OntarioTech Business & IT

Access Control

By

Dr. Madani

Winter 2023

Learning Objectives

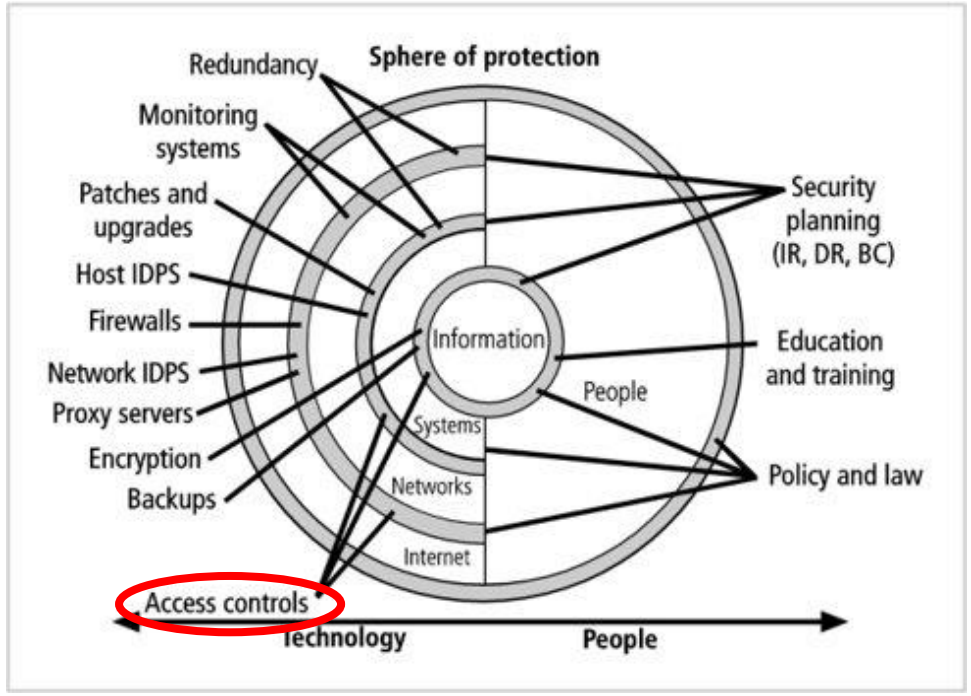
Upon completion of this material, you should be able to

- Discuss three main processes/stages encompassing access control.
- Distinguish between the major categories of access control policies.

Access Control - Principles

NISTIR 7298 defines access control as:

- "the process of granting or denying specific requests to: (1) obtain and use information and related information processing services; and (2) enter specific physical facilities"

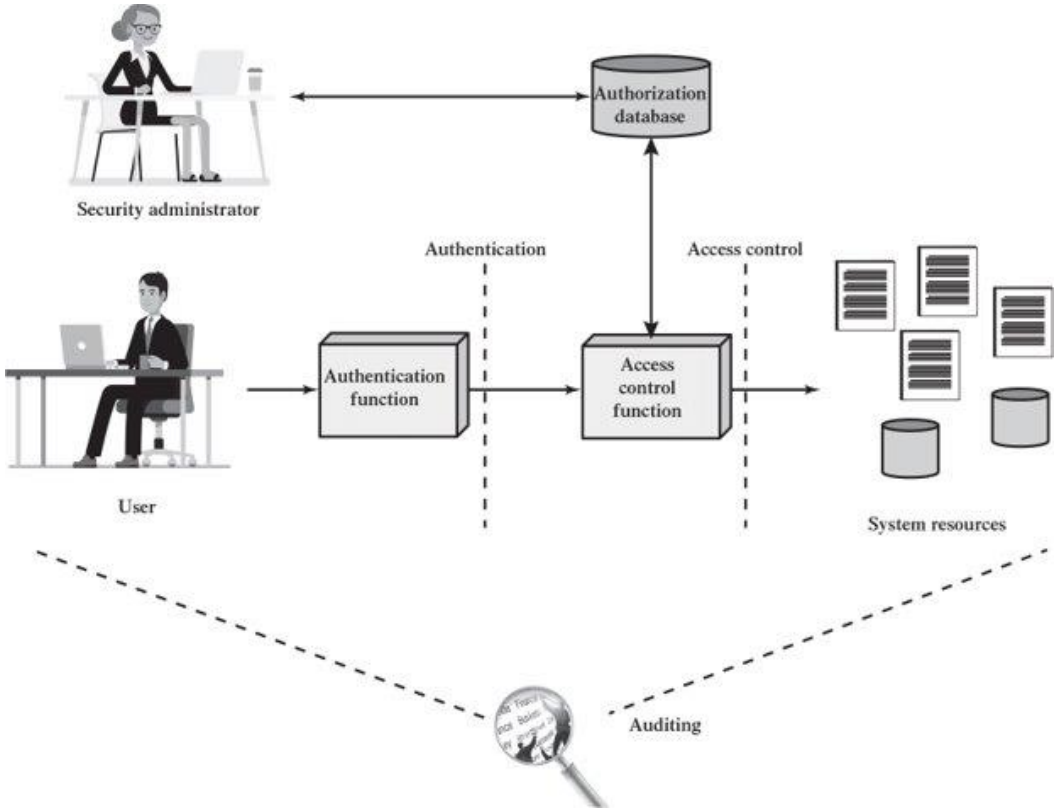


OntarioTech Business & IT

Access Control - Principles

Stages of Access Controls

1. Identification
   - Obtain identity of an entity requesting access to a logical or physical area.
2. Authentication
   - Confirm identity of the entity seeking access.
3. Authorization
   - Determine whether the authenticated entity is permitted! To access a particular system and its resources (e.g., system's files)



Access Control - Principles

- Example of stages

Action	Description	Scenario Example	Computer Process
Identification	Review of credentials	Delivery person shows employee badge	User enters username
Authentication	Validate credentials as genuine	Megan reads badge to determine it is real	User provides password
Authorization	Permission granted for admittance	Megan opens door to allow delivery person in	User authorized to log in
Access	Right given to access specific resources	Delivery person can only retrieve box by door	User allowed to access only specific data

Table 7-1 Basic steps in access control

Access Control - Principles

Important Notes

- Just because a user can authenticate to a system, it does not mean they are given access to anything and everything.
- Authorization ensues that the requested object or activity on an object is possible based on the privileges assigned to the subject.

Object vs Subject

Object

- is a resource to which access is controlled (e.g., records, blocks, pages, segments, files, portions of files).
- Some access control systems also encompass, bits, bytes, words, processors, communication ports, clocks, and network nodes.

Subject

- is an entity capable of accessing objects.
- Basic access control systems typically define three classes of subject
  - **Owner**: This may be the creator of a resource, such as a file.
  - **Group**: In addition to the privileges assigned to an owner, a named group of users may also be granted access rights, such that membership in the group is sufficient to exercise these access rights. In most schemes, a user may belong to multiple groups.
  - **World**: The least amount of access is granted to users who are able to access the system but are not included in the categories owner and group for this resource.

Access Rights

Access right: describes the way in which a subject may access an object. Access rights could include the following:

- **Read**: User may view information in a system resource (e.g., a file, selected records in a file, selected fields within a record, or some combination). Read access includes the ability to copy or print.

- **Write**: User may add, modify, or delete data in system resource (e.g., files, records, programs). Write access includes read access.

- **Execute**: User may execute specified programs.

- **Delete**: User may delete certain system resources, such as files or records.

- **Create**: User may create new files, records, or fields.

- **Search**: User may list the files in a directory or otherwise search the directory/

Access Control Policies

An access control policy, which is embodied in an authorization database, dictates what types of access are permitted, under what circumstances, and by whom. Access control policies are generally grouped into the following access control categories:

- **Mandatory access control (MAC)**: based on comparing security labels (which indicate how sensitive or critical system resources are) with security clearances (which indicate system entities are eligible to access certain resources). This policy is termed mandatory because an entity that has clearance to access a resource may not, just by its own volition, enable another entity to access that resource.
- **Discretionary access control (DAC)**: based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do. This policy is termed discretionary because an entity might have access rights that permit the entity, by its own volition, to enable another entity to access some resource.
- **Role-based access control (RBAC)**: 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.

These four policies are not mutually exclusive. Can employ two or even all three of these policies to cover different classes of system resources.

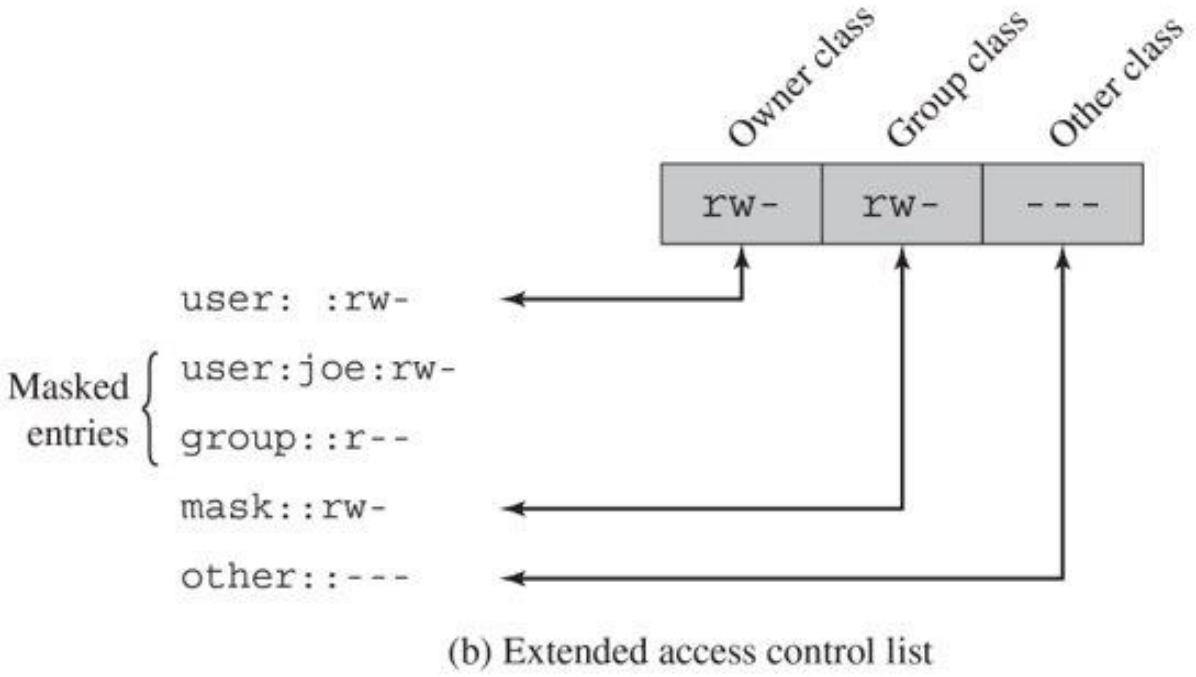
Discretionary Access Control (DAC)

- A discretionary access control scheme is one in which an entity may be granted access rights that permit the entity, by its own volition, to enable another entity to access some resource.
- Often provided using an access matrix
  - One dimension consists of identified subjects that may attempt data access to the resources
  - The other dimension lists the objects that may be accessed

		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

Discretionary Access Control (DAC)

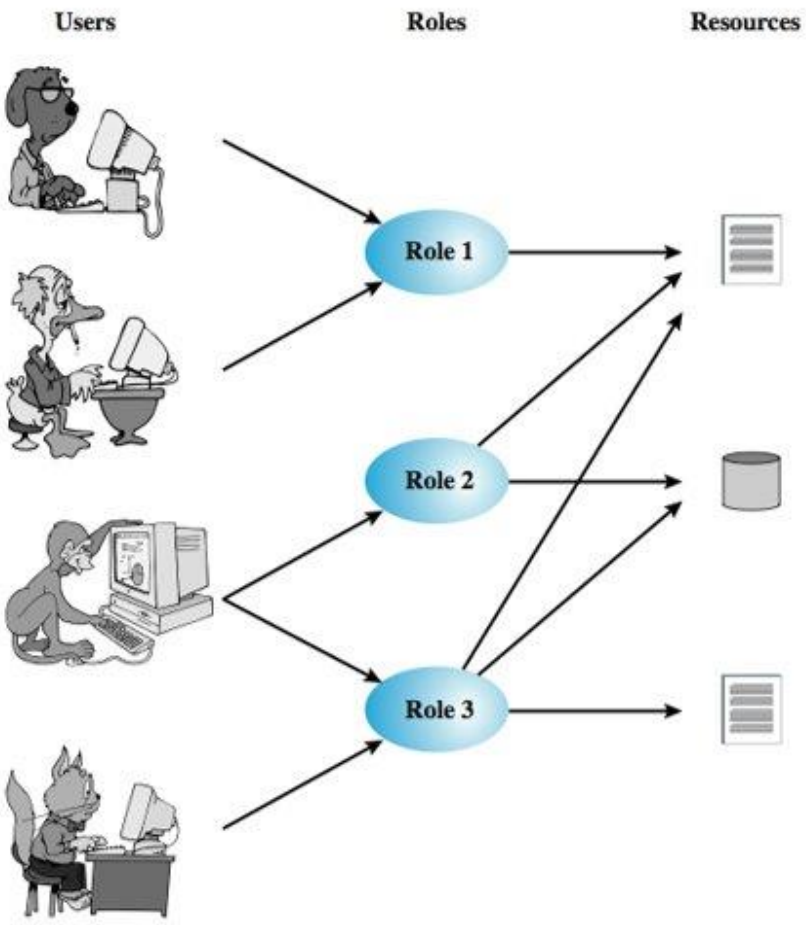


(b) Extended access control list

Figure 4.5 **UNIX File Access Control**

Role-based Access Control (RBAC)

- Traditional DAC systems define the access rights of individual users and groups of users. In contrast, RBAC is based on the roles that users assume in a system rather than the user's identity.
- Typically, RBAC models define a role as a job function within an organization.
- RBAC systems assign access rights to roles instead of individual users. In turn, users are assigned to different roles, either statically or dynamically, according to their responsibilities.
- RBAC now enjoys widespread commercial use and remains an area of active research.



Role-based Access Control (RBAC)

- Base model – Access Matrix Representation

(1) Create roles and assign objects access rights to each role!

ROLES \ OBJECTS	R_1	R_2	R_n	F_1	F_2	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			

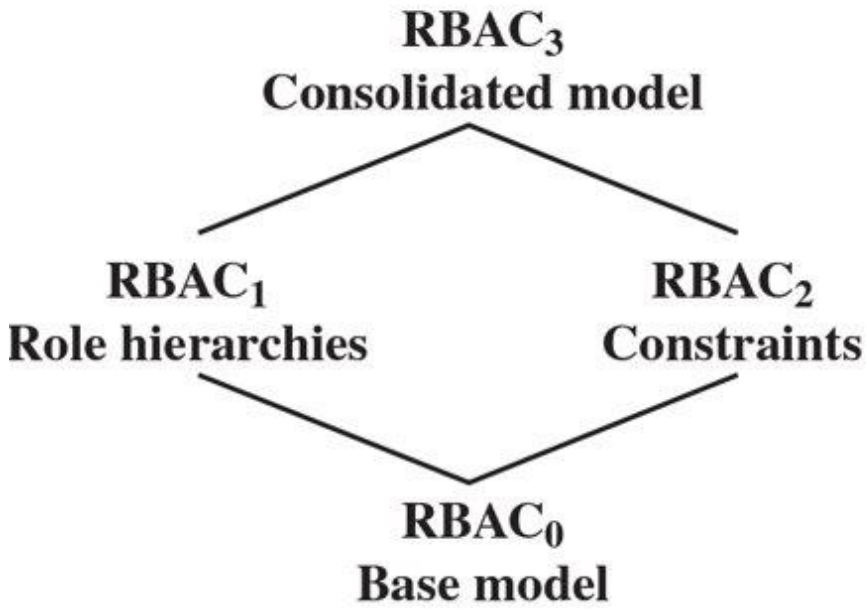
(2) Assign users to the created roles!

	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			

OntarioTech Business & IT

Role-based Access Control (RBAC)

- RBAC Version 1:
  - Role hierarchies provide a means of reflecting the hierarchical structure of roles in an organization. Typically, job functions with greater responsibility have greater authority to access resources. A subordinate job function may have a subset of the access rights of the superior job function. Role hierarchies make use of the concept of inheritance to enable one role to implicitly include access rights associated with a subordinate role.



(a) Relationship among RBAC models

Role-based Access Control (RBAC)

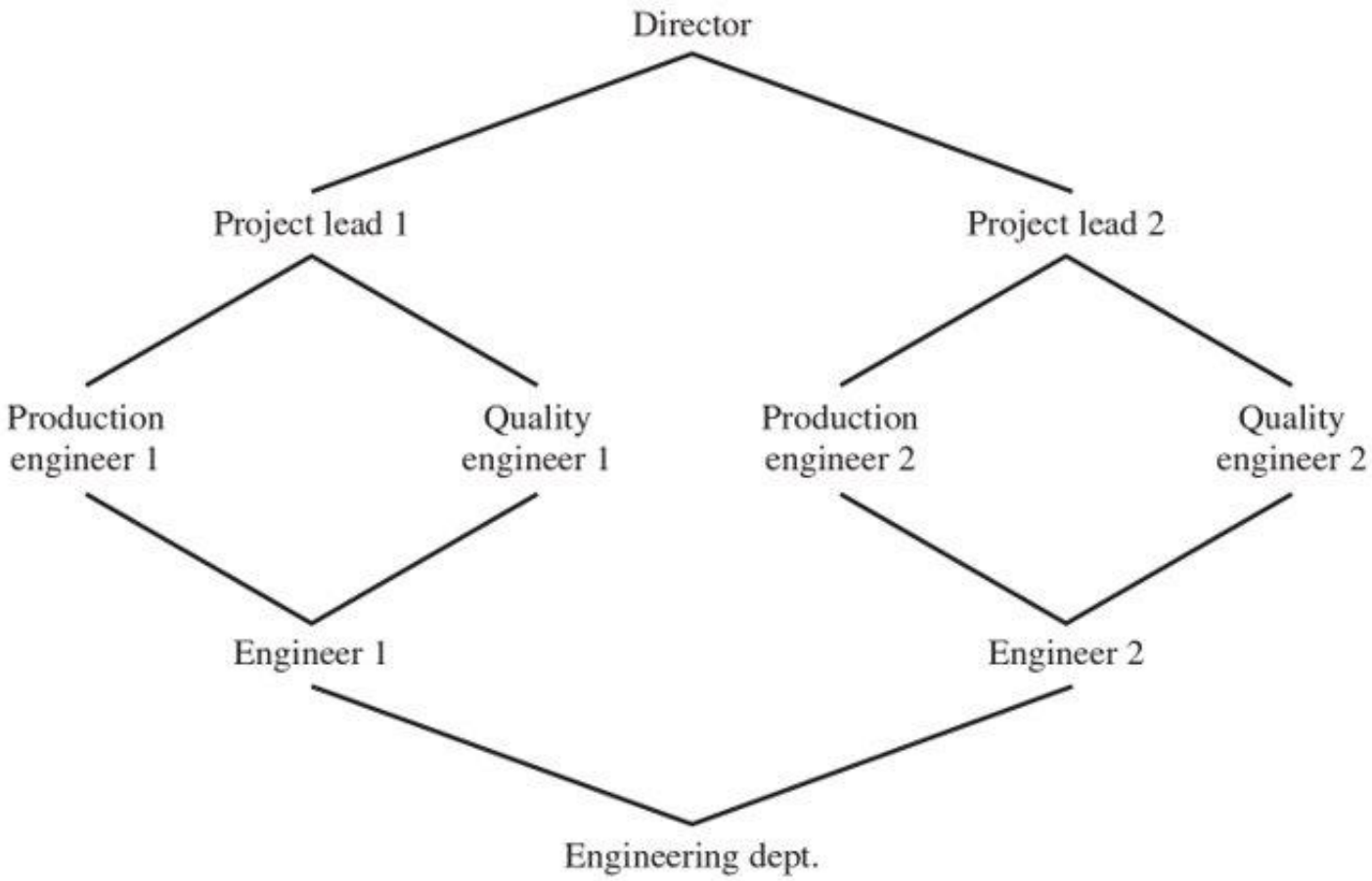


Figure 4.9 **Example of Role Hierarchy**

Role-based Access Control (RBAC)

- RBAC Version 2 – Constraints:
  - **Mutually exclusive** roles are roles such that a user can be assigned to only one role in the set. This limitation could be a static one, or it could be dynamic, in the sense that a user could be assigned only one of the roles in the set for a session. The mutually exclusive constraint supports a separation of duties and capabilities within an organization.
  - **Cardinality** refers to setting a maximum number with respect to roles. One such constraint is to set a maximum number of users that can be assigned to a given role. For example, a project leader role or a department head role might be limited to a single user.
  - A system might be able to specify a **prerequisite** role, which dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role. A prerequisite can be used to structure the implementation of the least privilege concept. In a hierarchy, it might be required that a user can be assigned to a senior (higher) role only if it is already assigned an immediately junior (lower) role.

Attribute-based Access Control (ABAC)

- Relatively new compared to other access control polices.
- Can define authorizations that express conditions on properties of both resource (object), subject and environmental conditions.

- Subject attributes:
  - Each subject has associated attributes that define the identity and characteristics of the subject. Such attributes may include the subject's identifier, name, organization, job title, and so on. A subject's role can also be viewed as an attribute.

- Object attributes:
  - An object, also referred to as a resource, is a passive information system-related entity (e.g., devices, files, records, tables, processes, programs, networks, domains) containing or receiving information. A Microsoft Word document, for example, may have attributes such as title, subject, date, and author. Object attributes can often be extracted from the metadata of the object.

- Environment attributes
  - They describe the operational, technical, and even situational environment or context in which the information access occurs. For example, attributes, such as current date and time, the current virus/hacker activities, and the network's security level (e.g., Internet vs. intranet), are not associated with a particular subject nor a resource

Attribute-based Access Control (ABAC)

- An access by a subject to an object proceeds according to the following steps:

1. A subject requests access to an object.

2. The access control mechanism is governed by a set of rules (2a) that are defined by a preconfigured access control policy. Based on these rules, the access control mechanism assesses the attributes of the subject (2b), object (2c), and current environmental conditions (2d) to determine authorization.

3. The access control mechanism grants the subject access to the object if access is authorized and denies access if it is not authorized.

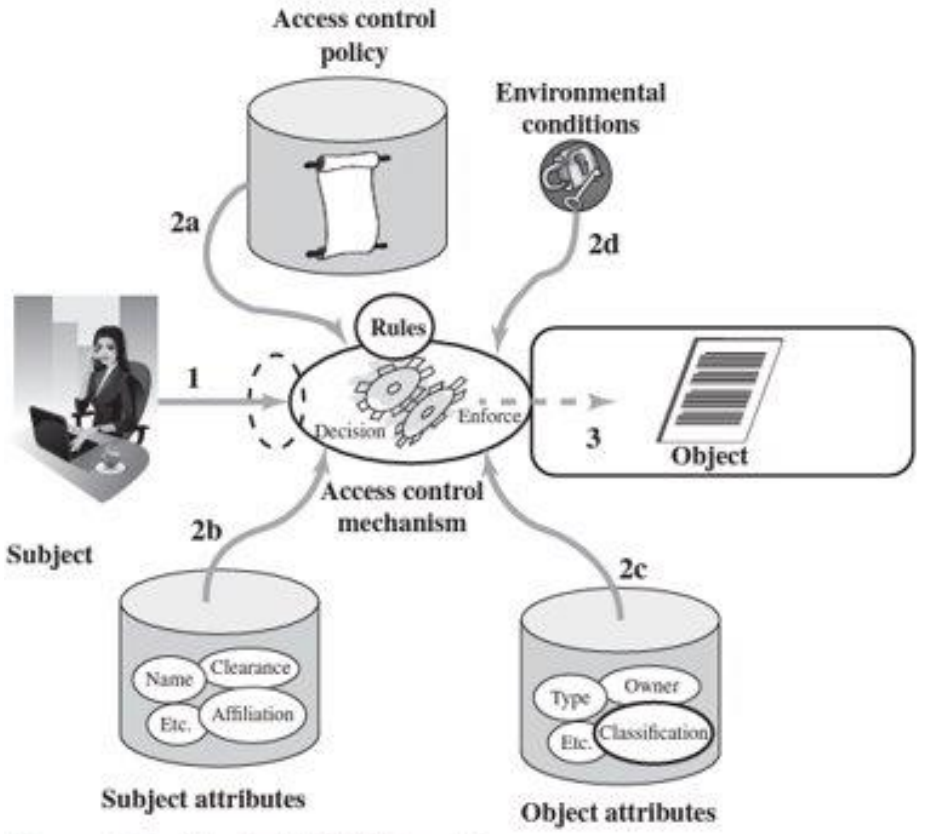


Figure 4.10 **Simple ABAC Scenario**

Attribute-based Access Control (ABAC)

- ABAC Policy
  - A policy is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions.

  - ABAC Privileges represent the authorized behavior of a subject; they are defined by an authority and embodied in a policy.

- In the most general form, a Policy Rule, which decides on whether a subject s can access an object o in a particular environment e, is a Boolean function of the attributes of `s, o,` and `e`:

  - `Rule: can_access (s, o, e) ← f(ATTR(s), ATTR(o), ATTR(e))`

Attribute-based Access Control (ABAC)

- Example
  - an online entertainment store that streams movies to users for a flat monthly fee. The store must enforce the following access control policy based on the user's age and the movie's content rating:

Movie Rating	Users Allowed Access
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

  - In an RBAC model, every user would be assigned one of three roles: Adult, Juvenile, or Child, possibly during registration. There would be three permissions created: Can view R-rated movies, Can view PG-13-rated movies, and Can view G-rated movies. The Adult role gets assigned with all three permissions; the Juvenile role gets Can view PG-13-rated movies and Can view G-rated movies permissions, and the Child role gets the Can view G-rated movies permission only. Both the user-to-role assignment and the permission-to-role assignment are manual administrative tasks.
  - In ABAC:

```
•R1:can_access(u, m, e) ←
(Age(u) ≥ 17 AND Rating(m) ∈ {R, PG-13, G})
OR (Age(u) ≥ 13 AND Age(u) < 17 AND Rating(m) ∈
{PG-13, G})
OR (Age(u) < 13 AND Rating(m) ∈ {G})
```

Attribute-based Access Control (ABAC)

- as the number of attributes increases to accommodate finer-grained policies, the number of roles and permissions grows exponentially (why?). In contrast, the ABAC model deals with additional attributes in an efficient way. [In class discussion]

- Example (cont'd): suppose our online streaming service has two different types of membership:

```
R1:can_access(u, m, e) ←
        (Age(u) ≥ 17 AND Rating(m) ∈ {R, PG-13, G})
OR (Age(u) ≥ 13 AND Age(u) < 17 AND Rating(m) ∈ {PG-13, G})
        OR (Age(u) < 13 AND Rating(m) ∈ {G})

R2:can_access(u, m, e) ←
        (MembershipType(u) = Premium)
OR (MembershipType(u) = Regular AND MovieType(m) = OldRelease)

R3:can_access(u, m, e) ← R2 AND R1
```

- Suppose we wish to add a new policy rule that is expressed in words as follows: Regular users are allowed to view new releases in promotional periods. This would be difficult to express in an RBAC model. In an ABAC model, we only need add a conjunctive (AND) rule that checks to see the environmental attribute today's date falls in a promotional period.

PyCasbin

- “Casbin is a powerful and efficient open-source access
  projects. It provides support for enforcing authorizatio
  models.” https://github.com/casbin/pycasbin