Lecture 9
Access Control

Dr. Alshaimaa Abo-alian
A_alian@cis.asu.edu.eg

LECTURE OUTLINE

- What is Access Control?
- Access control context
- Access Control Models

  1. Discretionary Access Control (DAC)

     - Example: UNIX File Access Control

  2. Mandatory Access Control (MAC)
  3. Role Based Access Control (RBAC)
  4. Attribute Based Access Control (ABAC)

What is Access Control?

In Encyclopedia of Cryptography and Security (2011):

Access control is a security function that protects shared resources against unauthorized accesses.

The distinction between authorized and unauthorized accesses is made according to an *access control policy*.

What is Access Control?

Definition of RFC 4949, Internet Security Glossary:

Access control is a process by which use of system resources is:

- regulated according to a security policy, and
- permitted only by authorized entities (users, programs, processes, or other systems) according to that policy.

What is Access Control?

- We can view access control as **a central element of computer security.**
- Access control is employed to enforce security requirements (**CIA triad**)
- **The principal objectives of computer security are to:**

  1. Prevent unauthorized users from gaining access to resources
  2. Prevent authorized users from accessing resources in an unauthorized manner
  3. Enable authorized users to access resources in an authorized manner.

Which security objective is not met?

1. A journalist reading a politician’s medical record
2. A criminal performing fake bank account bookings
3. A company overloading a competitor’s computers with requests in order to prevent him from meeting a critical business deadline

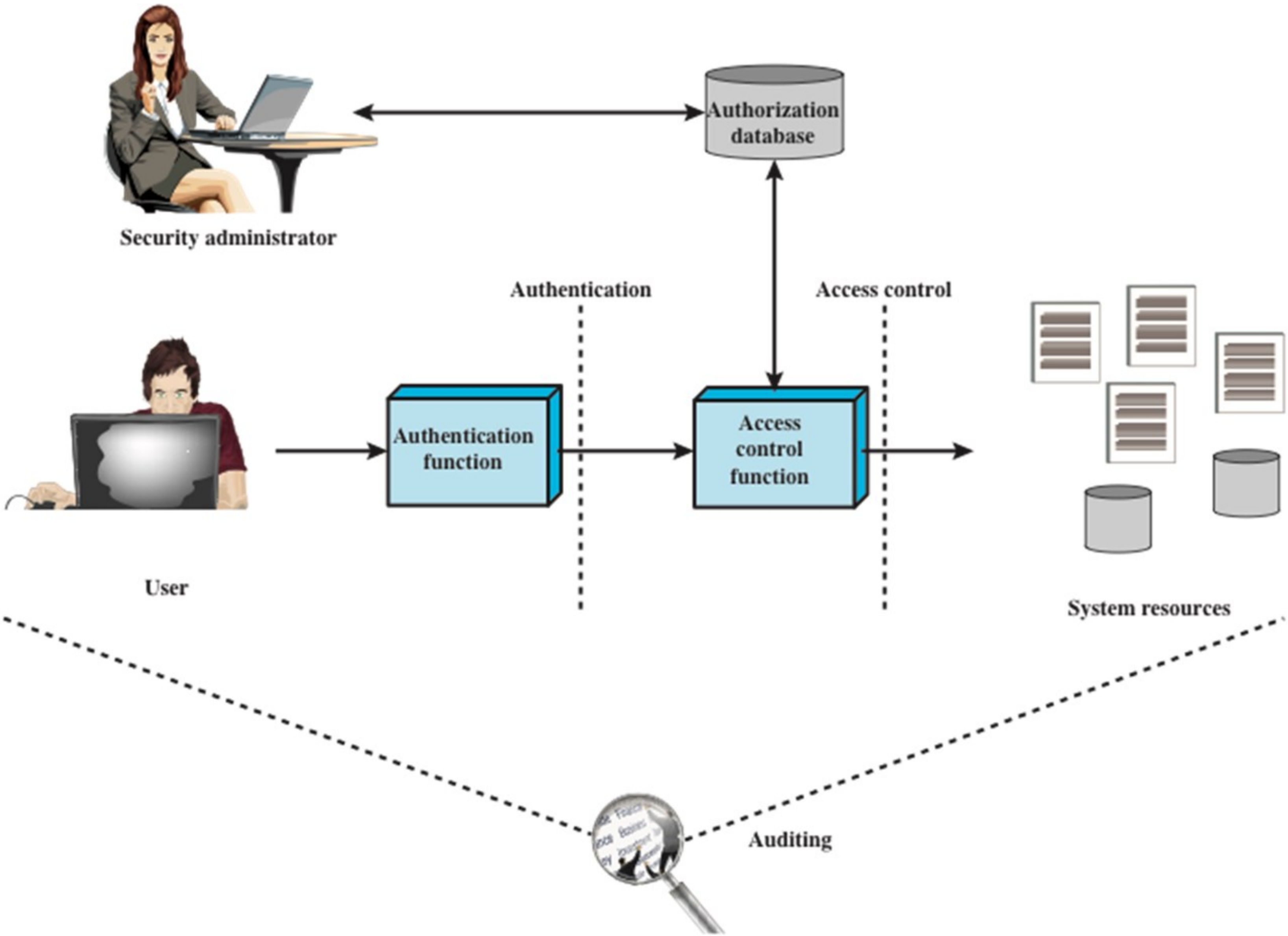


Figure 4.1 Relationship Among Access Control and Other Security Functions

Access Control Context

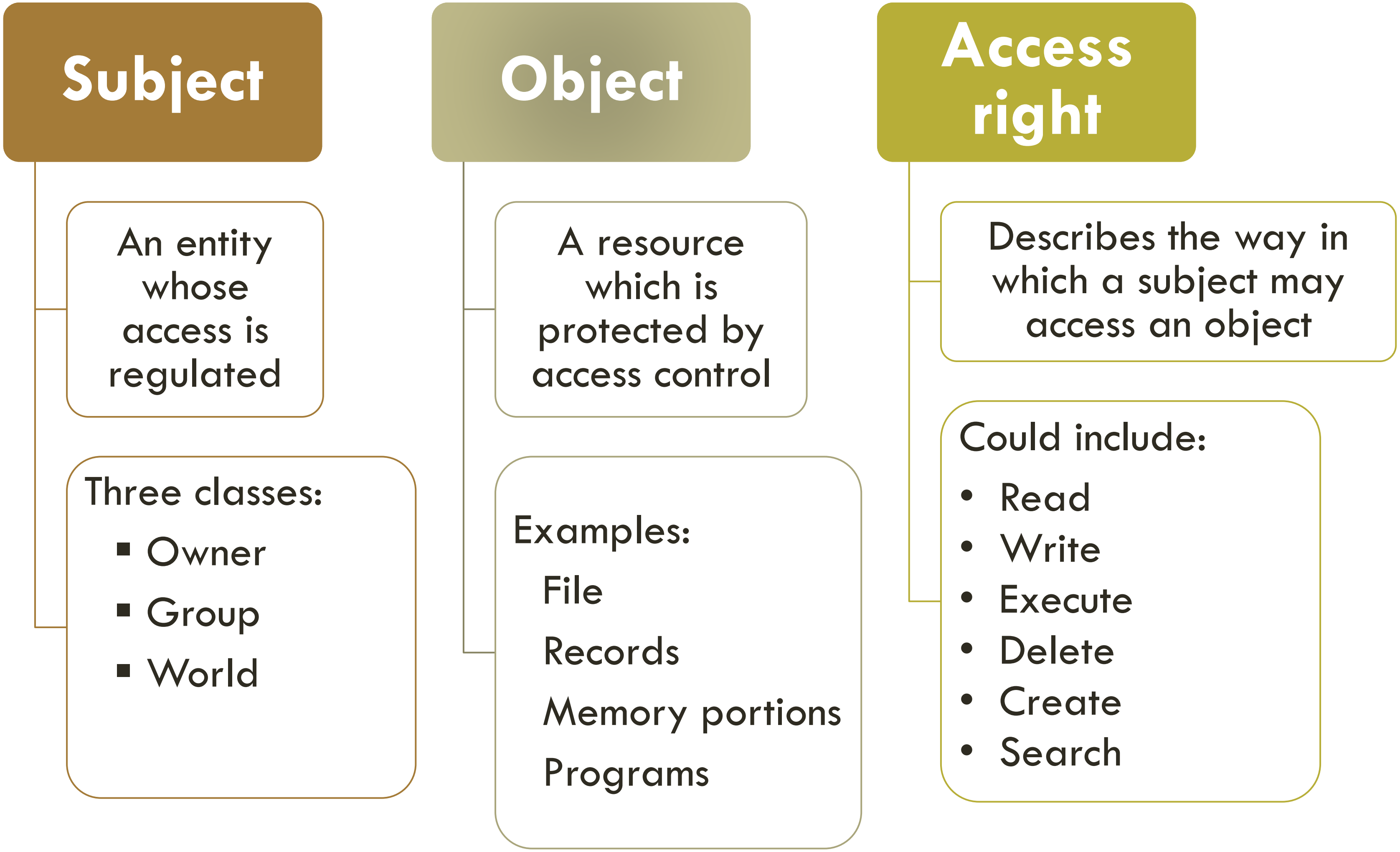
The access control context involves the following functions:

- **Authentication:** Verification that the credentials of a user are valid.
- **Authorization:** The granting/denying of a right or permission to a system entity to access a system resource.
- **Auditing :** An independent review and examination of system records and activities in order to:

1. Ensure compliance with established policy and operational procedures
2. Detect breaches in security
3. Recommend any indicated changes in control, policy and procedures.

An access control system helps to keep the wrong people out, let the right people in, and keep a log of all entries and exits.

Access Control Context



Subject

An entity whose access is regulated

Three classes:

- Owner
- Group
- World

Object

A resource which is protected by access control

Examples:

File

Records

Memory portions

Programs

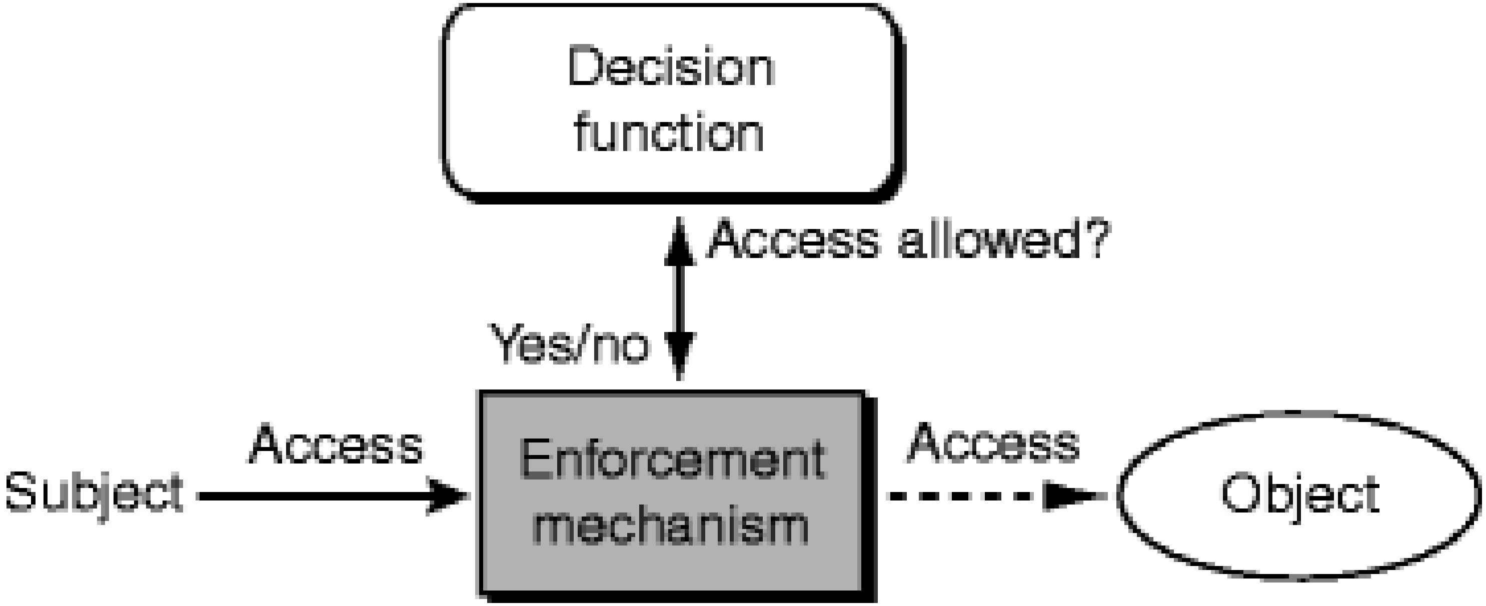
Access right

Describes the way in which a subject may access an object

Could include:

- Read
- Write
- Execute
- Delete
- Create
- Search

Access Control Context



reference monitor **property**

Access Control Models

Discretionary access control (DAC)

- Controls access **based on the identity of the subject** and on access rules (authorizations) stating what subjects are (or are not) allowed to do.

Mandatory access control (MAC)

- Controls access **based on comparing security labels** (which indicate how sensitive or critical system resources are) with security clearances

Role based access control (RBAC)

- Controls access **based on the roles** that users have within the system and on rules stating what accesses are allowed to users in given roles

Attribute based access control (ABAC)

- Controls access **based on attributes** of the user, the resource to be accessed, and current environmental conditions.

Discretionary Access Control (DAC)

- A model in which subjects (owners) can determine who has access to their objects
- Access to data objects (files, directories, etc.) is permitted based on the identity of users.
- The DAC model is implemented using four structures:

1. **Access matrix**
2. **Access control lists (ACL)**
3. **Capability list**
4. **Access control triples (Authorization Table)**

DAC Structures
Access Matrix

- An **access matrix** consists of:
  1. One dimension (**row**) represents identified **subjects** that may attempt data access to the resources
  2. The other dimension (**column**) lists the **objects** that may be accessed
  3. Each entry in the matrix indicates the access rights of a particular subject for a particular object

DAC Structures

Access Matrix

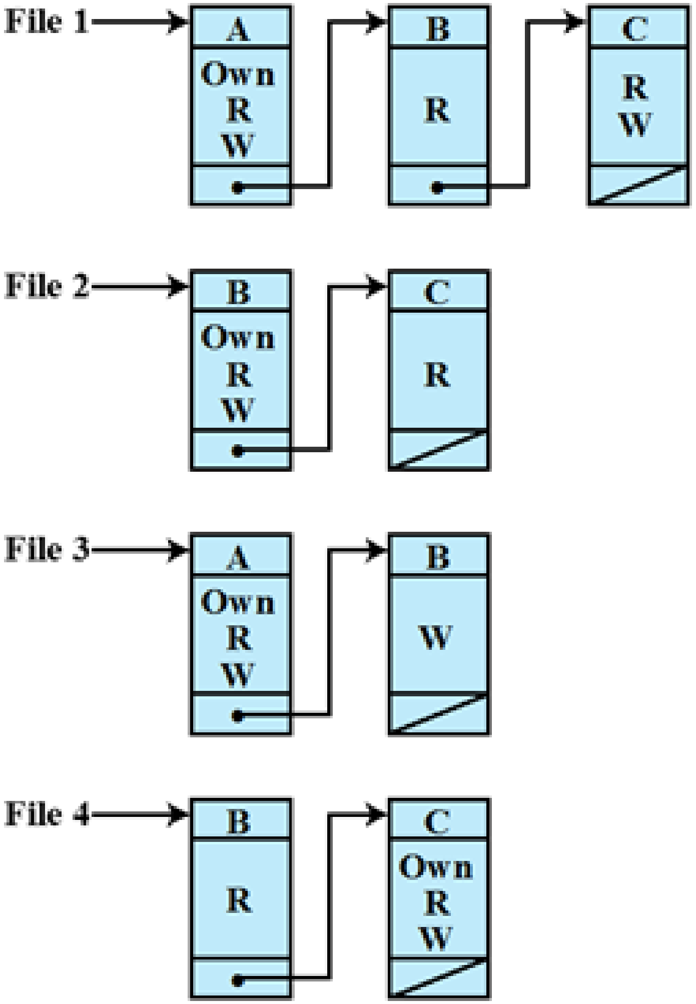
		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

(a) Access matrix

DAC Structures
Access Control List (ACL)

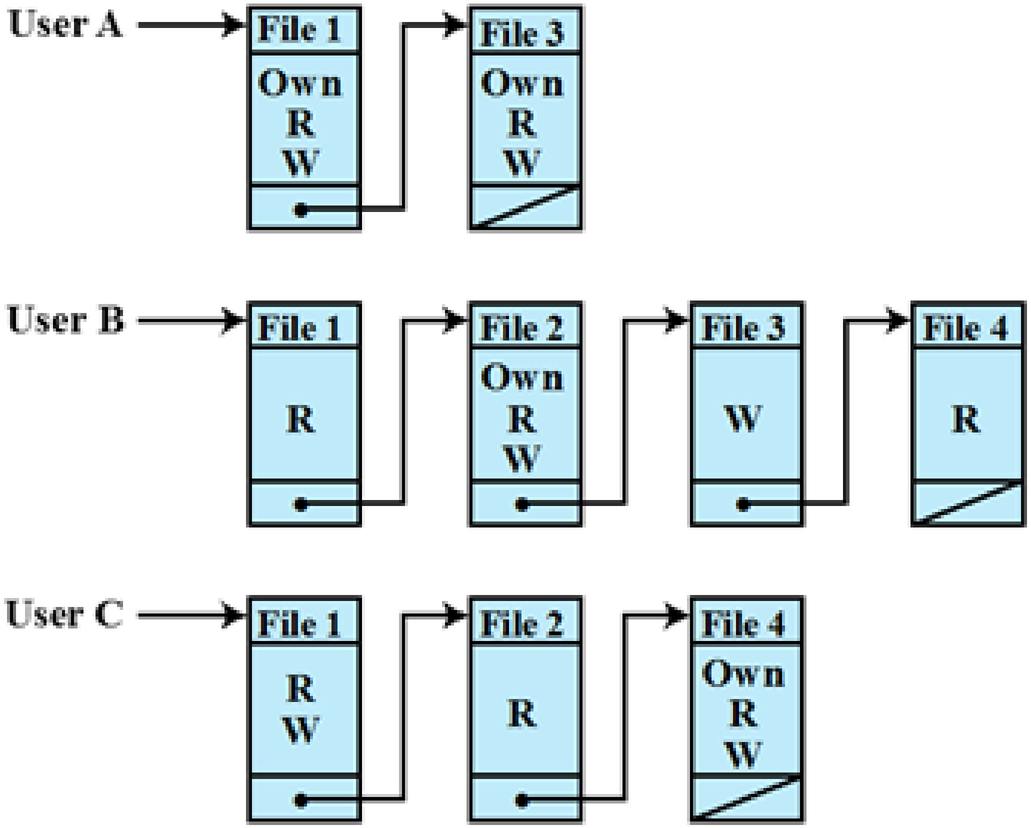
- In ACL, each object is associated with a list that indicates for each subject the access rights that the subject have on this object

  ➔ each **column of the access matrix** is stored with the object corresponding to that column



DAC Structures
Capability List

- Each subject (user) is associated with a capability list indicating its access rights (capabilities) on each object
- Each **row of the access matrix** is stored with the subject corresponding to that row



DAC Structures
Authorization Table

Subject	Access right	Object
User A	Own	File 1
User A	Read	File 1
User A	Write	File 1
User B	Read	File 1
User C	Read	File 1
User C	Write	File 1
User C	Read	File 2

Example: UNIX File Access Control

- Each user is assigned a unique user identification number (user ID)
- A user is a member of a primary group identified by a group ID
- When a file is created, it is marked with:
  - The **user ID** of its owner
  - A specific **group ID** (its owner's primary group)
  - **Sticky bit :** When applied to a directory, it specifies that only the owner of any file in the directory can rename, move, or delete that file

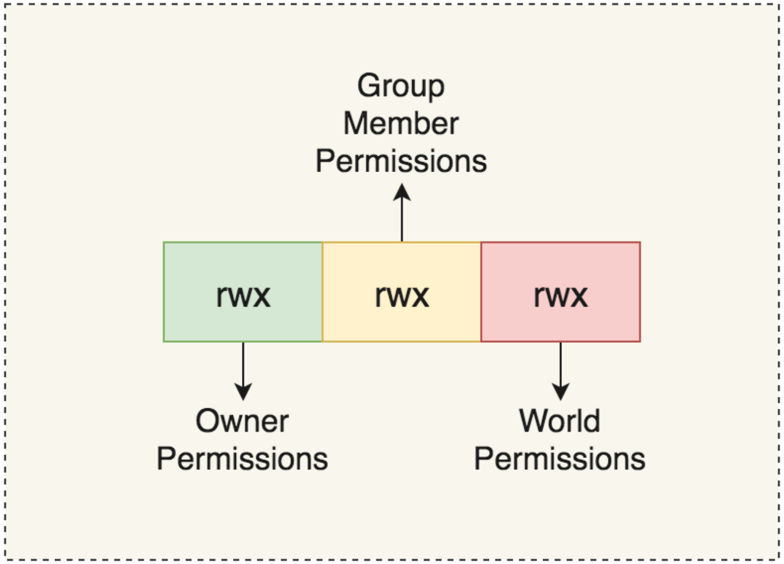
Example: UNIX File Access Control

Each UNIX file is administered using inode (index node)

- **Inode** is a control structure that contains key information needed for a particular file
- File attributes, permissions and control information are stored in the inode
- On the disk, there is an inode table (list) that contains the inodes of all the files in the file system
- When a file is opened, its inode is brought into main memory and stored in a memory resident inode table

Example: UNIX File Access Control

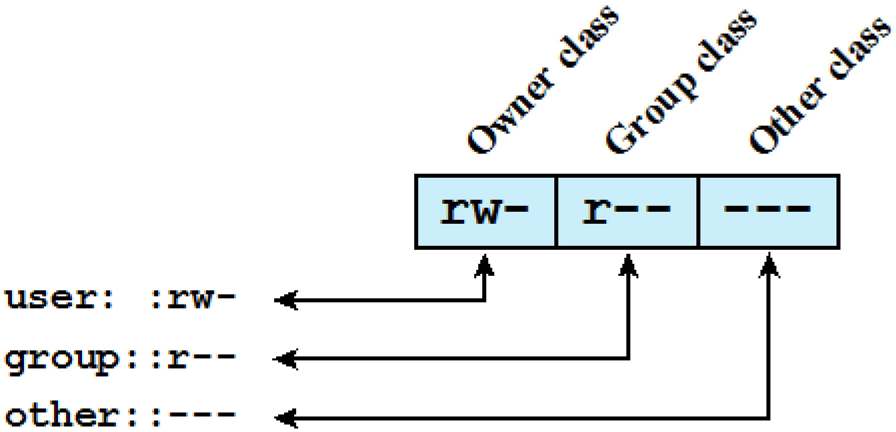
- Each file is associated with **9 protection bits** (are part of the **file's inode) to**
  - specify the 3 types of permissions which are :
    - read (**r**)
    - write (**w**)
    - execute (**x**)
  - For
    - the owner of the file
    - members of the group
    - all other users



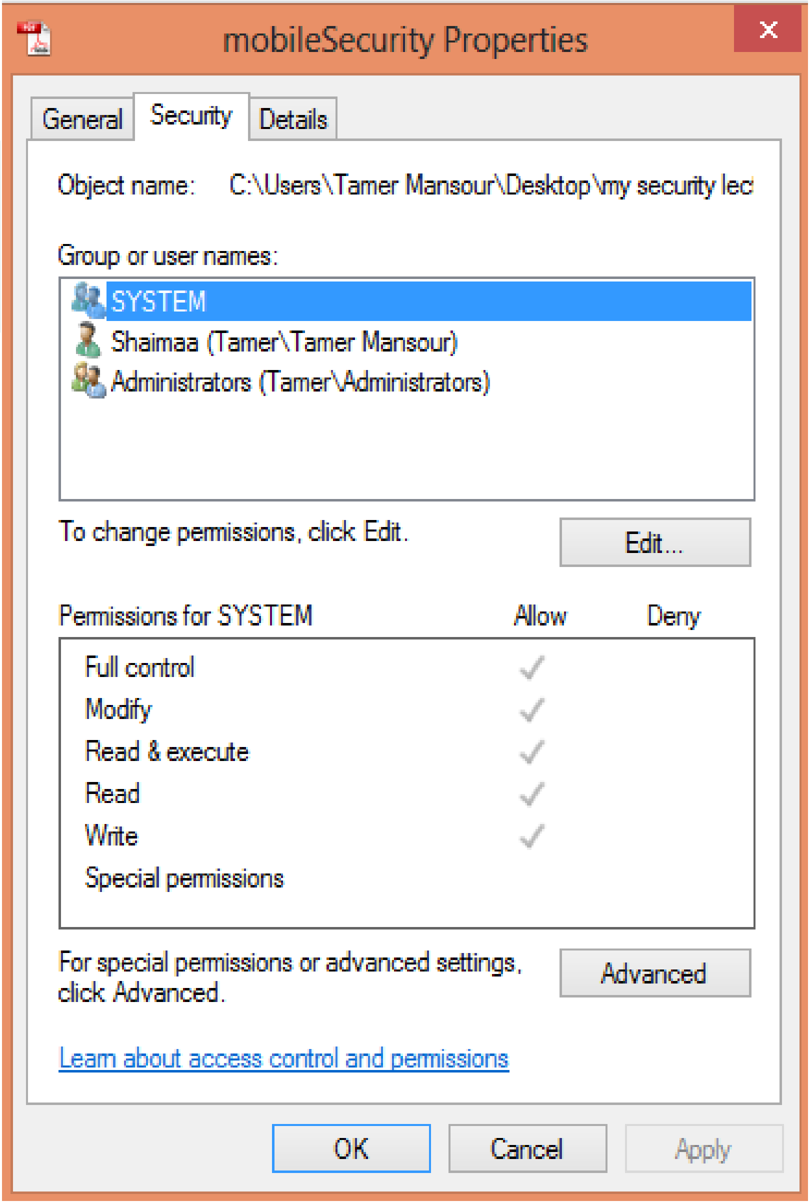
Example: UNIX File Access Control

Example:

- The file owner has read and write access
- All other members of the file’s group have read access
- Users outside the group have no access rights to the file.



(a) Traditional UNIX approach (minimal access control list)



mobileSecurity Properties

Object name: C:\Users\Tamer Mansour\Desktop\my security lec

Group or user names:

- SYSTEM
- Shaimaa (Tamer\Tamer Mansour)
- Administrators (Tamer\Administrators)

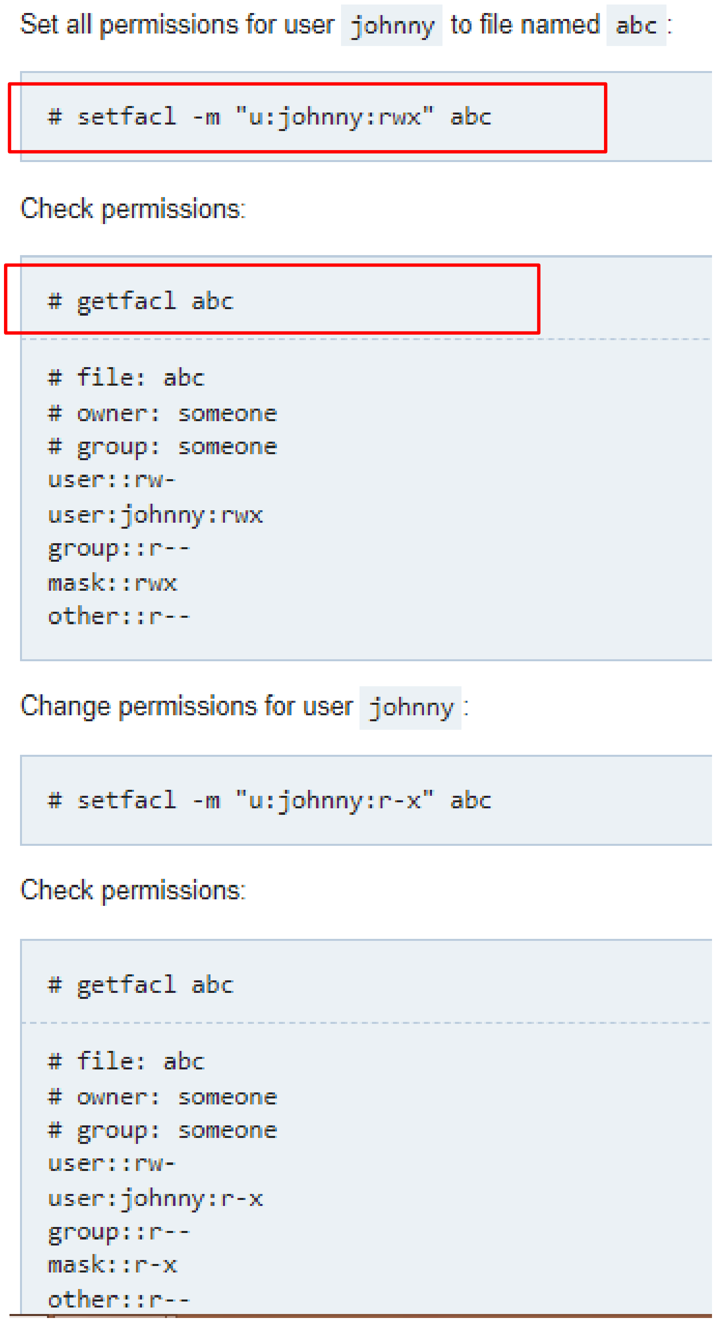
To change permissions, click Edit.

Permissions for SYSTEM	Allow	Deny
Full control	✓	
Modify	✓	
Read & execute	✓	
Read	✓	
Write	✓	
Special permissions		

For special permissions or advanced settings, click Advanced.

Learn about access control and permissions

Windows ACL UI



Set all permissions for user `johnny` to file named `abc` :

```
# setfacl -m "u:johnny:rwx" abc
```

Check permissions:

```
# getfacl abc
```

```
# file: abc
# owner: someone
# group: someone
user::rw-
user:johnny:rwx
group::r--
mask::rwx
other::r--
```

Change permissions for user `johnny` :

```
# setfacl -m "u:johnny:r-x" abc
```

Check permissions:

```
# getfacl abc
```

```
# file: abc
# owner: someone
# group: someone
user::rw-
user:johnny:r-x
group::r--
mask::r-x
other::r--
```

Example: Linux ACL Commands

Mandatory Access Control (MAC)

- Developed by U.S. Dept. of Defense in 2003 and commonly used by the government.
- Uses a hierarchical approach to control access to resources.
- Its motivation is to control the flow of information
- Prevents gaining access to protected data and transfer the data into some other objects accessible to subjects not authorized to access the protected data

Mandatory Access Control (MAC)

Bell-LaPadula (BLP) Model

- The **Bell-LaPadula (BLP)** model was developed by David E. Bell & Leonard J. LaPadula to formalize the US department of defense (DoD) **multilevel security (MLS)** policy.
- The information flows are authorized based on comparing the **security clearance** of the subjects and the **security classification** of the objects
- Each object is assigned a **security classification of** a given level that indicates the sensitivity of the object.
- Each subject is assigned a **security clearance** of a given level that indicates how much the subject can be trusted
- **Example:**

```
Top secret > secret > confidential > restricted > unclassified
```

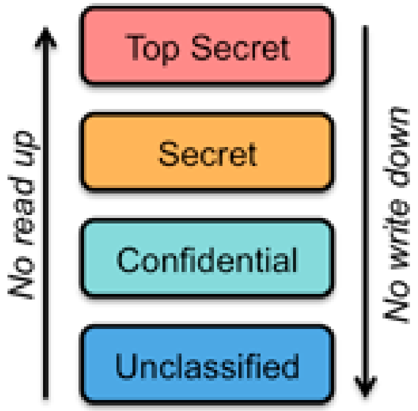
- Subject clearance and object classification are determined by security administrator.
- Users cannot overwrite the security policy.

Mandatory Access Control (MAC)

Bell-LaPadula (BLP) Model

- Required properties for confidentiality:

1. The simple security rule (No read up): Subject can only read an object of less or equal security level
2. The star *-property rule (No write down): Subject can only write an object of greater or equal security level
3. The strong star property rule:
   - Alternative to *-Property; motivated by integrity concerns.
   - Subjects can write to objects with only a matching security level



Confidential cannot read Secret
Confidential cannot write Unclassified

Write Up, Read Down (WURD)

Role Based Access Control (RBAC)

- Traditional DAC systems define the access rights of individual users
- RBAC systems assign access rights to roles instead of individual users.
- Users are assigned to different roles according to their responsibilities.

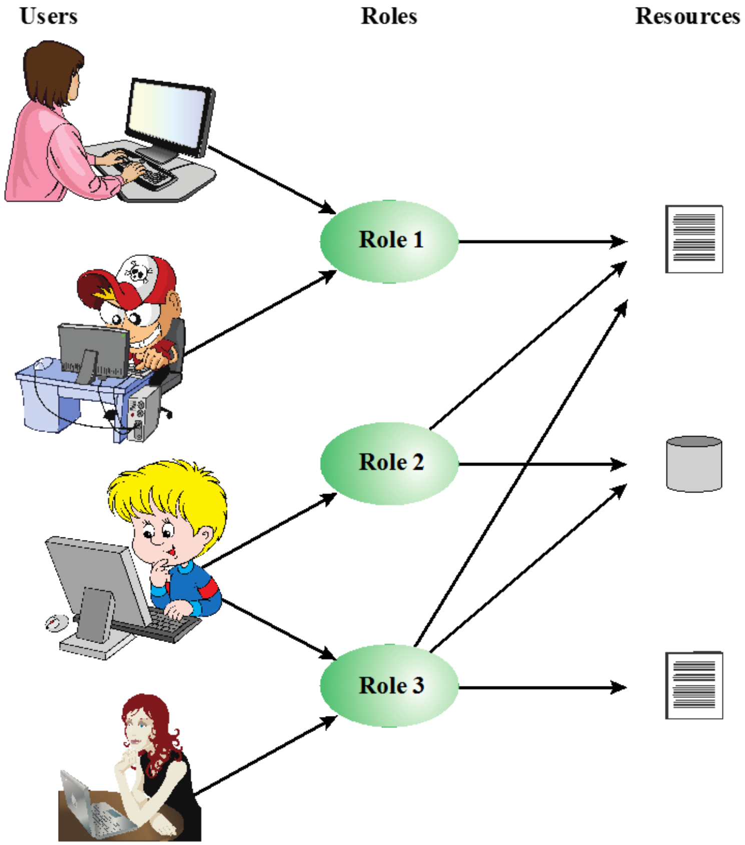
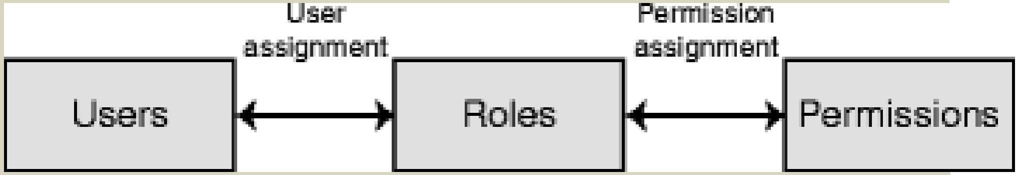


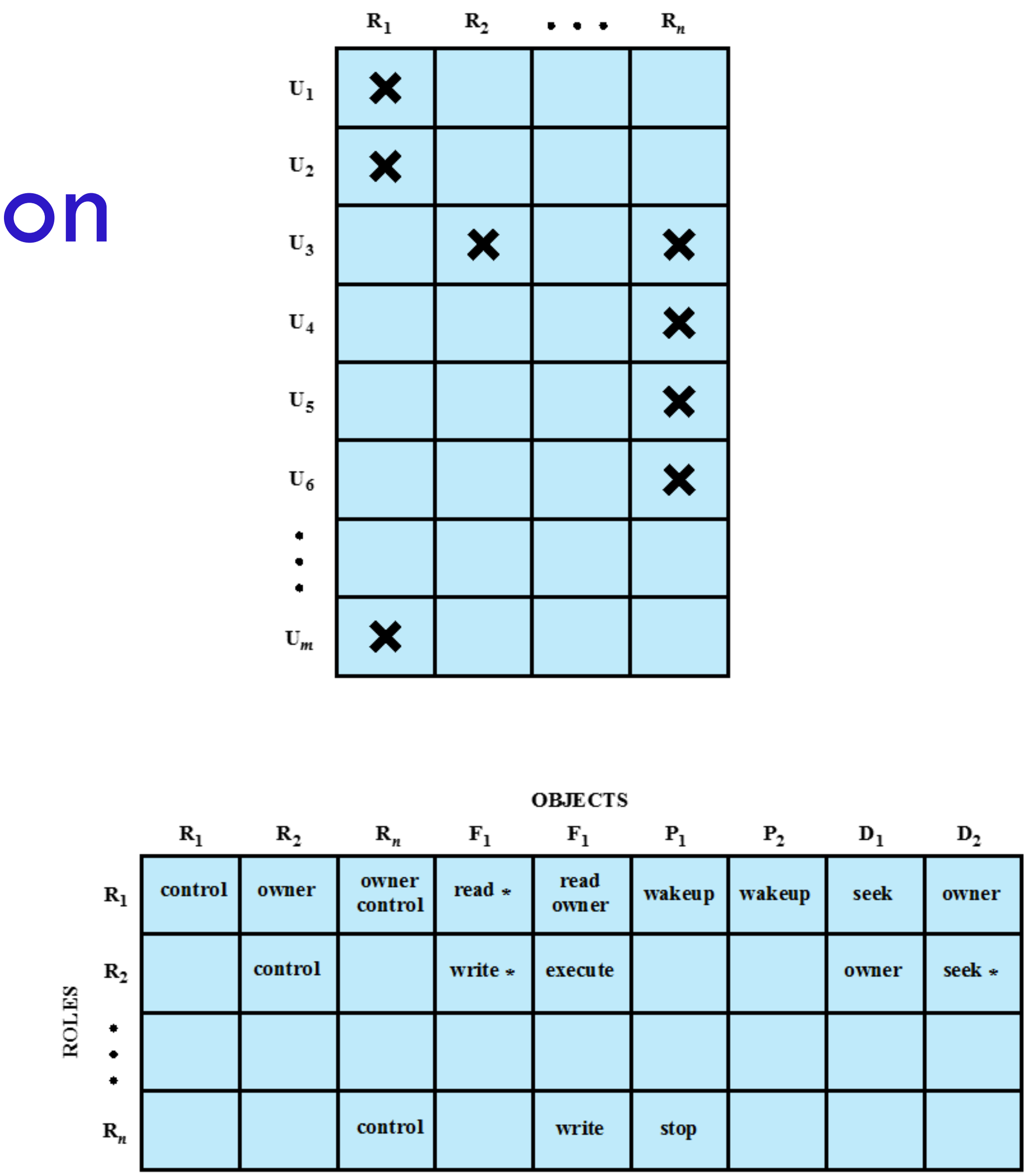
Figure 4.6 Users, Roles, and Resources



Role Based Access Control (RBAC)

- The relationship of **users to roles** and the relationship of **roles to resources** are **M:N**
- The **set of users** may change frequently
- The **assignment of a user** to one or more roles may also be dynamic.
- In most environments, The **set of roles** is relatively static, with only occasional additions or deletions.
- The **set of resources** and the specific **access rights** associated with a particular role are also likely to change infrequently

RBAC Implementation



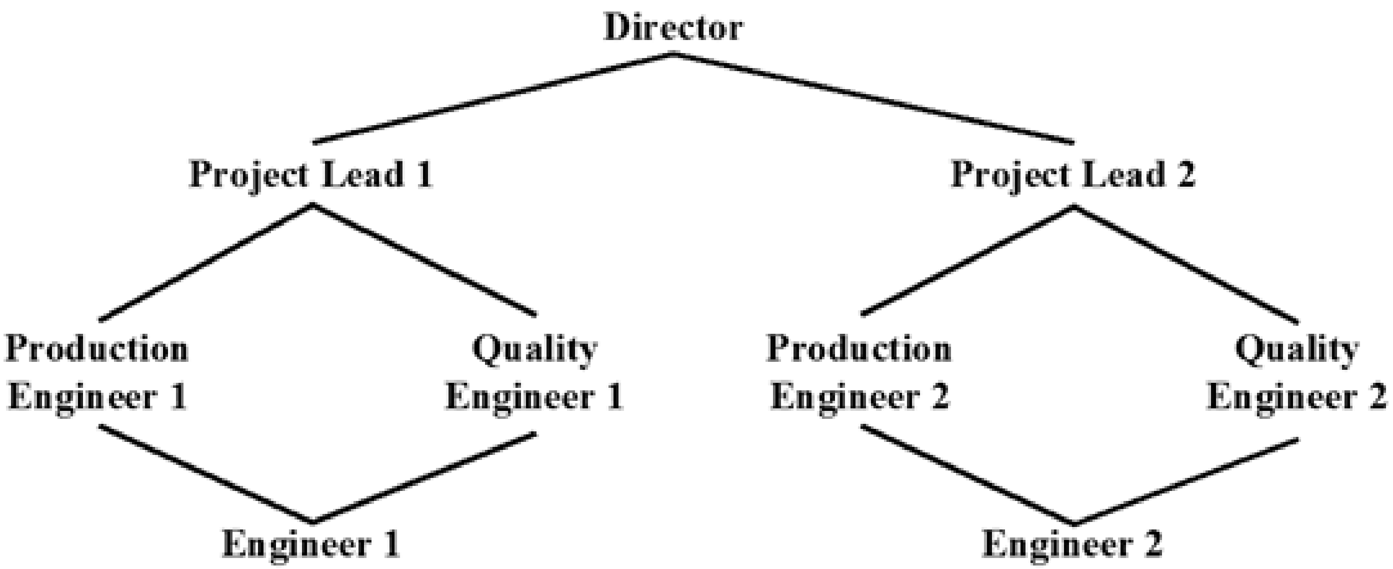
	R_1	R_2	• • •	R_n
U_1	✖			
U_2	✖			
U_3		✖		✖
U_4				✖
U_5				✖
U_6				✖
⋮				
U_m	✖			

		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			

Figure 4.7 Access Control Matrix Representation of RBAC

Hierarchical Role-based Access Control (HRBAC)

A higher role includes all access rights of lower roles (subordinates) in the hierarchy



RBAC Constraints

- Provide a means of adapting RBAC to the specifics of administrative and security policies of an organization
- A constraint is a defined relationship among roles, or a condition related to roles
- **Types:**

Mutually exclusive roles

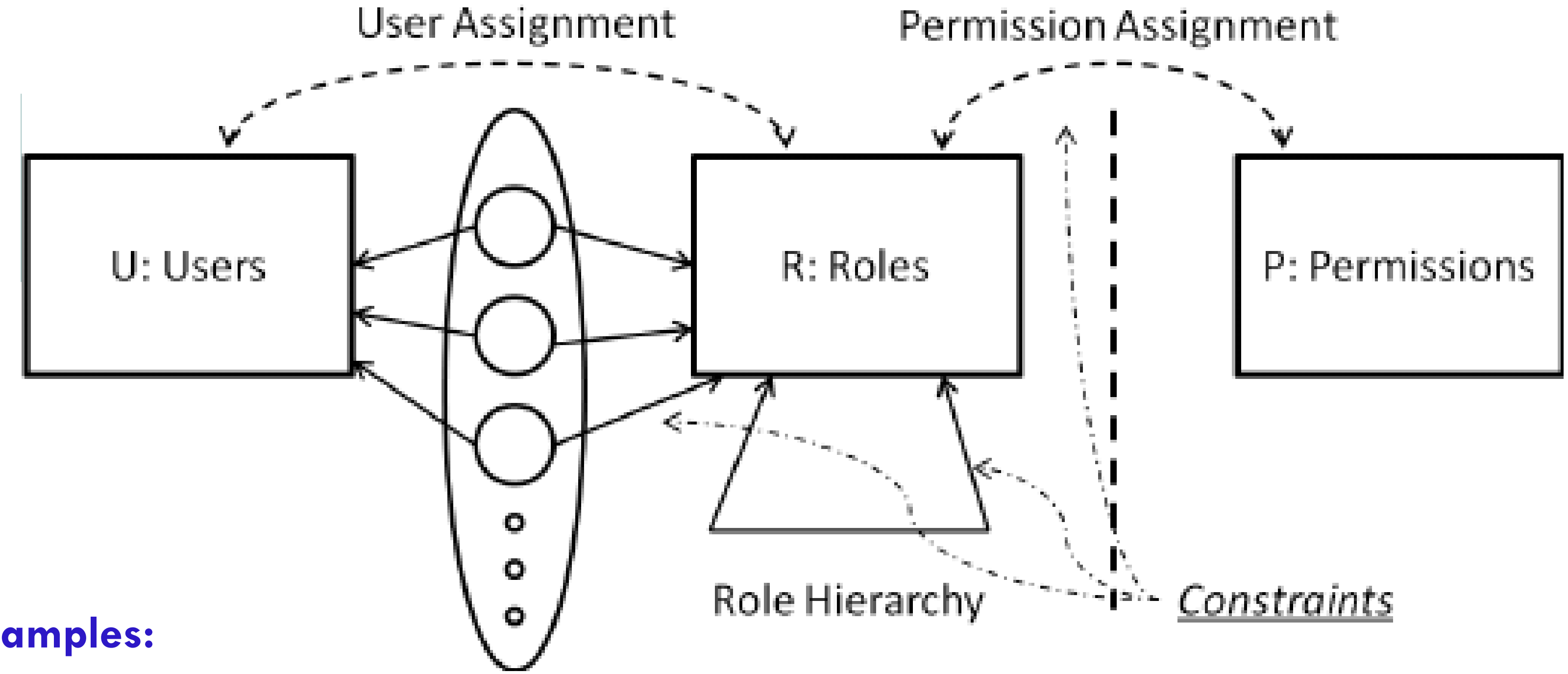
- A user can only be assigned to one role in the set

Cardinality

- Setting a max. number with respect to roles

Prerequisite roles

- a user can only be assigned to a particular role if it is already assigned to some other specified role



Examples:

A student cannot be assigned to the "Undergraduate" role and the "Postgraduate" role at the same time	**Mutually Exclusive Constraint**
A user cannot be assigned to the "Department Head" role unless s/he is assigned to the "Professor" role.	**Prerequisite Constraint**
There is only one user assigned to the "Dean" role	**Cardinality Constraint**
The max. no. of roles that can review the control sheets is three	**Cardinality Constraint**
The max. no. of roles a staff member can be assigned to is two	**Cardinality Constraint**
The "Execute Backup" permission can be assigned to either "IT manager" role or "risk management" role	**Mutually Exclusive Constraint**

Attribute Based Access Control (ABAC)

- Access policy is based on conditions on properties of both the resource and the subject.

Example:

- Attributes of user can be specialty, department, hire date, etc.
- Attributes of a medical file can be type, department, creation date, etc.
- Access policy consists of AND, OR, NOT, or threshold gates (<, >, ≤, ≥, …).

- The strength of the ABAC approach is its flexibility and expressive power.
- the main obstacle: concern about the performance impact of evaluating predicates on both resource and user properties for each access.

ABAC Model: Attributes

Subject attributes

Object attributes

Environment attributes

- A subject can be a user, an application, or a device
- Attributes define the identity and characteristics of the subject

➔ e.g., name, organization, job title, security clearance and so on.

- An object (or resource) can be files, records, tables, printers, etc.

➔ A Word document may have attributes such as title, subject, date, and author. (Meta data)

- Describe the operational, technical, and even situational environment or context in which the information access occurs

➔ e.g.: current date and time, device and the network's security level (e.g., Internet vs. intranet)

- Ignored in most access control policies

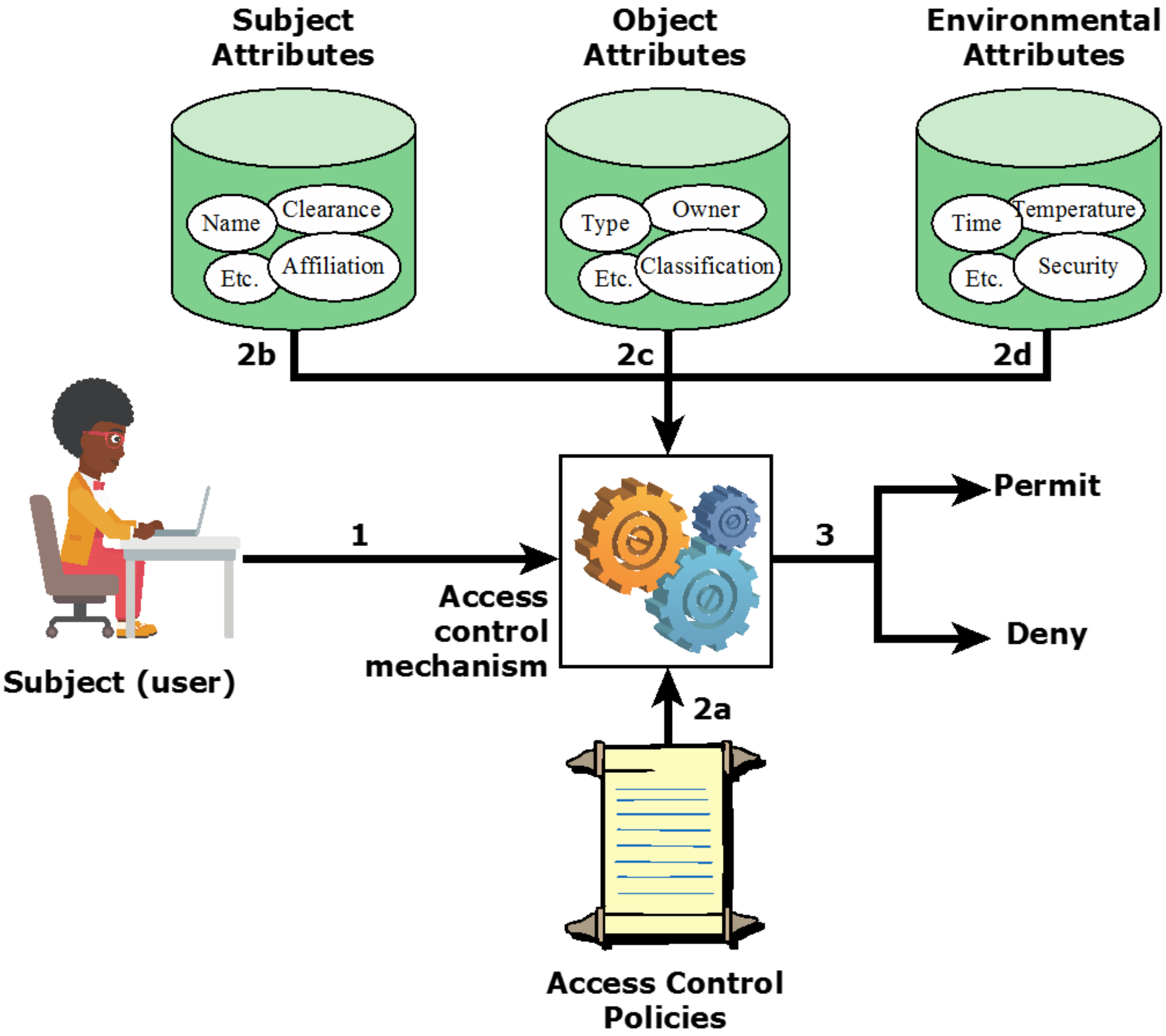


Figure 4.10 ABAC Scenario

ABAC Policy Model

To define an ABAC policy model, The following conventions are used:

- S, O, and E are subjects, objects, and environments, respectively.
- ATTR(*s*), ATTR(o), and ATTR(e) are attribute assignment relations for subject *s*, object o, and environment e, respectively
- **Example** :
  - Grade (s) = 4
  - Category (o) = ‘Practical’
  - Location (e) = ‘on Campus’

Example

An online entertainment store that streams movies to users for a monthly fee.

The value assignment of individual attributes can be:

Role(S) = *“Service Consumer”*

Age(S) = *17*

Rating(O) = *“R”*

CurrentDate(e) = *“01-10-2022”*

The access control policy:

Movie Rating	Users Allowed access
R	Age ≥ 17
PG-13	Age ≥ 13
G	Everyone

Example

The policy rule can be defined as:

$R1: can\_access(S, O, e) \leftarrow$
$(Age(S) \geq 17 \wedge \quad Rating(O)$ {R, PG-13, G }) ∨
$(Age(S) \geq 13 \wedge Age\ (S) < 17 \wedge \quad Rating(O)$ {PG-13, G }) ∨
$(Age(S) < 13 \wedge \quad Rating(O)$ {G })

Another access policy can be "users with premium membership can access all movies whereas regular user can access only old-release movies "

$R2: can\_access(S, O, e) \leftarrow$
$(MembershipType(S)\ =\ Premium$)) ∨
$(MembershipType(S)\ =\ Regular \wedge \quad MovieType(O)$
$=\ OldRelease)$

Lecture References

1. “**Computer Security: Principles and Practice**”, 4/e, by William Stallings and Lawrie Brown

Chapter 4 “Access Control”.

2. Brose, G. (2011). Access Control. In: **Encyclopedia of Cryptography and Security.** Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-5906-5_179

Thank you