Computer Security DD2395

http://www.csc.kth.se/utbildning/kth/kurser/DD2395/dasak11/

Fall 2011

Sonja Buchegger
buc@kth.se

Lecture 4

Access Control

Access Control

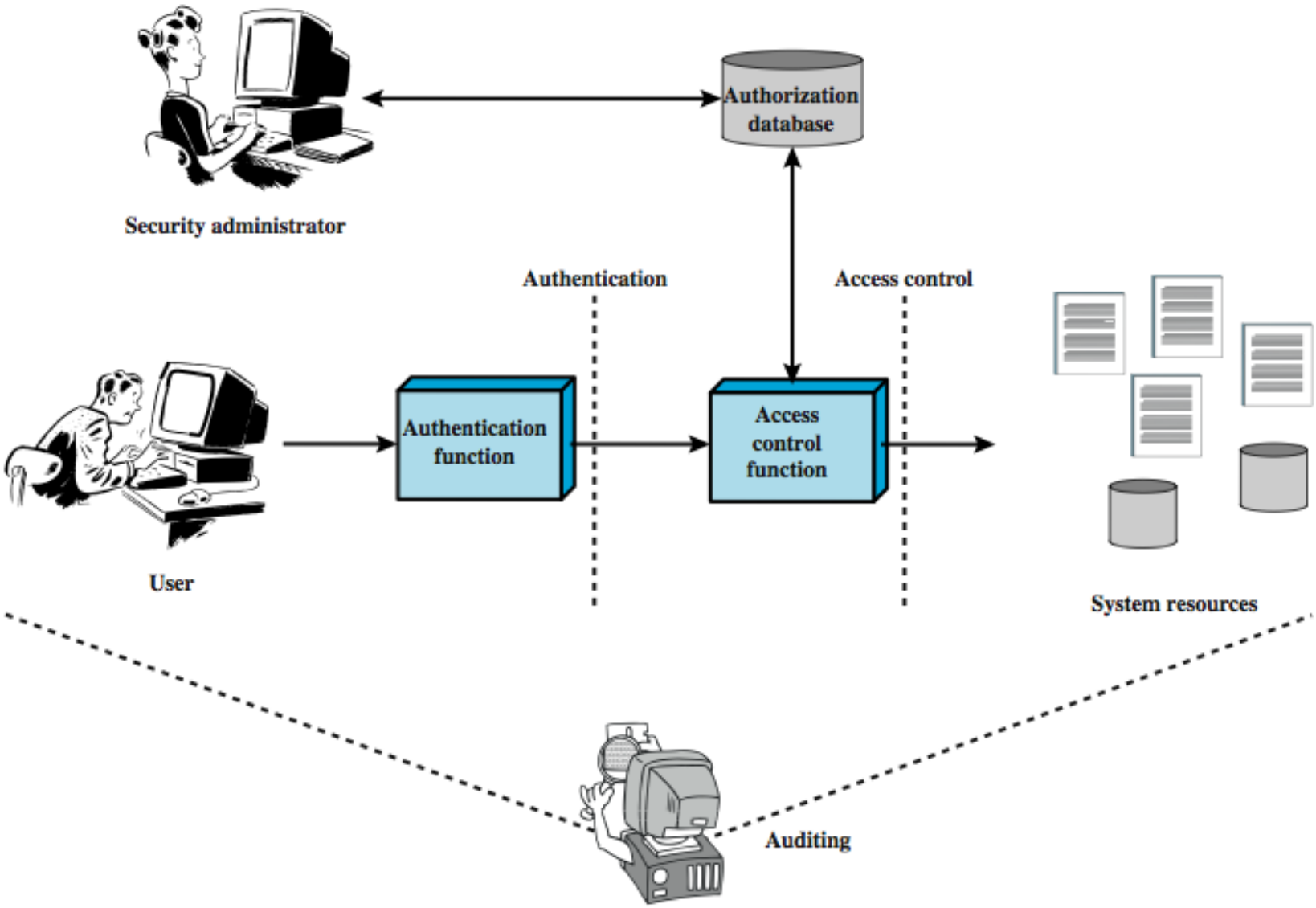
- The prevention of unauthorized use* of a resource, including the prevention of use of a resource in an unauthorized manner, and to enable legitimate users to access resources in an authorized manner
- *intentional or accidental
- central element of computer security

- Schneier: “We want to make sure that authorized people are able to do whatever they are authorized to do, and that everyone else is not”
- Like access to buildings, now on computers
- History: shared access, stand-alone, now networked

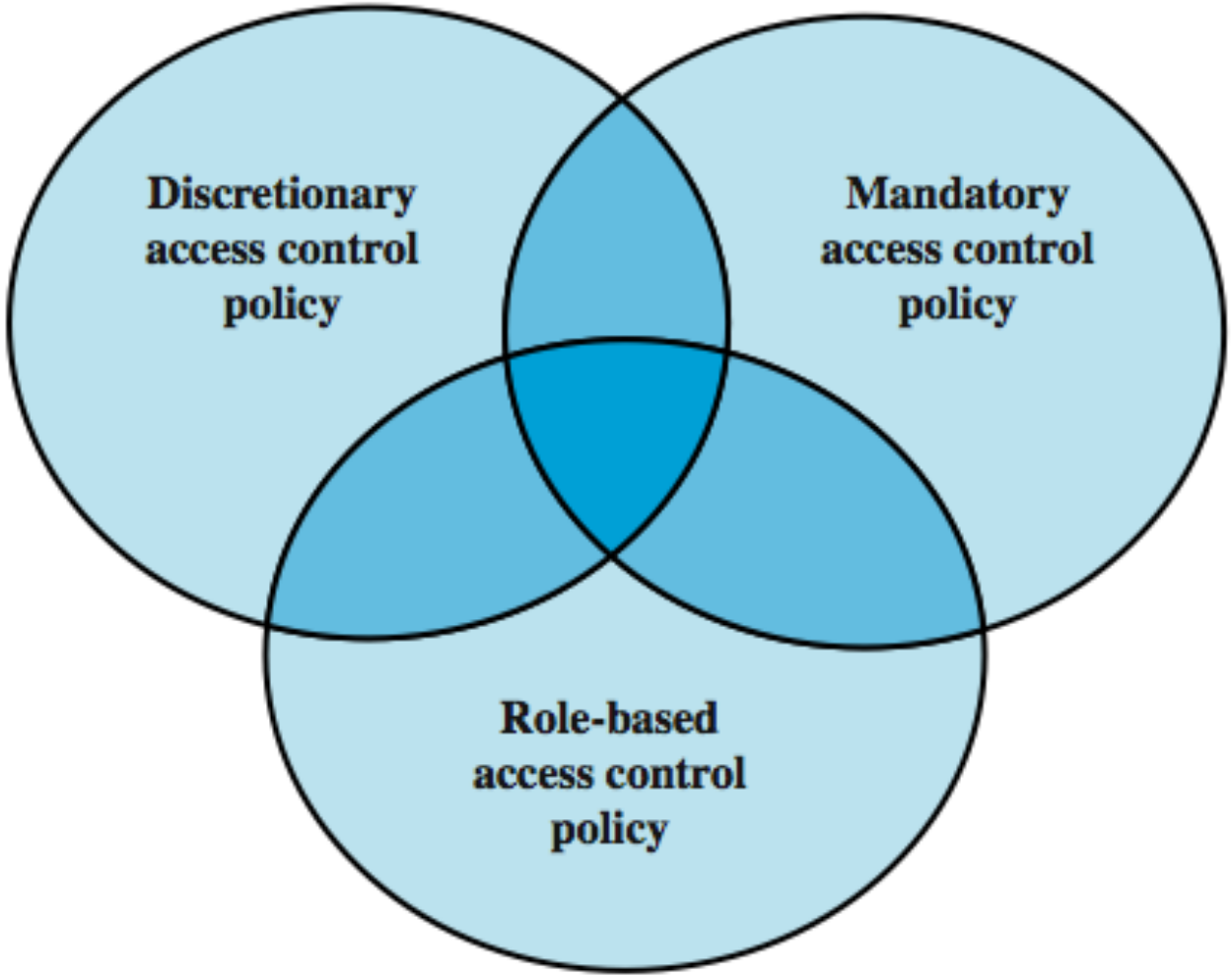
Access Control

- CIA triangle: Confidentiality, Integrity, Availability
- All need access control
- Assume users and groups
  - authenticate to system (who? Has any rights?)
  - assigned access rights to certain resources on system (authorization)
  - audit

Access Control Principles



Access Control Policies



Access Control Requirements

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

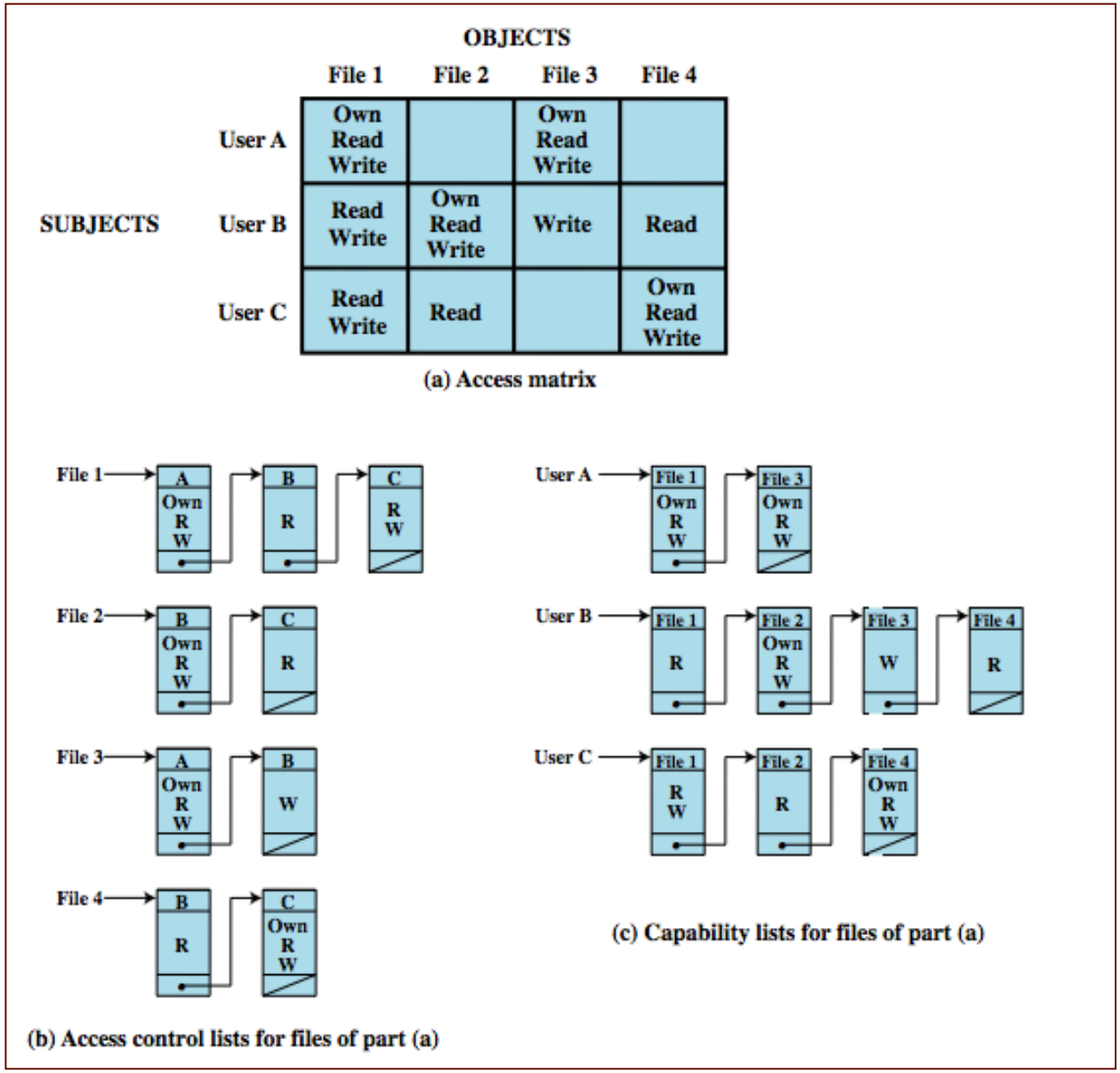
Access Control Elements

- 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



What would you do?

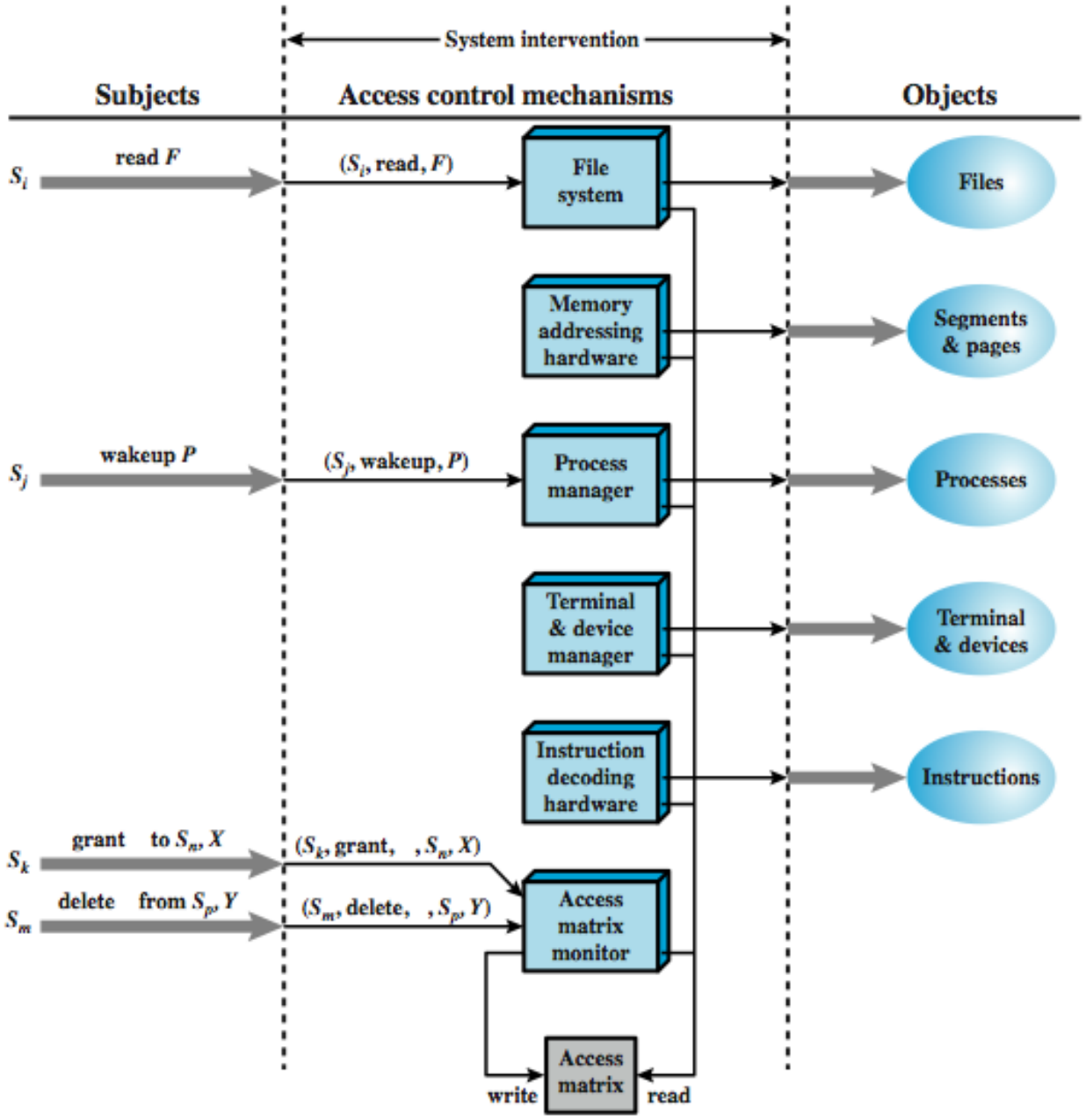
- Case: Employee leaves the company, you want to remove their access rights.
- Access control list vs. capabilities

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

What would you do?

- Why would you run a program with fewer access rights?

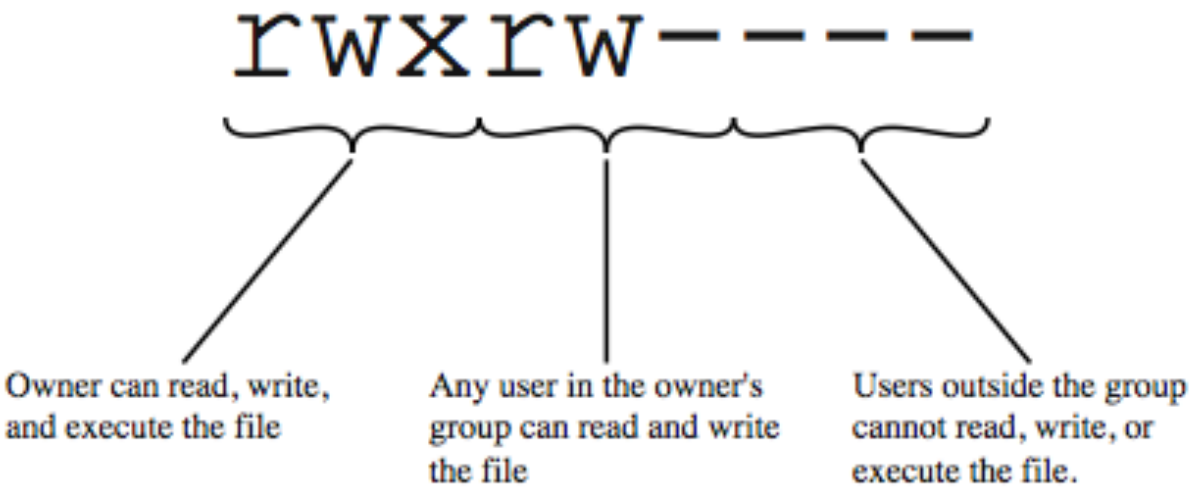
Alternatives

- Sandboxing
- Proof-carrying code
- Virtual machines
- Trusted computing

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



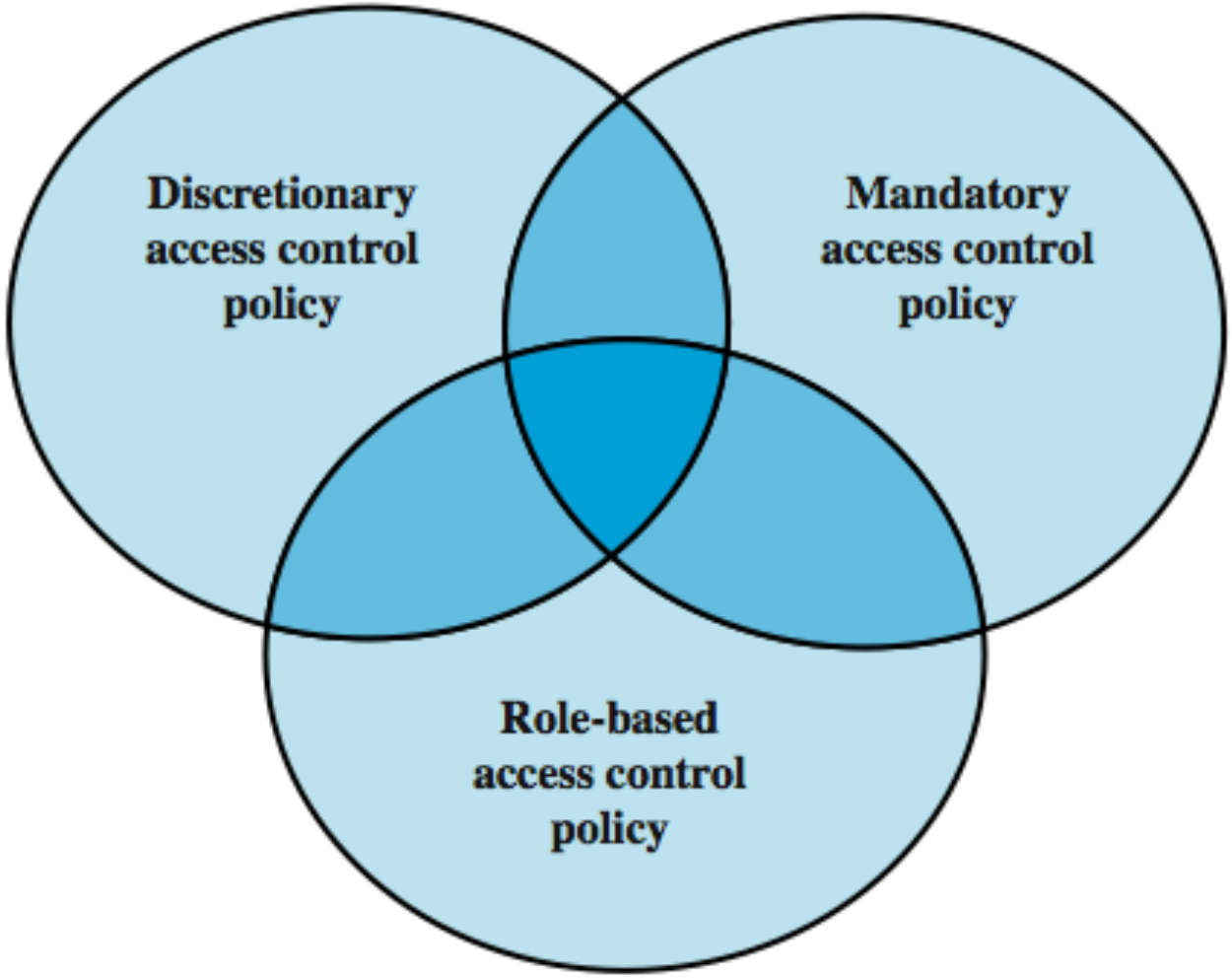
UNIX File Access Control

- "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

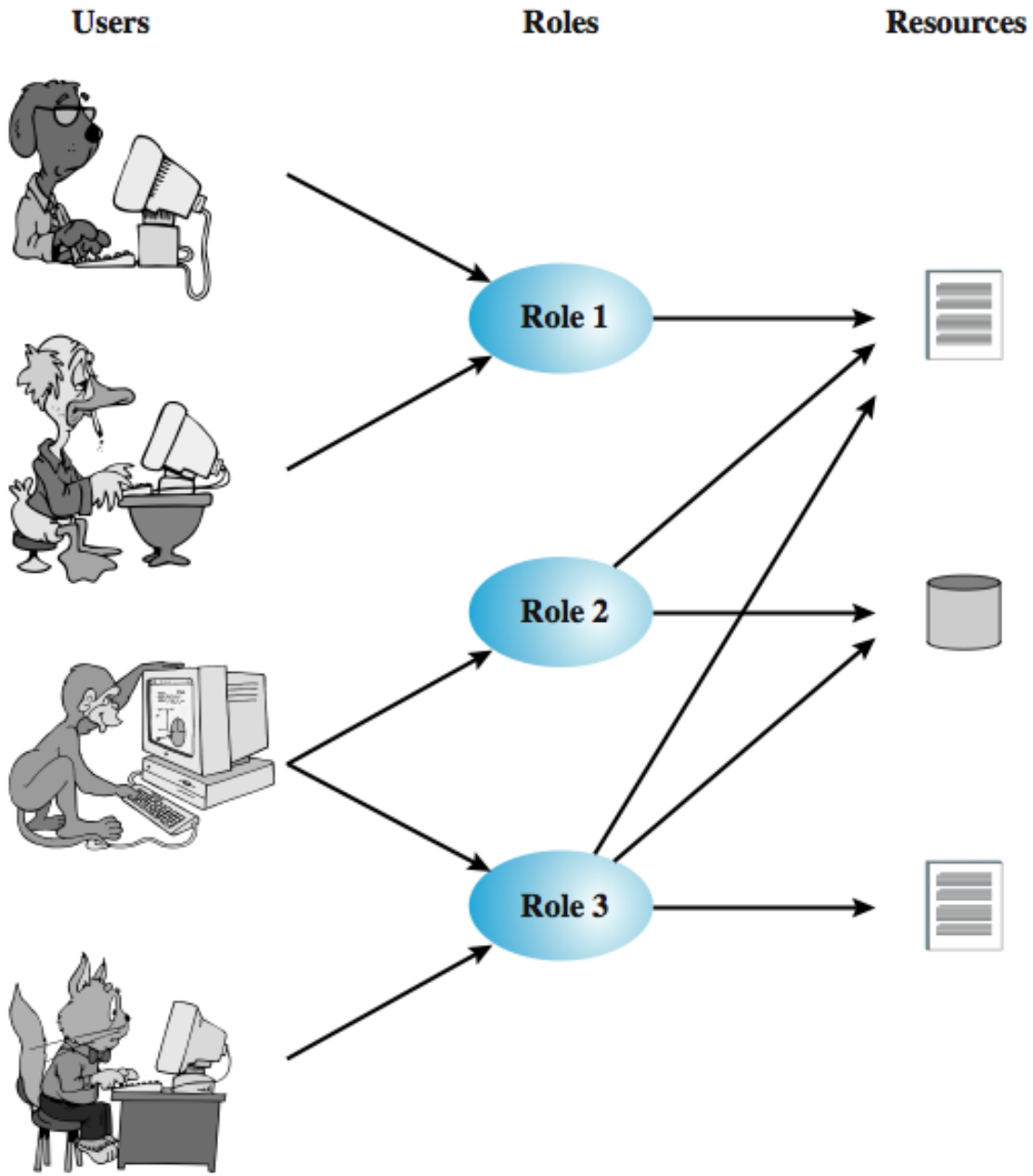
UNIX Access Control Lists

- 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

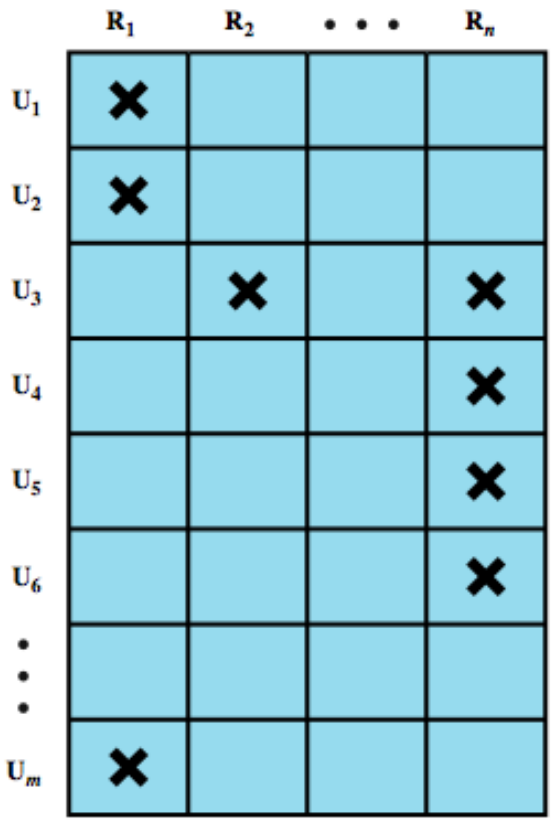
Access Control Policies



Role-Based Access Control



Role-Based Access Control



OBJECTS

ROLES	R_1	R_2	R_n	F_1	F_1	P_1	P_2	D_1	D_2
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			

- How can RBAC be used to deal with access rights removal when an employee leaves a company?

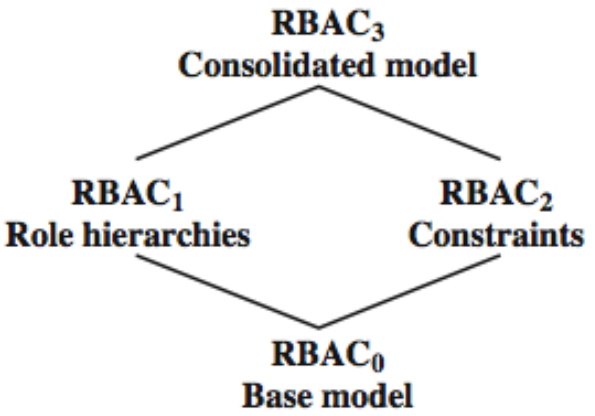
Hierarchies

- By convention, inheritance inverse to role hierarchy

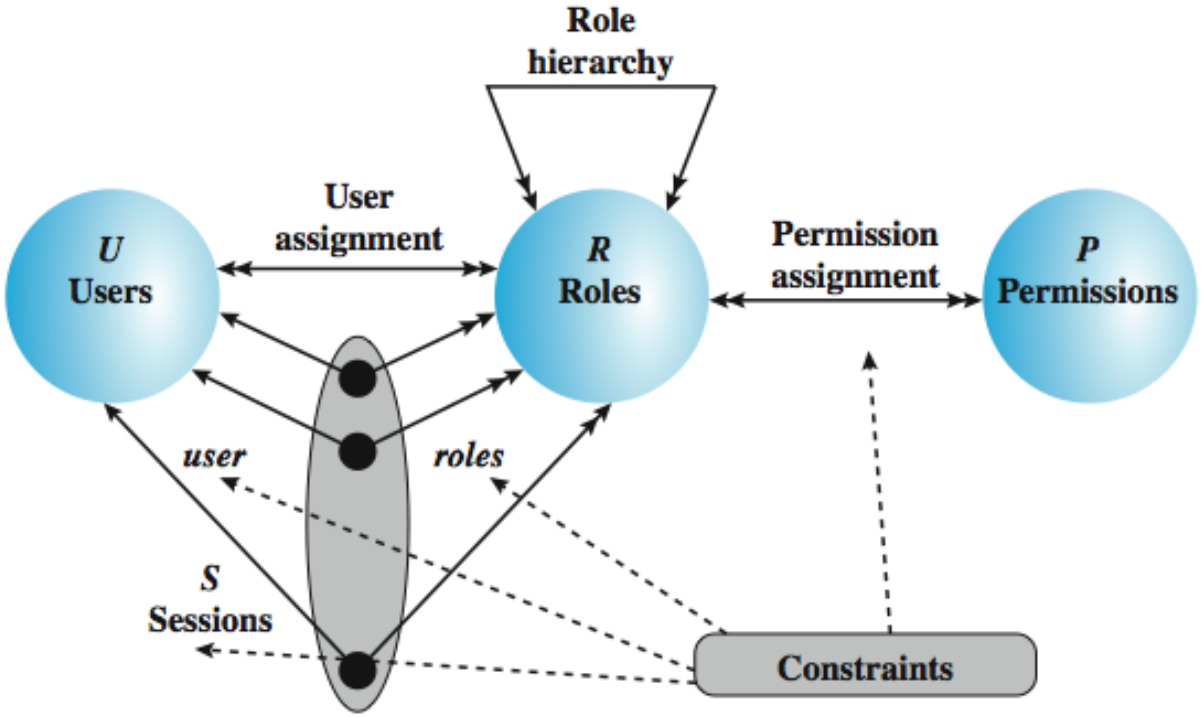
Constraints

- Mutually exclusive roles
  - A user can only be assigned to one role in a set, in a session or statically
  - Any permission can be granted to only one role in the set
- Cardinality: only n users per role, n roles per user
- Prerequisite, can be hierarchical

Role-Based Access Control

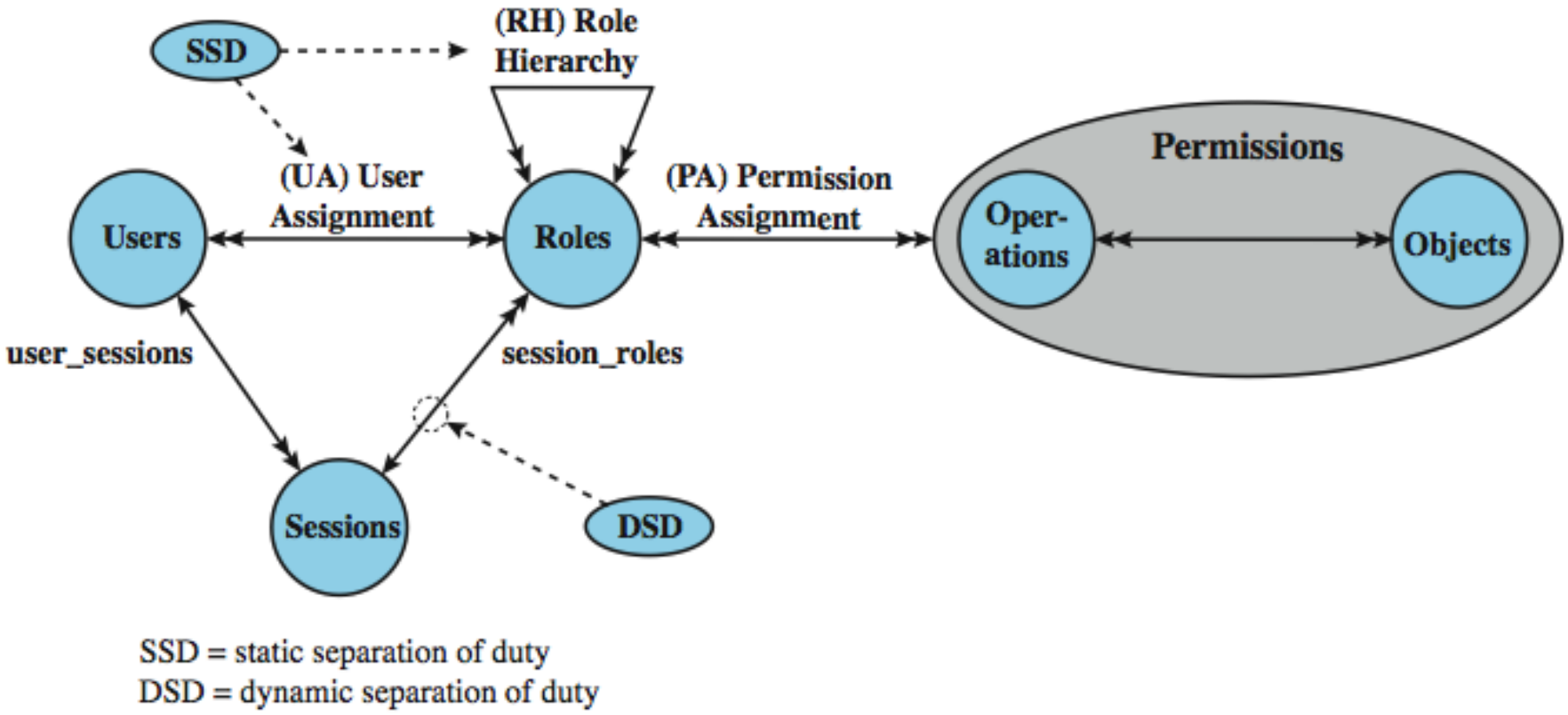


(a) Relationship among RBAC models

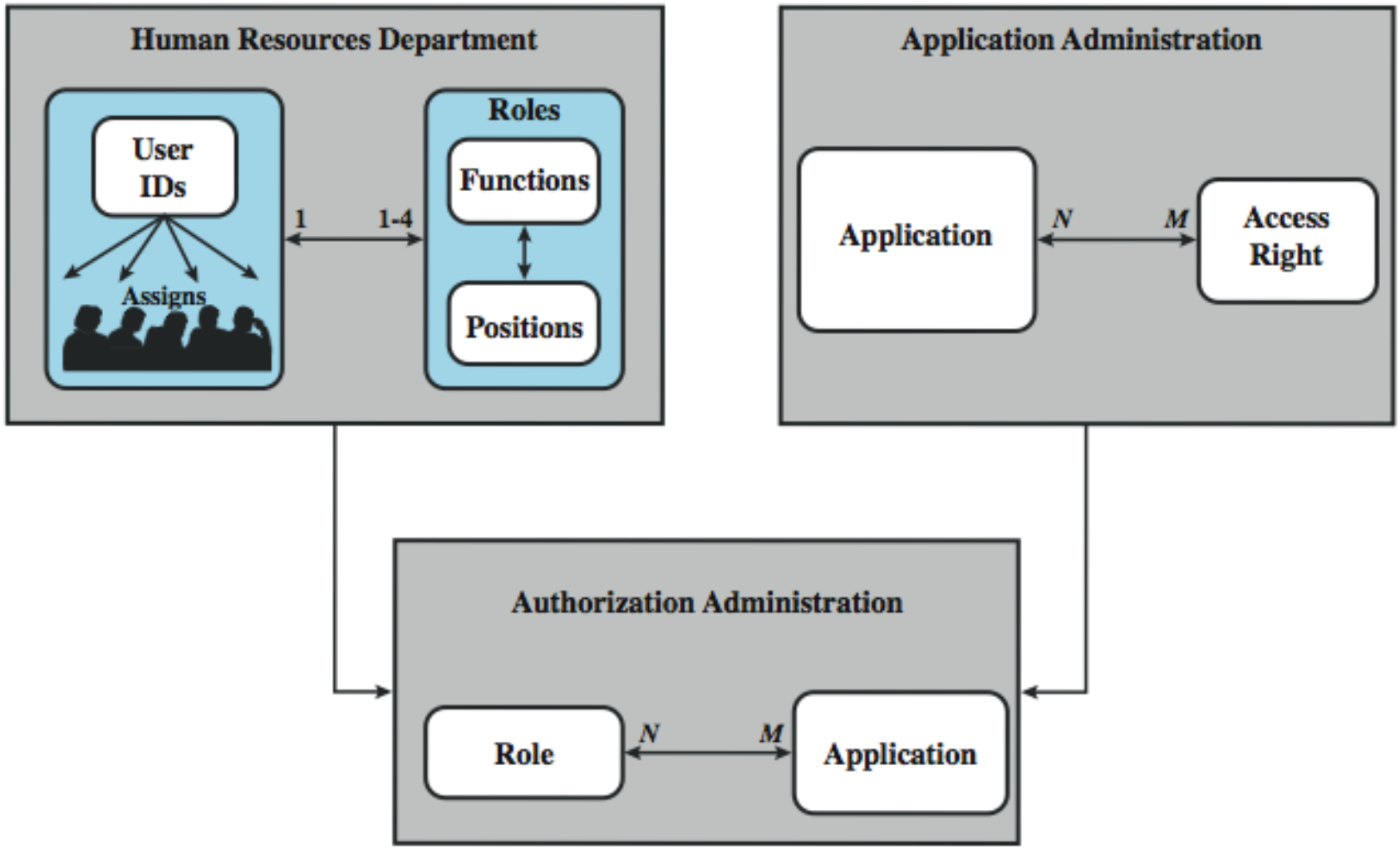


(b) RBAC models

NIST RBAC Model



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

Type Safety:

- Smashing the stack, Stack overflow
  overlong input, data gets executed
  example: finger

- Format string vulnerability, e.g. printf, formatting instructions get interpreted, can write to stack
- SQL insertion

Attacks

Timing:

- Race conditions, e.g. mkdir, login, tmp
- Overwrite userid while password is being validated
- Create directory in two steps: allocate storage, transfer rights to user
- Tmp file by privileged user, change to symbolic link, file will be removed

Remedies?

- Remedies

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

User Interface Failures

- Trojan horse
- Games that check for admin access
- Same name as other programs, e.g. ls
- When users need admin rights to install anything
- Active content, e.g. macros

Why do things go wrong?

- OS and program size, complexity
- Bugs
- Bugs publicized, no all reported
- Patches not applied
- Patch Tuesday, exploit Wednesday reverse attacks
- Programs running as root

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.