Computer Security DD2395

http://www.csc.kth.se/utbildning/kth/kurser/DD2395/dasak10/

Spring 2010

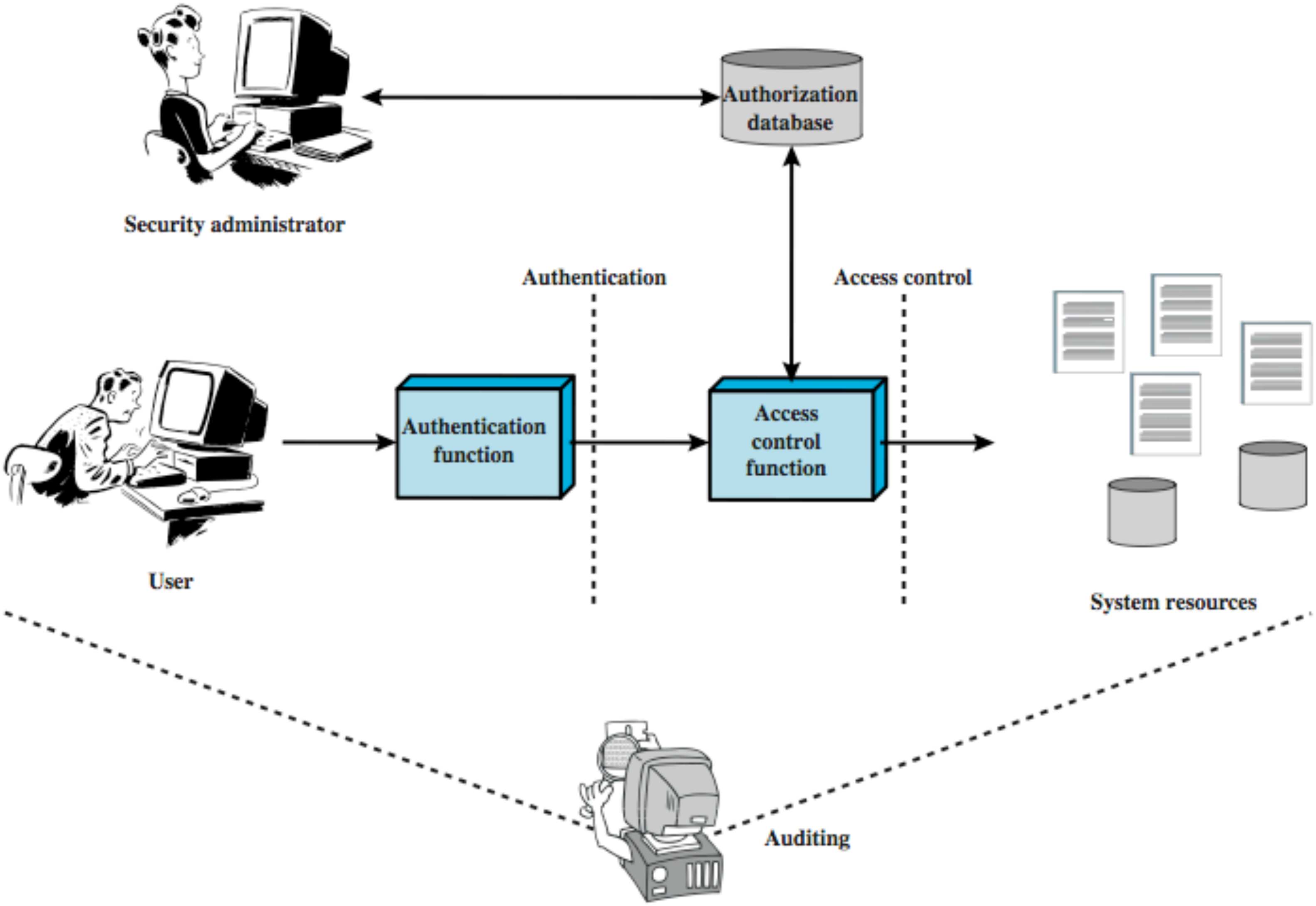
Sonja Buchegger
buc@kth.se

Lecture 4, Jan. 27, 2010
Access Control

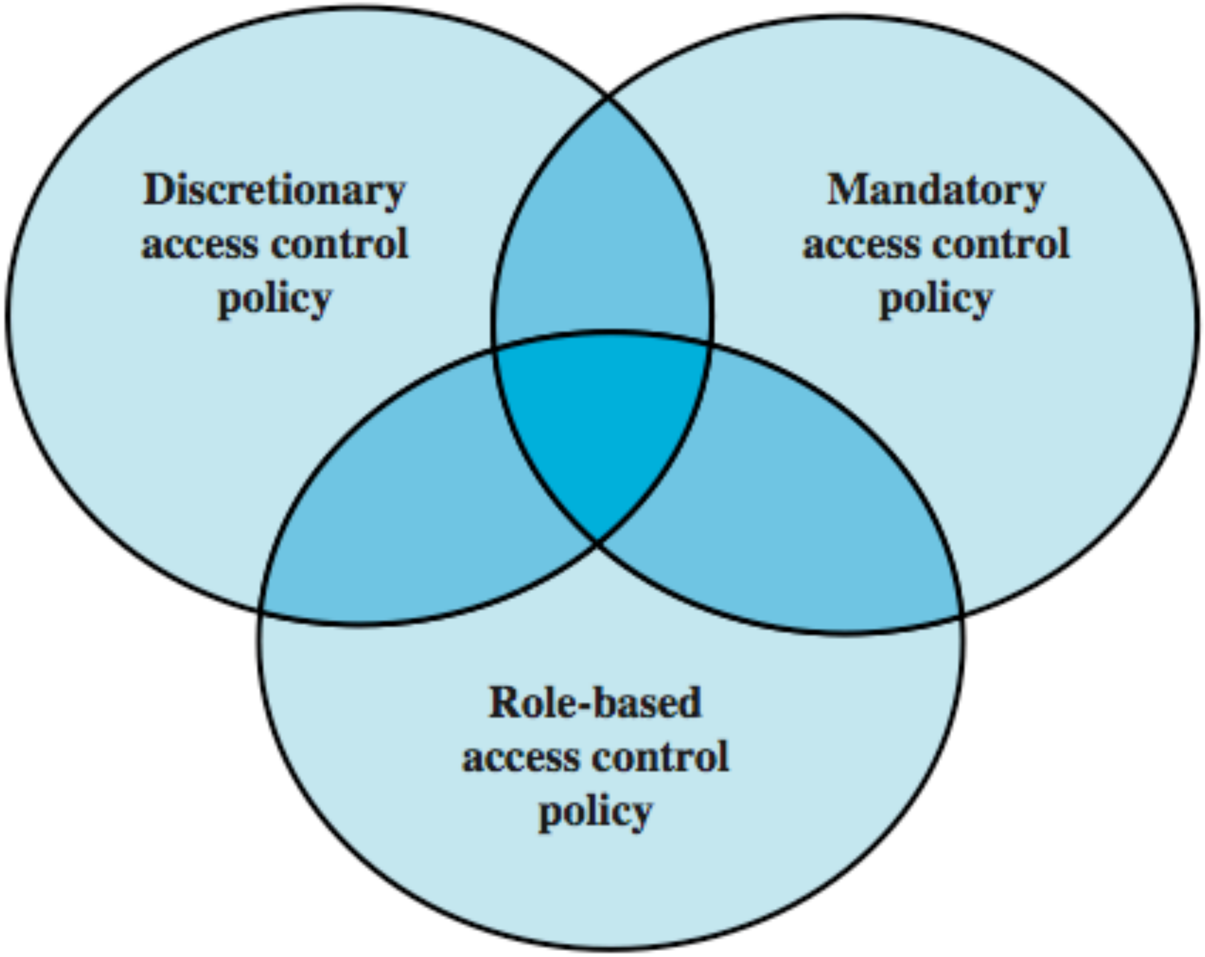
Access Control

- “The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner“
- central element of computer security
- assume have users and groups
  - authenticate to system
  - assigned access rights to certain resources on system

Access Control Principles



Access Control Policies


Discretionary access control policy
Mandatory access control policy
Role-based access control policy

Access Control Requirements

- reliable input
- fine and coarse specifications
- least privilege
- separation of duty
- open and closed policies
- policy combinations, conflict resolution
- administrative policies

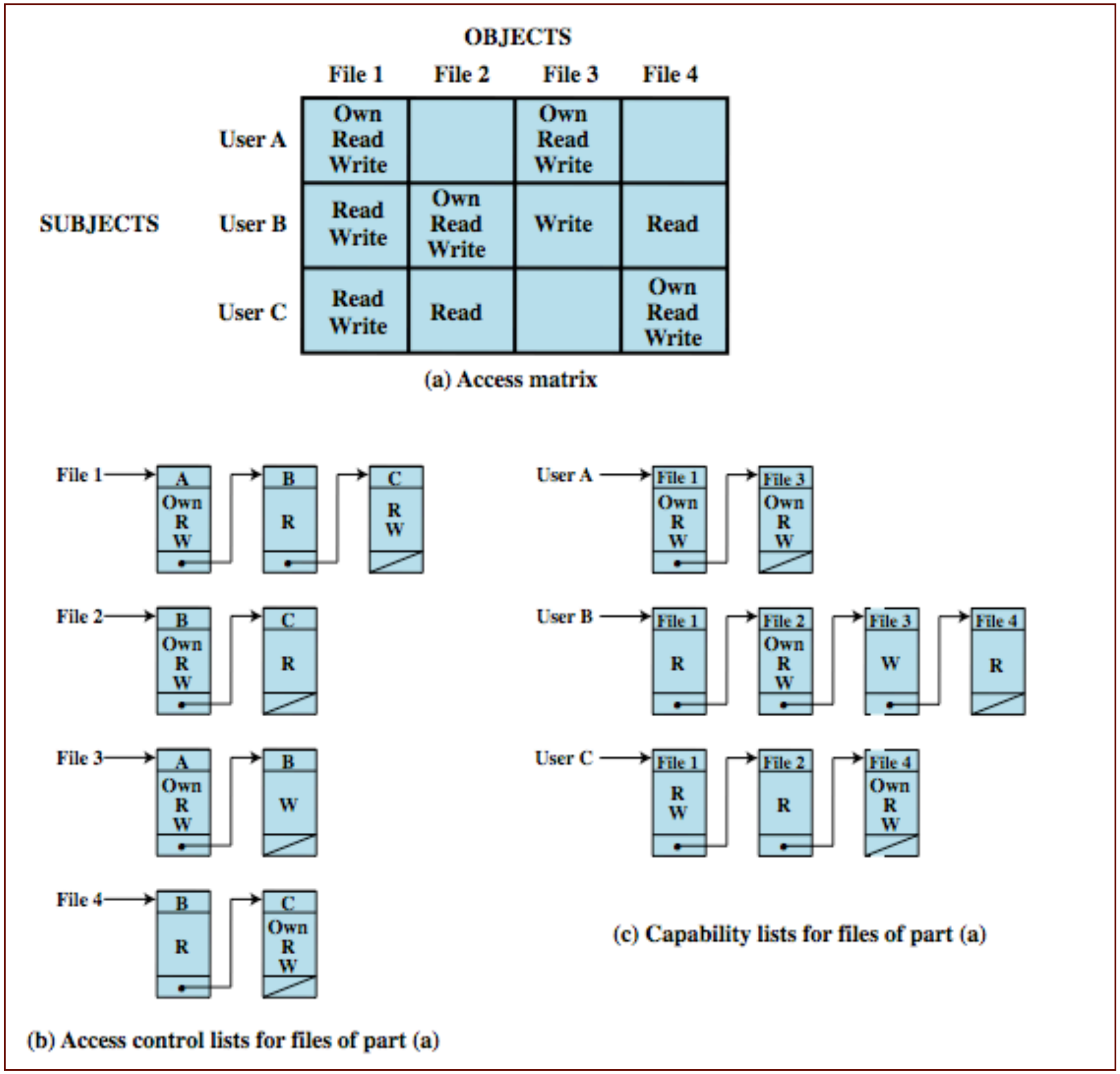
Access Control Elements

- subject - entity that can access objects
  - a process representing user/application
  - often have 3 classes: owner, group, world
- object - access controlled resource
  - e.g. files, directories, records, programs etc
  - number/type depend on environment
- access right - way in which subject accesses an object
  - e.g. read, write, execute, delete, create, search

Discretionary Access Control

- often provided using an access matrix
  - lists subjects in one dimension (rows)
  - lists objects in the other dimension (columns)
  - each entry specifies access rights of the specified subject to that object
- access matrix is often sparse
- can decompose by either row or column

Access Control Structures



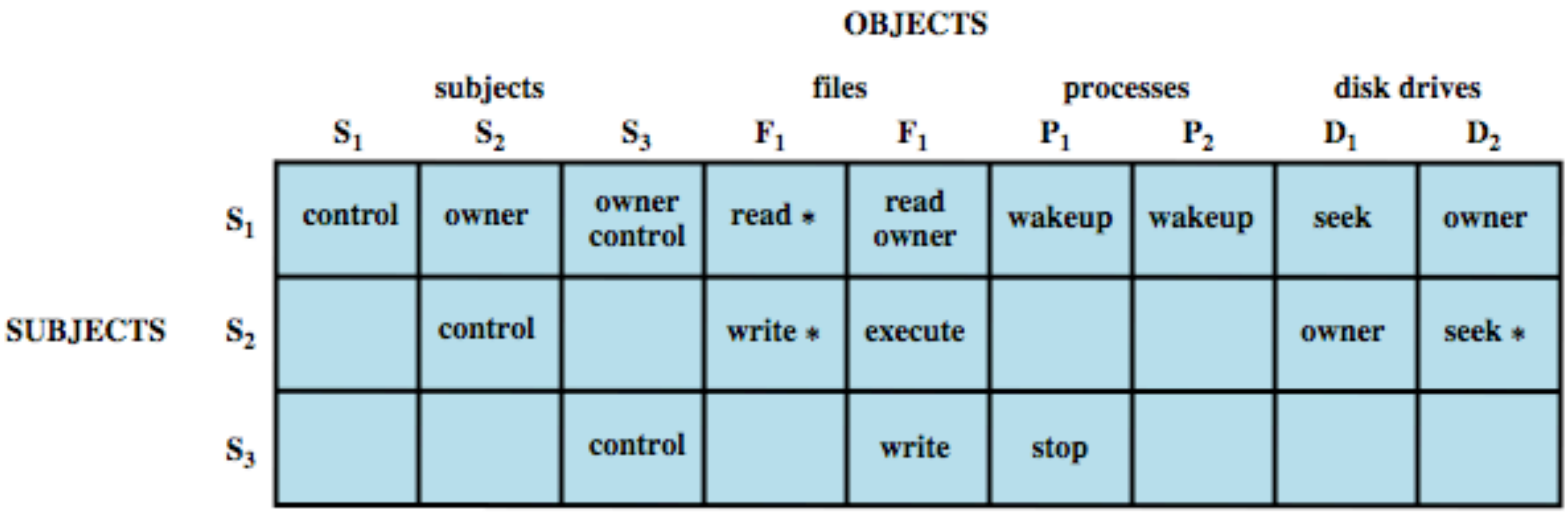
		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read Write	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

(a) Access matrix

(b) Access control lists for files of part (a)

(c) Capability lists for files of part (a)

Access Control Model

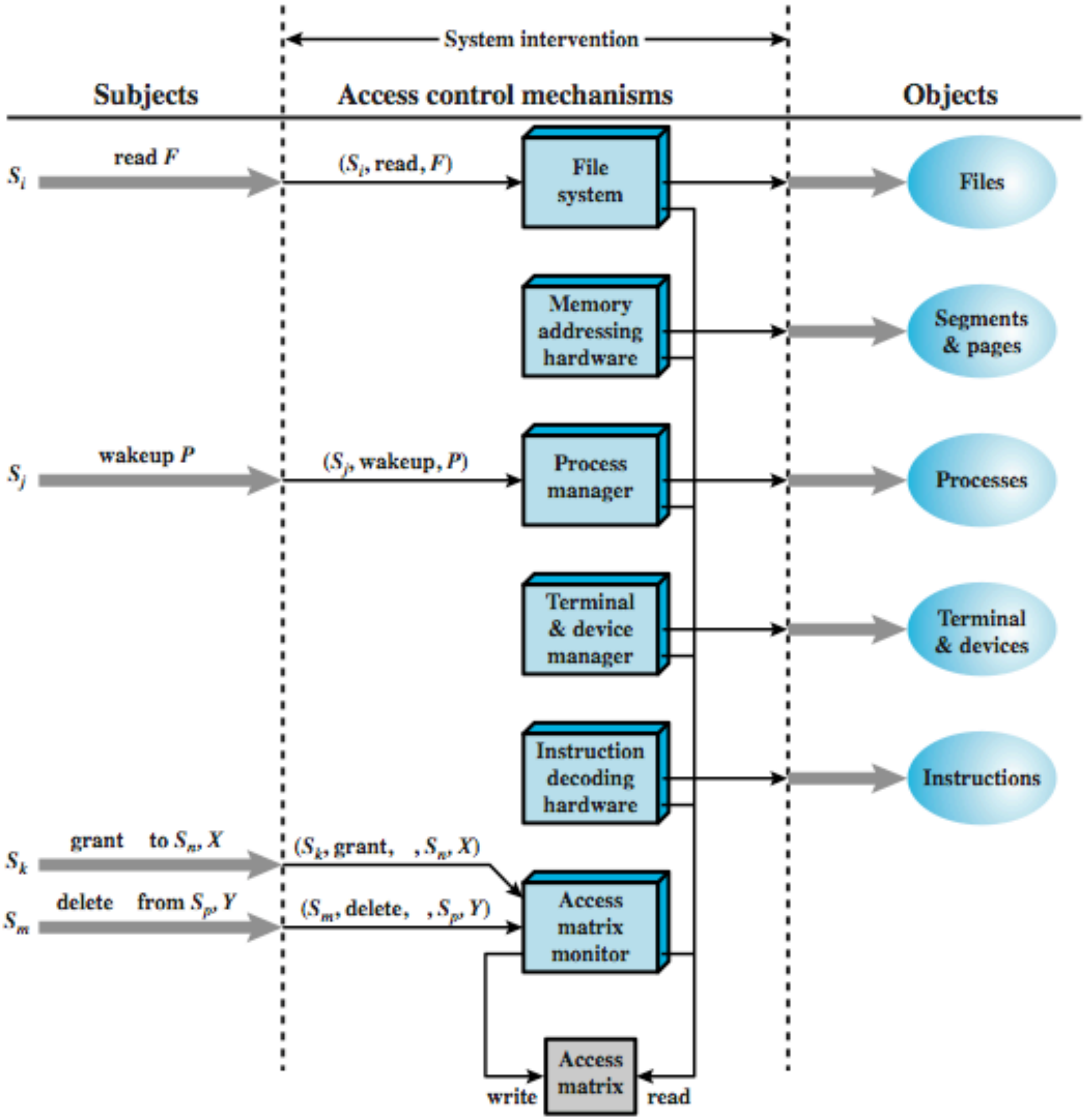


OBJECTS

		subjects			files		processes		disk drives	
		S_1	S_2	S_3	F_1	F_1	P_1	P_2	D_1	D_2
SUBJECTS	S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S_2		control		write *	execute			owner	seek *
	S_3			control		write	stop			

* - copy flag set

Access Control Function



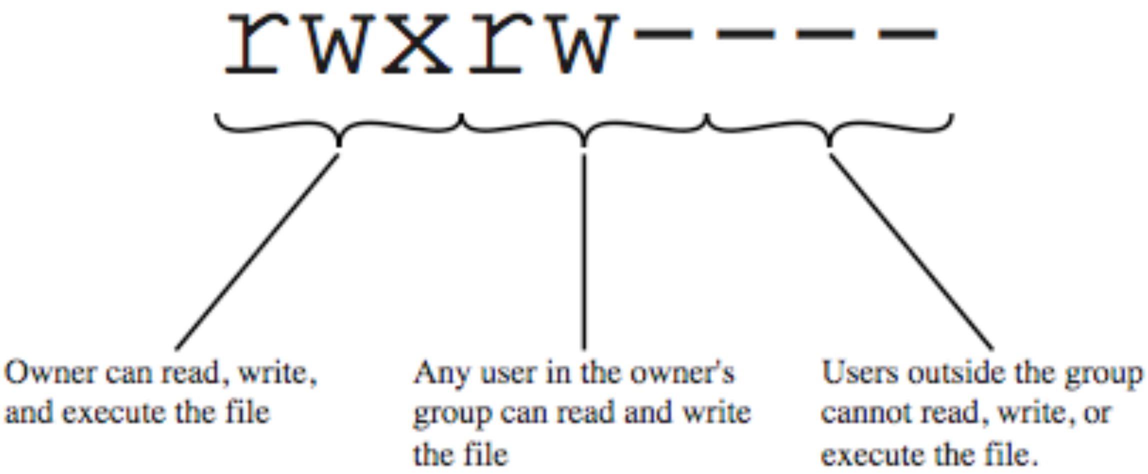
Protection Domains

- set of objects with associated access rights
- in access matrix view, each row defines a protection domain
  - but not necessarily just a user
  - may be a limited subset of user’s rights
  - applied to a more restricted process
- may be static or dynamic

UNIX File Concepts

- UNIX files administered using inodes
  - control structure with key info on file
    - attributes, permissions of a single file
  - may have several names for same inode
  - have inode table / list for all files on a disk
    - copied to memory when disk mounted
- directories form a hierarchical tree
  - may contain files or other directories
  - are a file of names and inode numbers

UNIX File Access Control



rwxrw----

Owner can read, write, and execute the file

Any user in the owner's group can read and write the file

Users outside the group cannot read, write, or execute the file.

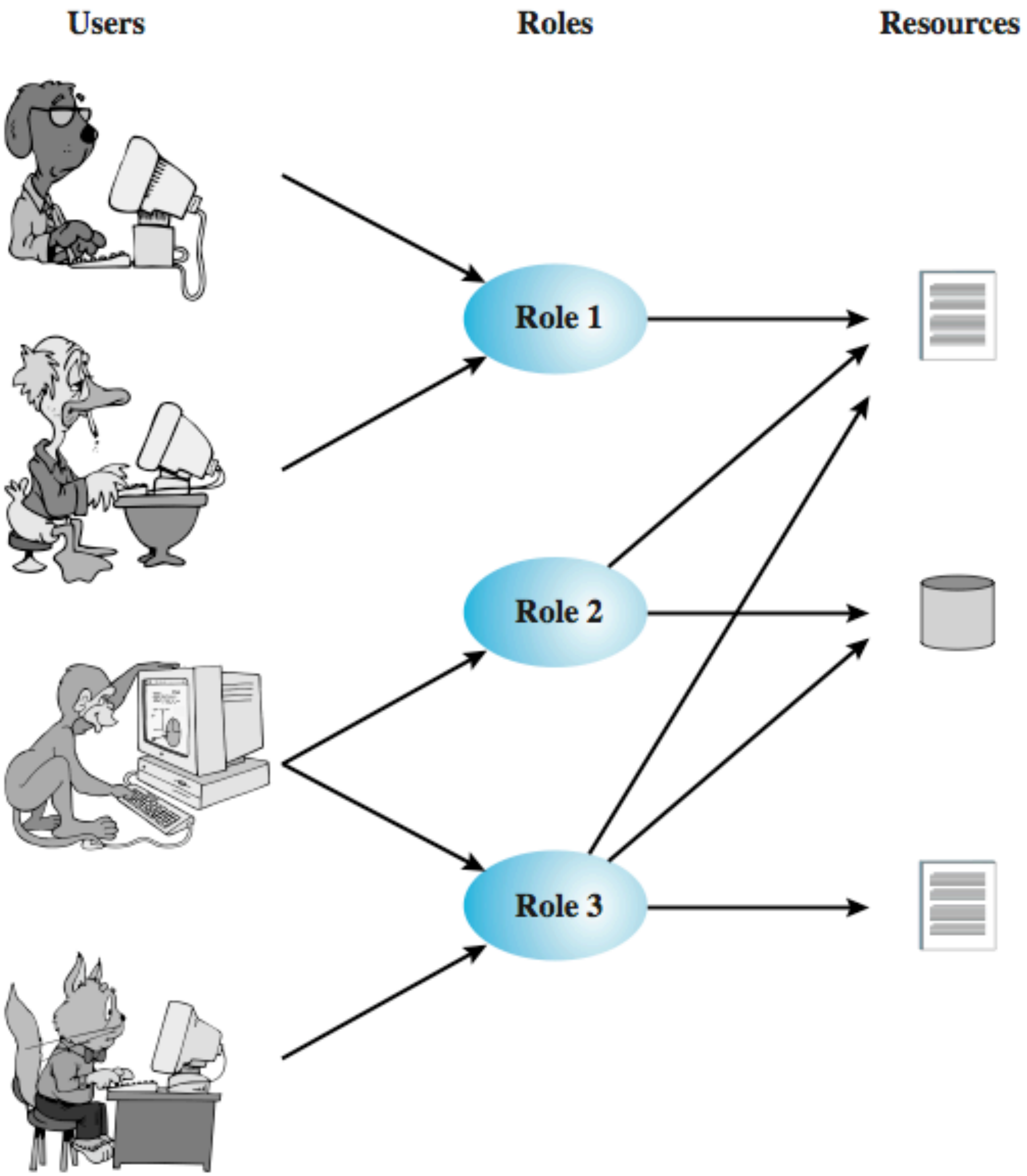
UNIX File Access Control

- “set user ID”(SetUID) or “set group ID”(SetGID)
  - system temporarily uses rights of the file owner / group in addition to the real user’s rights when making access control decisions
  - enables privileged programs to access files / resources not generally accessible
- sticky bit
  - on directory limits rename/move/delete to owner
- superuser
  - is exempt from usual access control restrictions

UNIX Access Control Lists

- modern UNIX systems support ACLs
- can specify any number of additional users / groups and associated rwx permissions
- ACLs are optional extensions to std perms
- group perms also set max ACL perms
- when access is required
  - select most appropriate ACL
    - owner, named users, owning / named groups, others
  - check if have sufficient permissions for access

Role-Based Access Control

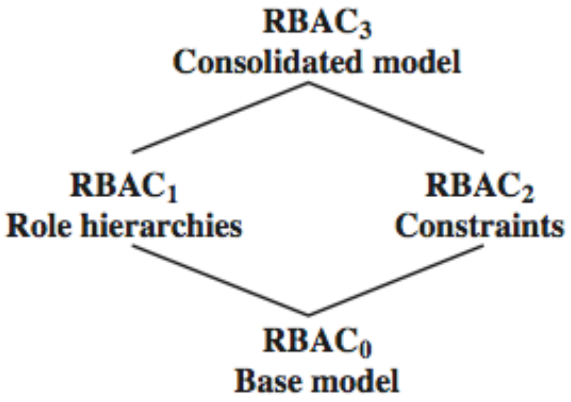


Role-Based Access Control

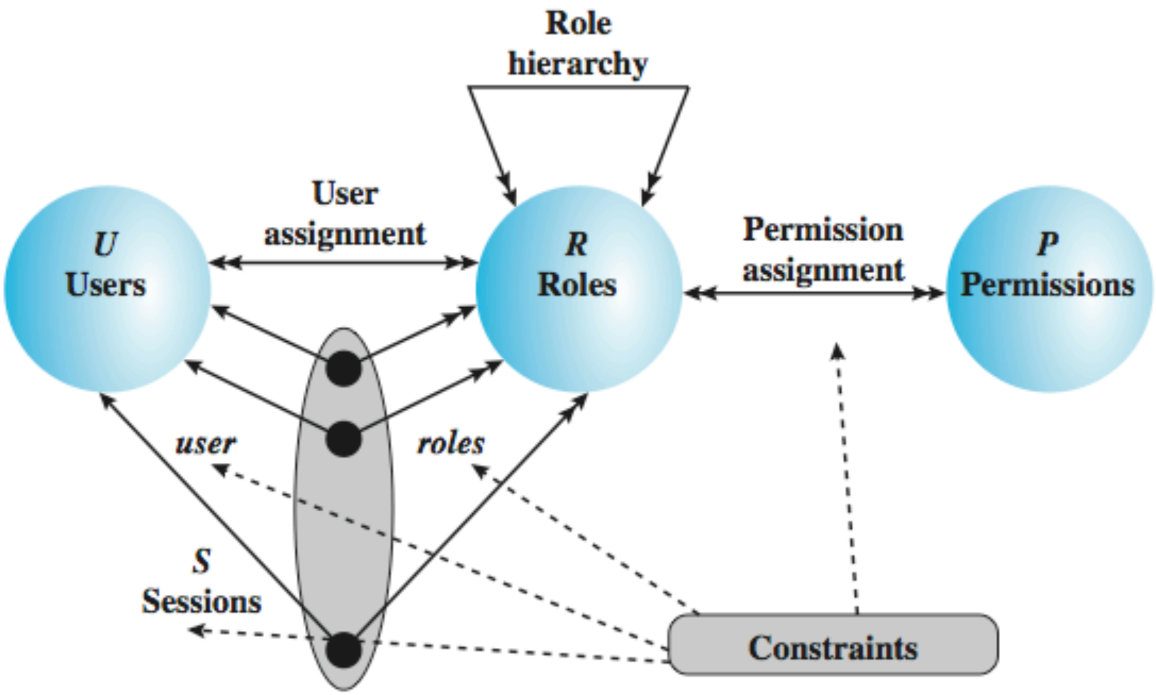
	R_1	R_2	• • •	R_n
U_1	✖			
U_2	✖			
U_3		✖		✖
U_4				✖
U_5				✖
U_6				✖
⋮				
U_m	✖			

		OBJECTS								
		R_1	R_2	R_n	F_1	F_1	P_1	P_2	D_1	D_2
ROLES	R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	R_2		control		write *	execute			owner	seek *
	⋮									
	R_n			control		write	stop			

Role-Based Access Control

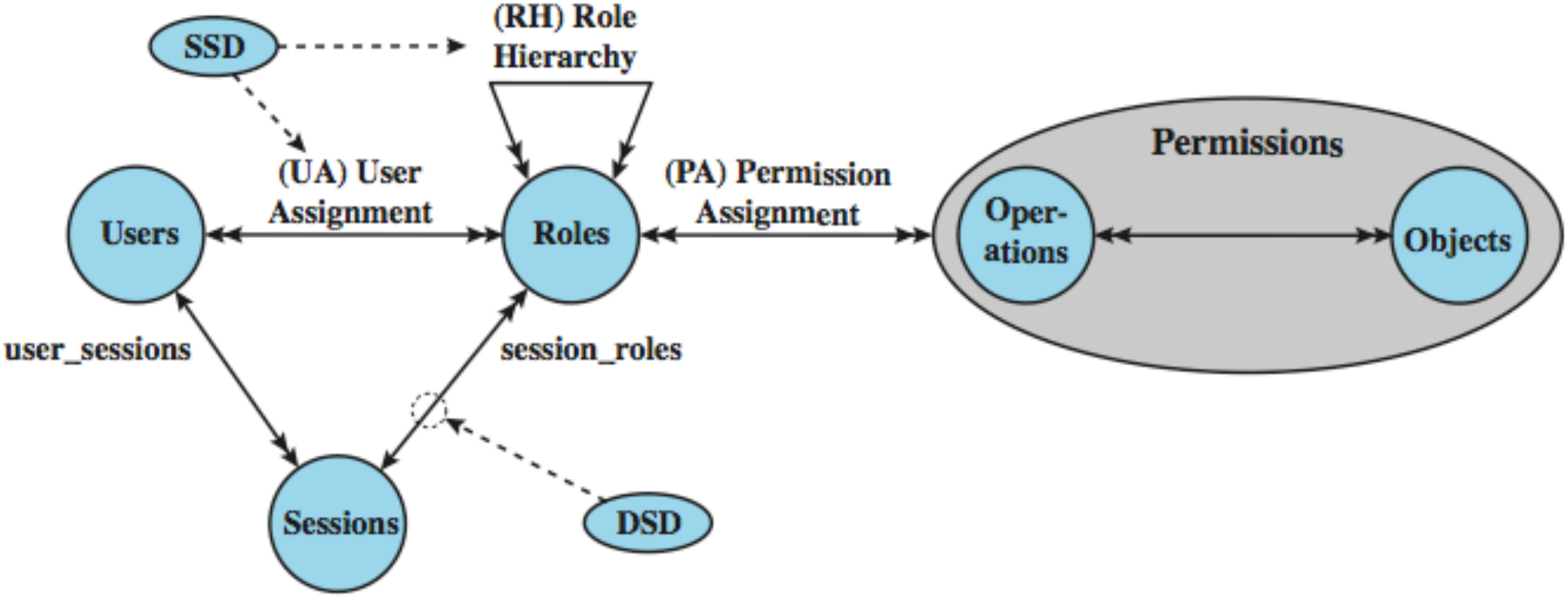


(a) Relationship among RBAC models



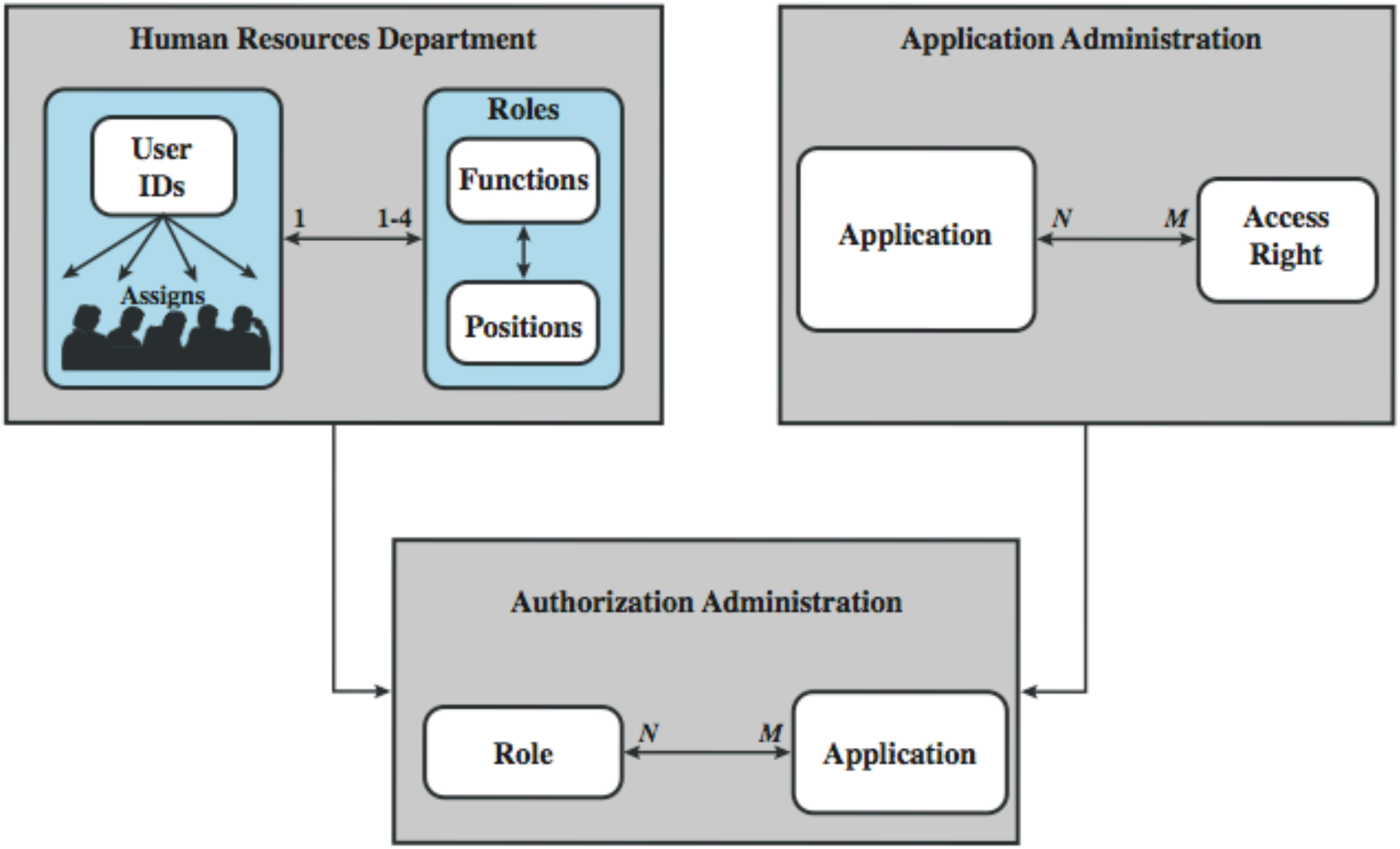
(b) RBAC models

NIST RBAC Model



SSD = static separation of duty
DSD = dynamic separation of duty

RBAC For a Bank



Summary

- introduced access control principles
  - subjects, objects, access rights
- discretionary access controls
  - access matrix, access control lists (ACLs), capability tickets
  - UNIX traditional and ACL mechanisms
- role-based access control
- case study

What goes wrong

- huge systems, many bugs, many users
- known vulnerabilities
- scripts circulating
- posted to CERT or vendor (or not)
- patches
- reverse-engineering -> exploits
- goal: get access to normal account, become sysadmin. Now: many programs as admin, when compromised give admin rights

Attacks

1)Smashing the stack, Stack overflow

2)Format string vulnerability

3)SQL insertion

4)Race conditions

Exercise

- Read about your attack (5 min)
- Thi
  n
  k about how to prevent it, recover from it (5 min)
- Form groups of 4 people (turn around)
- Everyone explains their attack to the group (15)
- Discuss remedies, brainstorm on others (5 min)
- Collect remedies for the class

Remedies

- sql insertion: don't print error messages, escape characters, don't evaluate user input as code
- formating: parse data before use
- stack smashing: executable bits on pages, machine-level memory protection
- race condition: make file operation atomic, lock operations

Remedies

- proper bounds checking in C
- (even automated, compiler patch StackGuard)
- tools, training
- better design, coding, testing
- principle of least privilege
- default config safe

Summary

- AC at many levels, more expressive on upper levels, but more vulnerable
- Most attacks exploit bugs, environment creep
- Main function of AC is to limit the damage that can be done by particular groups, users, and programs whether through error or malice.