EDAF35: OPERATING SYSTEMS

MODULE 9.A PROTECTION

CONTENTS
PROTECTION

- Goal and Principles
- Models and Abstractions:
  - Domains of Protection
  - Access Matrix
- Implementations
- Language-Based Protection

CHAPTER 14
PROTECTION

GOAL AND PRINCIPLES

PROTECTION

- “control the access of programs/processes to resources” (= HW, SW objects)
  - to prevent violations
  - to improve reliability
  - to enforce policies

- separate “how?” (mechanism) from “what?” (policy)

- principle of least privilege:
  “give no more than enough rights to carry out operation”

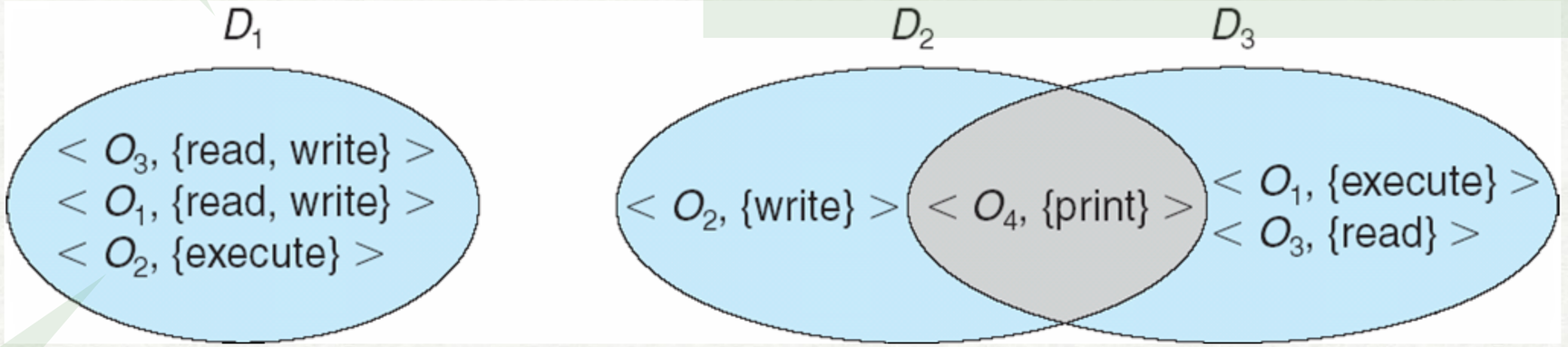
- (similar) need-to-know principle:
  “allow access only to the information needed for the operation”

DOMAINS OF PROTECTION

MODELS AND ABSTRACTIONS

DOMAIN
(SET OF ACCESS RIGHTS)

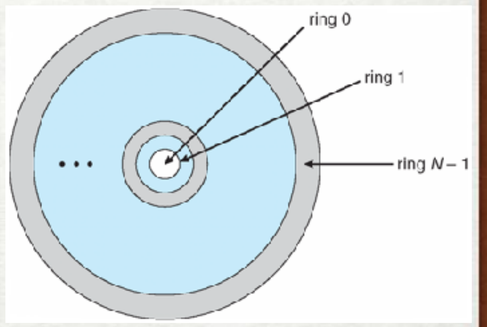
USERS/PROCESSES OPERATE IN DOMAINS
AND MAY SWITCH BETWEEN THEM



ACCESS RIGHT:
OBJECT, OPERATIONS

UNIX
DOMAIN = USER ID SWITCH DOMAINS = SETUID BIT ACCESS RIGHTS = RWX, GROUP USER OTHERS MAN SU, SUDO

MULTICS
DOMAINS = RING STRUCTURE (RING0 HAS MOST PRIVILEGE) SWITCH DOMAINS = CROSS RINGS NO SUPPORT FOR “NEED-TO-KNOW”



ACCESS MATRIX

MODELS AND ABSTRACTIONS

- mechanism; allows for different policies

domain \ object	F_1	F_2	F_3	laser printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

+ owner, copy rights

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

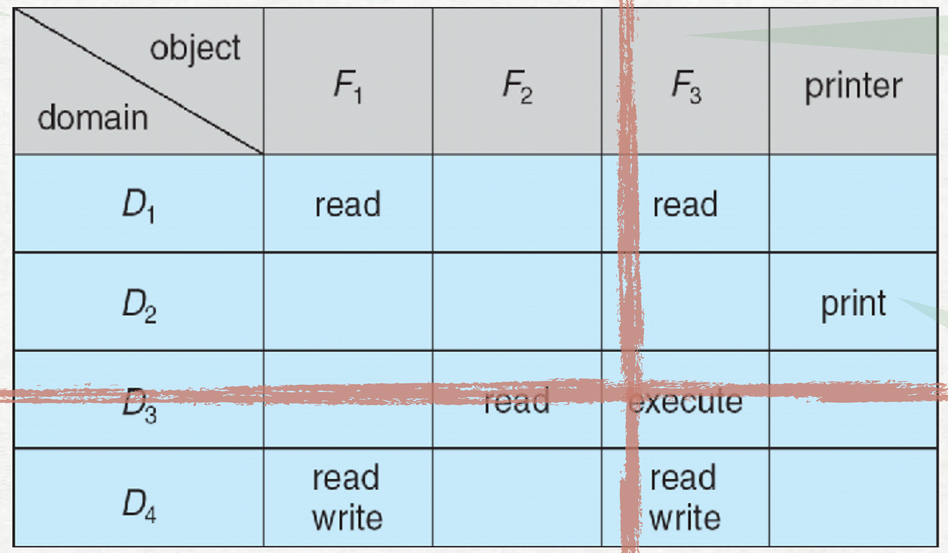
(a)

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

(b)

IMPLEMENTATIONS OF THE ACCESS MATRIX

1. FULL TABLE



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	

2. ACCESS LISTS

column-wise: which domain can access this object and how

3. CAPABILITY LISTS

row-wise: which object can be accessed and how by this domain

4. LOCK & KEY

bit patterns: match a "key" with a "lock" for certain operations on an object

WHICH TO CHOOSE? DEPENDS:
REVOCATION OF RIGHTS FOR AN OBJECT IS TRICKY IN 3, EASY IN 2

COMBINATIONS EXIST
(E.G. UNIX 2,3 - FILES, OPEN, DESCRIPTORS)

LANGUAGE-BASED PROTECTION

- application developers to implement own policies based on existing mechanisms = allows for finer access control, specific policies:
  - declare and distribute capabilities, access rights, and even order of operations
- partly already there: types, objects, references, ownership, mutability (in some)

JAVA

- stack inspection
  (how did we get here?)
- take responsibility via
  doPrivileged(), checkPermission()
- trusted/untrusted resources in same VM

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 MODULE 9.A