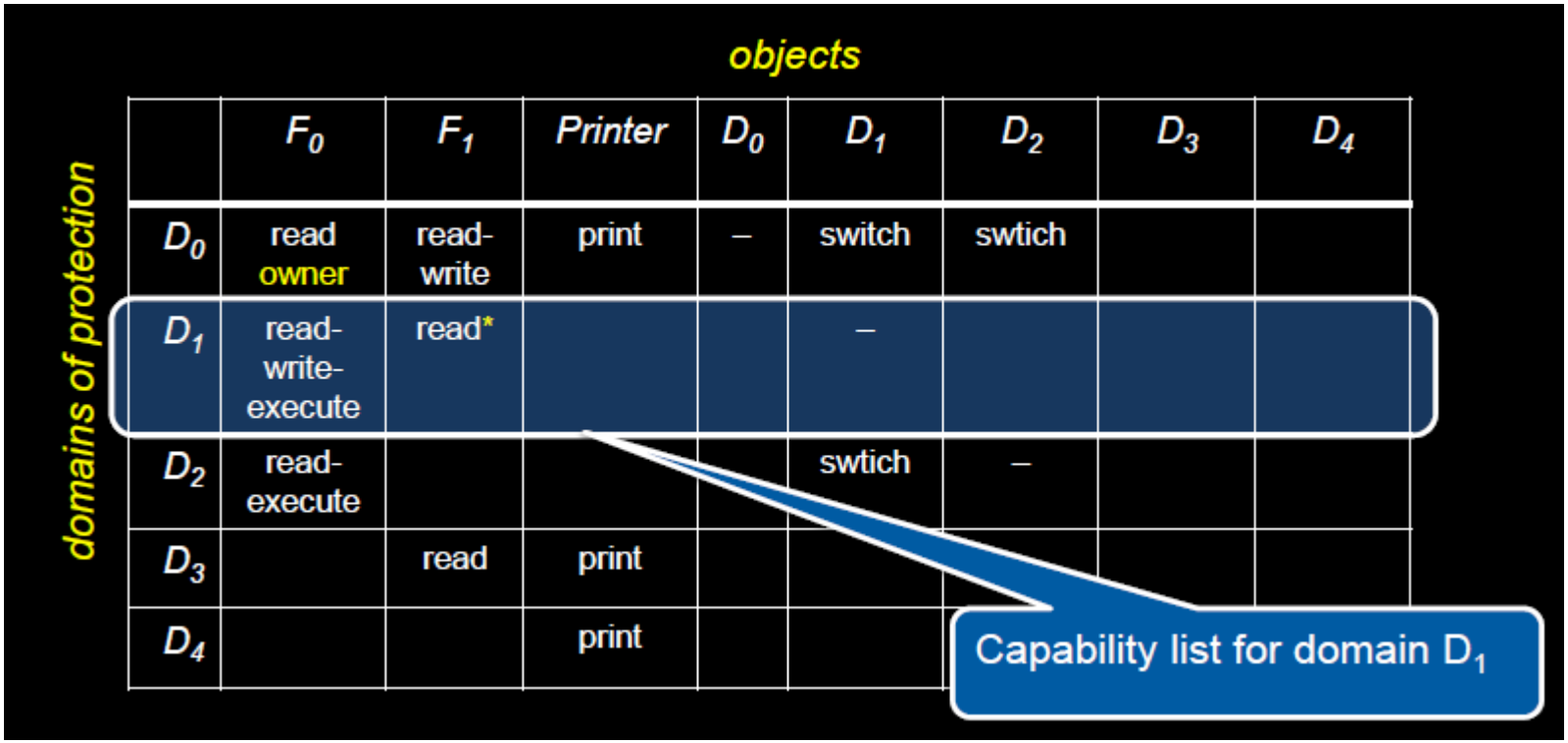
ACL pada Linux

- ACL memiliki ukuran tidak tetap (tergantung jumlah domain yang dapat mengakses objek tersebut)
  - tidak dapat disimpan pada i-node
- Kompromi:
  - file didefinisikan memiliki definisi access right pada 3 domain: owner, group dan other
  - permission: read, write, execute, directory search, set uid, set gid
  - default: umask system call
  - chown: mengubah owner
  - chgrp: mengubah group
  - chmod: mengubah permission

Full ACL pada POSIX systems

- bagaimana jika kita ingin mengimplementasikan full ACL
- extended attribute disimpan di luar inode
- list of permissions pada users dan groups
  - operasi untuk semua objek: delete, readattr, writeattr, readextattr, writeextattr, readsecurity, writesecurity, chown
  - operasi pada direktori: list, search, add_file, add_subdirectory, delete_child
  - operasi pada file: read, write, append, execute
  - inheritance controls

Capability List



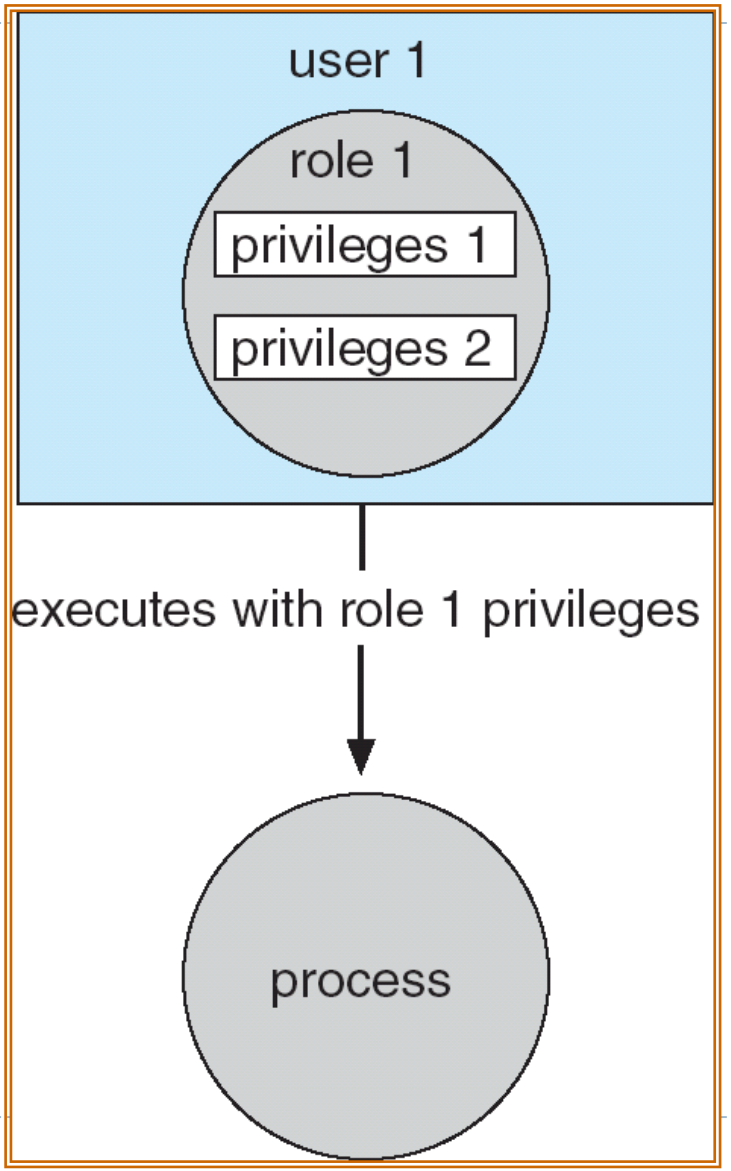
Capability List

- Berisi daftar objek dan operasi yang dapat dilakukan terhadap objek tersebut
- setiap item pada list disebut sebagai capability: operasi yang dapat dilakukan pada sebuat objek
- proses memberikan capability saat melakukan request untuk sebuah operasi
- proses tidak dapat mengubah capability list

Access Control

- Protection dapat diterapkan pada non-file resources
- Solaris 10 menyediakan **role-based access control** untuk mengimplementasikan least privilege
  - Privilege adalah right untuk menjalankan system call atau menggunakan sebuah opsi di dalam system call
  - Dapat diassign ke processes
  - Users assigned roles granting access to privileges and programs

Role-based Access Control in Solaris 10



Revocation of Access Rights

- *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