NETWORK AND SYSTEM SECURITY (CORE ELECTIVE - 5) CS424

Unit 3 : SYSTEM SECURITY

Unit 3 Overview

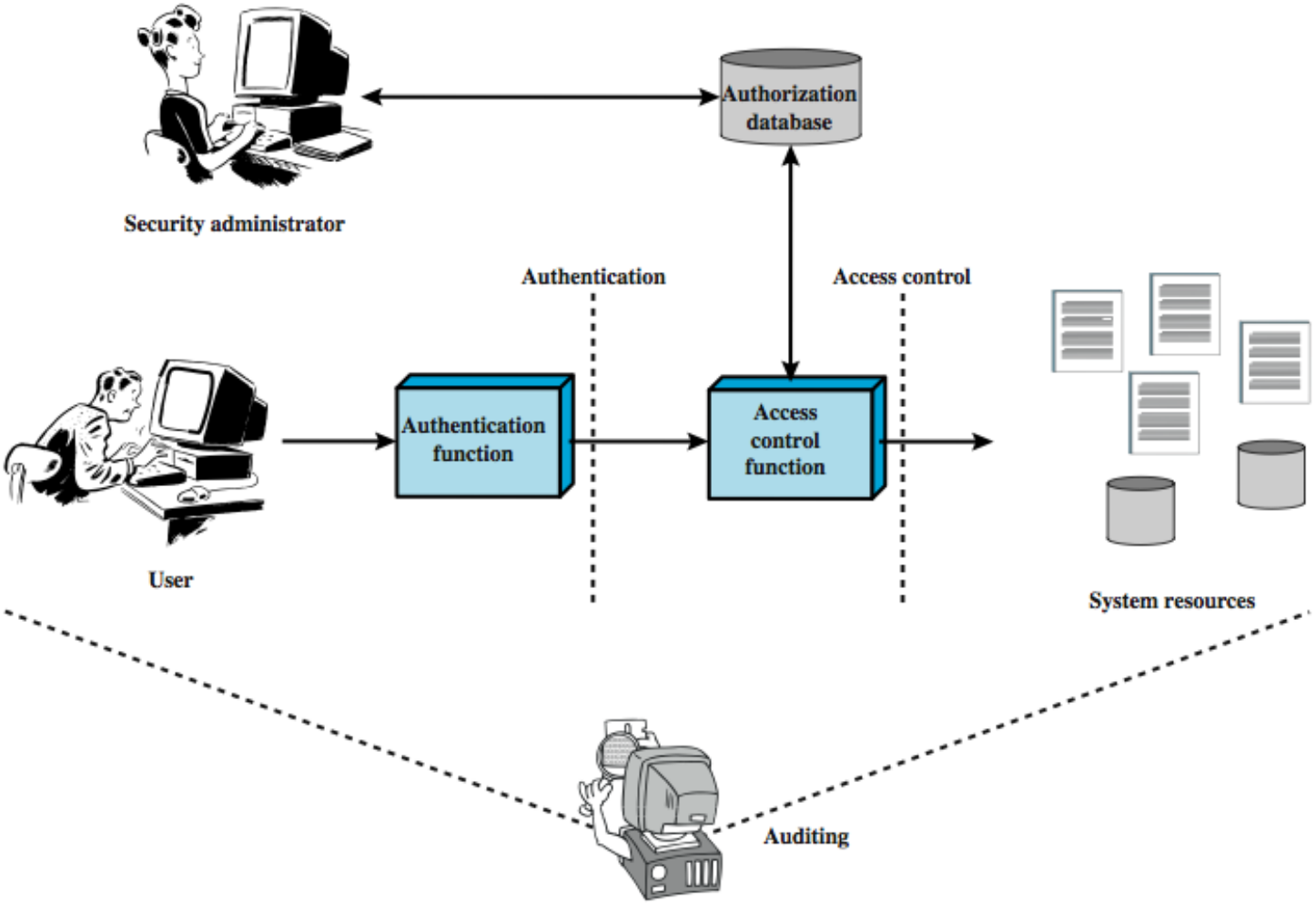
- User Authentication
- **Access Control**
- Database and Cloud security
- Malicious Software
- Denial of Service Attack
- Intrusion Detection
- Intrusion Prevention

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 Principles…

- **Authentication:** Verification that the credentials of a user or other system entity are valid.
- A**uthorization:** The granting of a right or permission to a system entity to access a system resource. This function determines who is trusted for a given purpose.
- Au**dit:** An independent review and examination of system records and activities in order to test for adequacy of system controls.

Access control policies

An access control policy, which can be embodied in an authorization database, dictates what types of access are permitted, under what circumstances, and by whom.

- **Discretionary** access control (DAC): based on the identity of the requestor and access rules (an entity might have access rights that permit the entity, by its own volition, to enable another entity to access some resource.)
- **Mandatory** access control (MAC): based on comparing security labels with security clearances (mandatory: one with access to a resource cannot pass to others)
- **Role-based** access control (RBAC): based on user roles
- **Attribute-based** access control: based on the attributes of the user, the resources and the current environment
- These four policies are not mutually exclusive

Access Control Requirements

- Reliable input: a mechanism to authenticate
- Fine and coarse specifications: regulate access at varying levels (e.g., an attribute or entire DB)
- Least privilege: min authorization to do its work
- Separation of duty: divide steps among different individuals
- Open and closed policies: accesses specifically authorized or all accesses except those prohibited
- Administrative policies: who can add, delete, modify rules

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

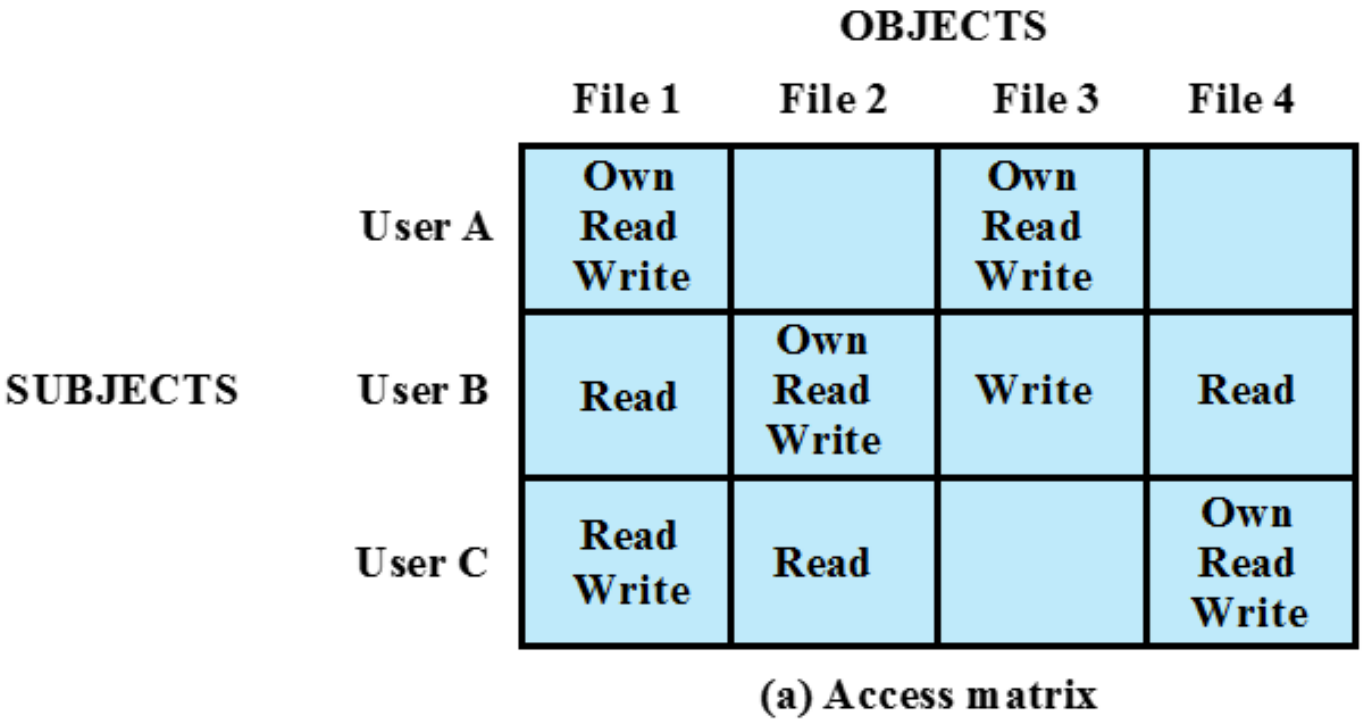
Discretionary Access Control

- 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

- Access control lists (decomposed by column)
- Capability tickets (decomposed by row)
- Also see alternative table representation (tabular but not sparse)

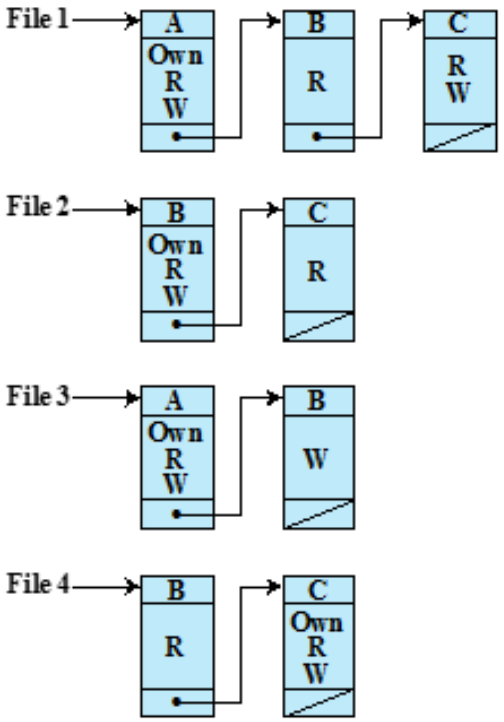
An access matrix



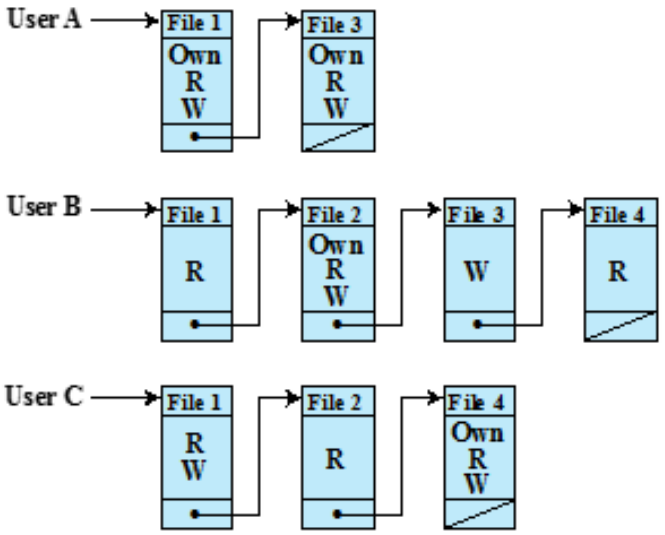
SUBJECTS	OBJECTS: File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

(a) Access matrix

Access matrix data structures



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



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

Capability tickets

- Each user has a number of tickets and may be authorized to loan or give them to others.
- Because tickets may be dispersed around the system, they present a greater security problem than access control lists.
- The integrity of the ticket must be protected, and guaranteed (usually by the operating system).
- In particular, the ticket must be unforgeable.

Capability tickets…

- One way to accomplish this is to have the operating system hold all tickets on behalf of users.
- These tickets would have to be held in a region of memory inaccessible to users.
- Another alternative is to include an unforgeable token in the capability.
- This could be a large random password, or a cryptographic message authentication code.
- This value is verified by the relevant resource whenever access is requested

Alternate authorization table

- The convenient and inconvenient aspects of capability tickets are the opposite of those for ACLs.
- It is easy to determine the set of access rights that a given user has.
- But more difficult to determine the list of users with specific access rights for a specific resource.

Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

An Access Control Model

- Protection state of a system to be the set of information, at a given point in time, that specifies the access rights for each subject with respect to each object.

Three requirements:

- representing the protection state,
- enforcing access rights,
- and allowing subjects to alter the protection state in certain ways

An Access Control Model

- To represent the protection state, we extend the universe of objects in the access control matrix to include the following:
- **Processes:** Access rights include the ability to delete a process, stop (block), and wake up a process.
- **Devices:** Access rights include the ability to read/write the device, to control its operation (e.g., a disk seek), and to block/unblock the device for use.
- **Memory locations or regions:** Access rights include the ability to read/write certain regions of memory that are protected such that the default is to disallow access.
- **Subjects:** Access rights with respect to a subject have to do with the ability to grant or delete access rights of that subject to other objects, as explained subsequently.

An Access Control Model

- Extend the universe of objects to include processes, devices, memory locations, subjects

		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

An Access Control Model…

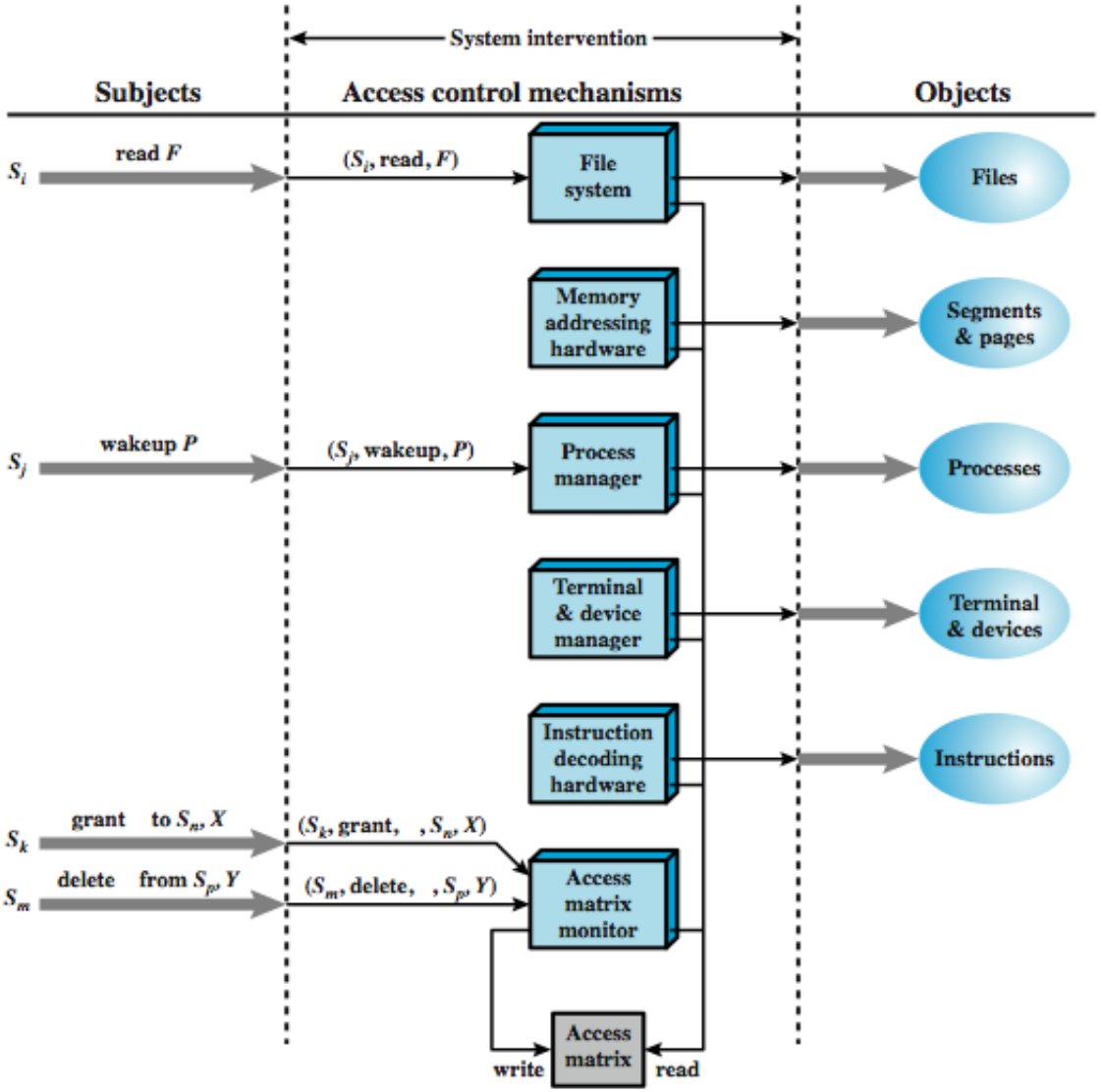
- The module evaluates each request by a subject to access an object to determine if the access right exists.
- An access attempt triggers the following steps:

1. A subject *S*0 issues a request of type α for object *X*.

2. The request causes the system (the operating system or an access control interface module of some sort) to generate a message of the form (*S*0, α, *X*) to the controller for *X*.

3. The controller interrogates the access matrix *A* to determine if α is in *A*[*S*0, *X*]. If so, the access is allowed; if not, the access is denied and a protection violation occurs. The violation should trigger a warning and appropriate action.

Access Control Function



Access control system commands

Rule	Command (by S_0)	Authorization	Operation
R1	**transfer** $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ **to** S, X	'α*' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R2	**grant** $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ **to** S, X	'owner' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R3	**delete** α **from** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	delete α from $A[S, X]$
R4	$w \leftarrow$ **read** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	copy $A[S, X]$ into w
R5	**create object** X	None	add column for X to A; store 'owner' in $A[S_0, X]$
R6	**destroy object** X	'owner' in $A[S_0, X]$	delete column for X from A
R7	**create subject** S	none	add row for S to A; execute **create object** S; store 'control' in $A[S, S]$
R8	**destroy subject** S	'owner' in $A[S_0, S]$	delete row for S from A; execute **destroy object** S

Protection Domains: More Useful

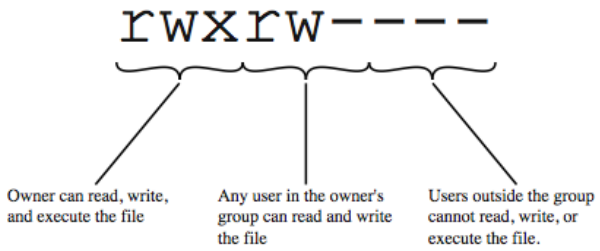
- Set of objects together with access rights to those objects
- More flexibility when associating capabilities with protection domains
- In terms of the access matrix, a row defines a protection domain
- User can spawn processes with a subset of the access rights of the user
- Association between a process and a domain can be static or dynamic
- In user mode certain areas of memory are protected from use and certain instructions may not be executed
- In kernel mode privileged instructions may be executed and protected areas of memory may be accessed

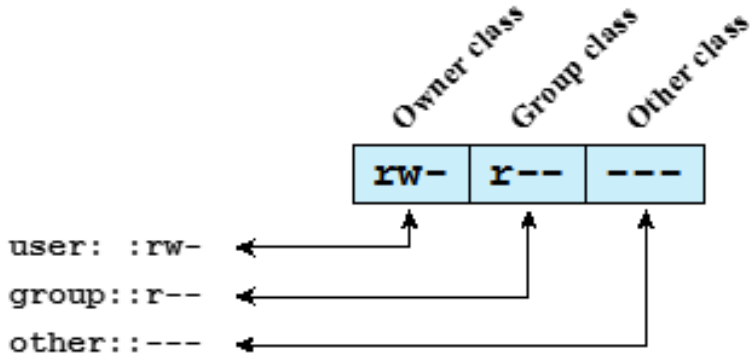
UNIX File Concepts

- UNIX files administered using inodes (index nodes)
- An inode:
  - control structure with key info on file (attributes, permissions, …)
  - on a disk: an inode table for all files
  - when a file is opened, its inode is brought to RAM
- Directories form a hierarchical tree
  - may contain files or other directories
  - are a file of names and inode numbers

UNIX File Access Control

- Unique user identification number (user ID)
- Member of a primary group identified by a group ID
- 12 protection bits
  - 9 specify read, write, and execute permission for the owner of the file, members of the group and all other users
  - 2 speficiy SetID, SetGID
  - 1 is the sticky bit (only owner can remove, delete, …, a directory)
- The owner ID, group ID, and protection bits are part of the file's inode





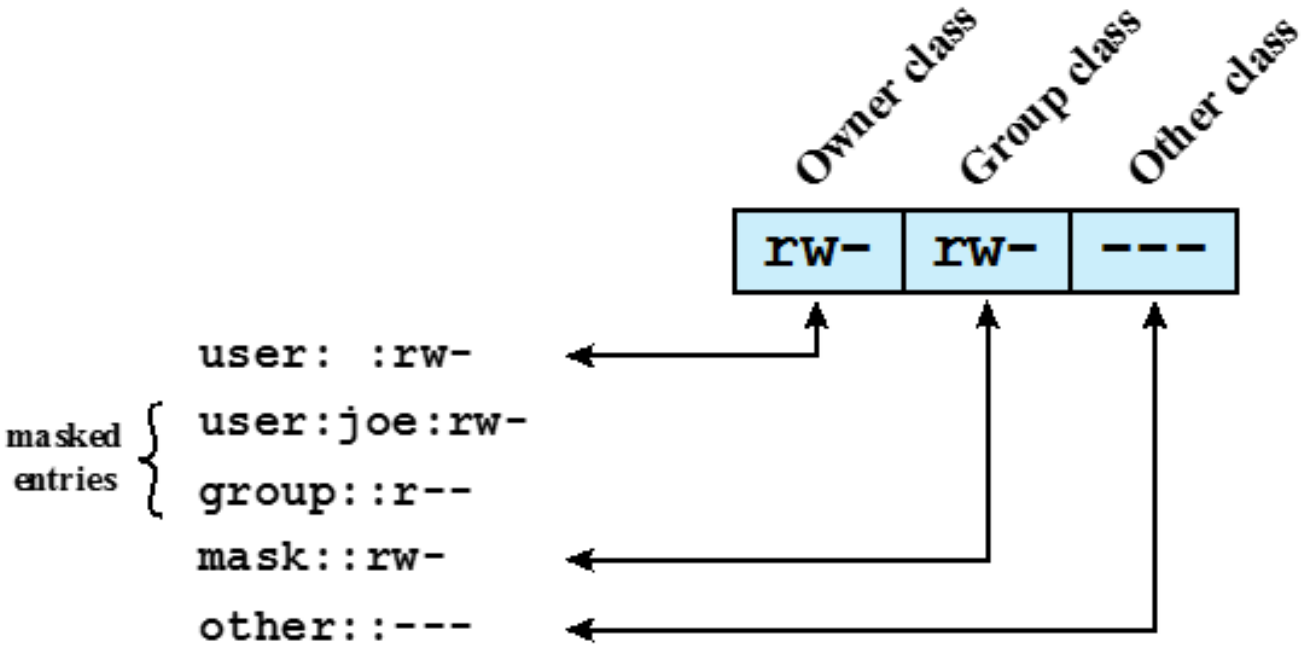
UNIX File Access Control

- "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
- When access is required
  - select most appropriate ACL
    - owner, named users, owning/named groups, others
  - check if have sufficient permissions for access

UNIX extended access control list



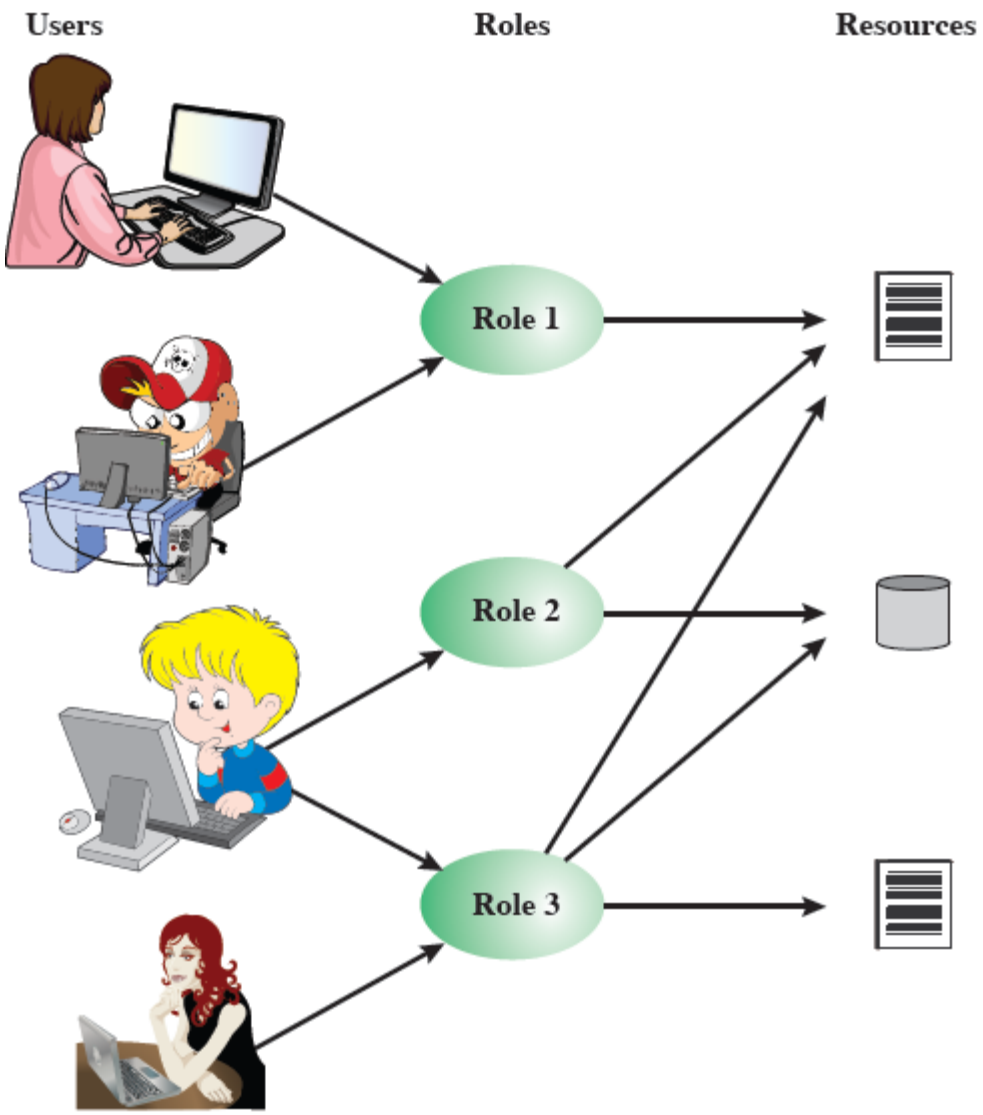
(b) Extended access control list

Role-Based Access Control

Access based on ‘role’, not identity

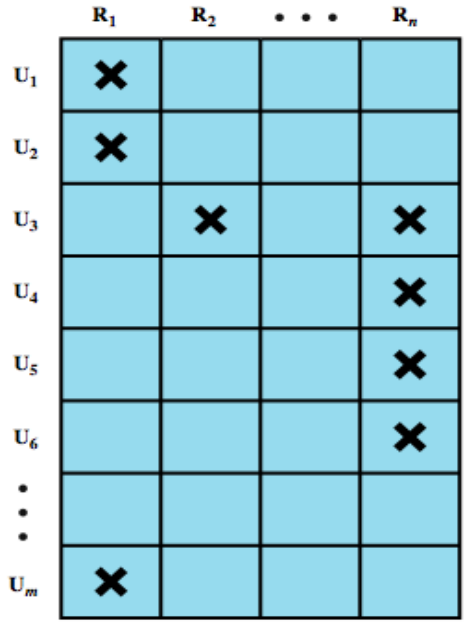
Many-to-many relationship between users and roles

Roles often static



Role-Based Access Control

Role-users and roles-object access matrix



	R_1	R_2	...	R_n
U_1	X			
U_2	X			
U_3		X		X
U_4				X
U_5				X
U_6				X
...				
U_m	X			

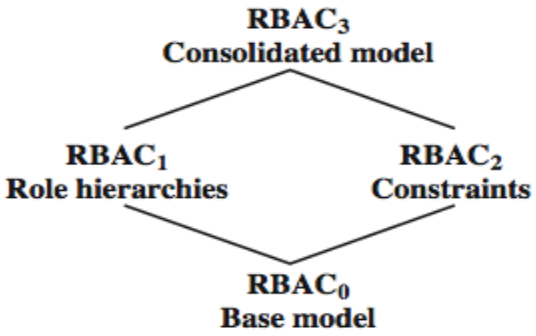
	OBJECTS								
ROLES	R_1	R_2	R_n	F_1	F_1	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R_2		control		write *	execute			owner	seek *
...									
R_n			control		write	stop			

General RBAC, Variations

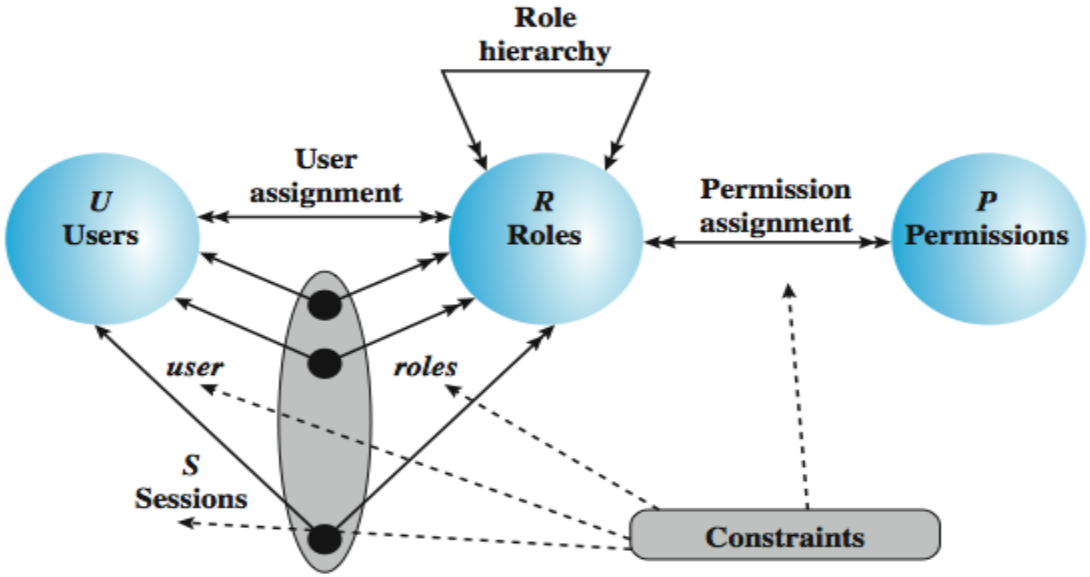
- A family of RBAC with four models
  1. RBAC0: min functionality
  2. RBAC1: RBAC0 plus role (permission) inheritance
  3. RBAC2: RBAC0 plus constraints (restrictions)
  4. RBAC3: RBAC0 plus all of the above
- RBAC0 entities
  - User: an individual (with UID) with access to system
  - Role: a named job function (tells authority level)
  - Permission: equivalent to access rights
  - Session: a mapping between a user and set of roles to which a user is assigned

Role-Based Access Control

Double arrow: ‘many’ relationship
Single arrow: ‘one’ relationship



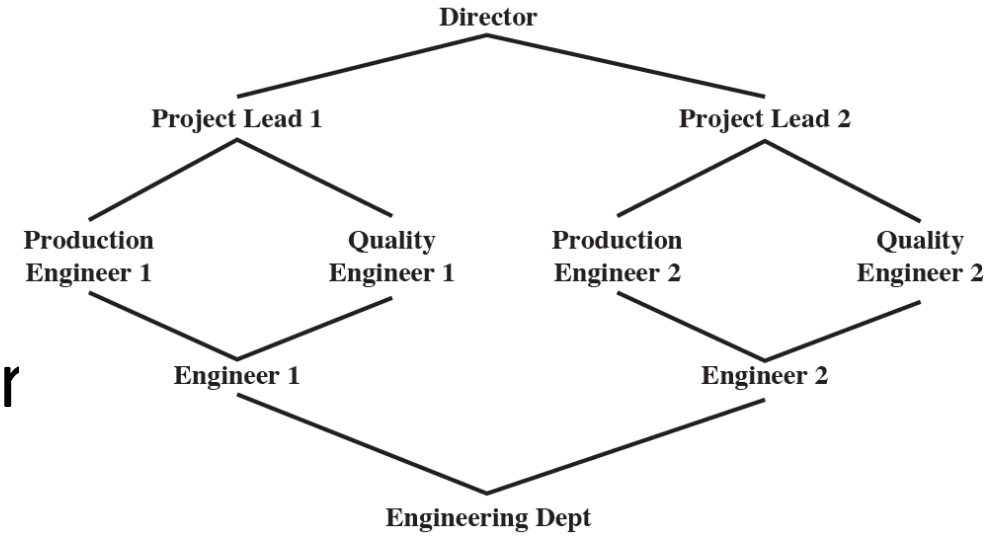
(a) Relationship among RBAC models



(b) RBAC models

Example of role hierarchy

- Director has most privileges
- Each role inherits all privileges from lower roles
- A role can inherit from multiple roles
- Additional privileges can be assigned to a role



Constraints

- A condition (restriction) on a role or between roles
  - **Mutually exclusive**
    - role sets such that a user can be assigned to only one of the role in the set
    - Any permission can be granted to only one role in the set
  - **Cardinality**: set a maximum number (of users) wrt a role (e.g., a department chair role)
  - **Prerequisite role**: a user can be assigned a role only if that user already has been assigned to some other role

Thank You !!!