Cybersecurity

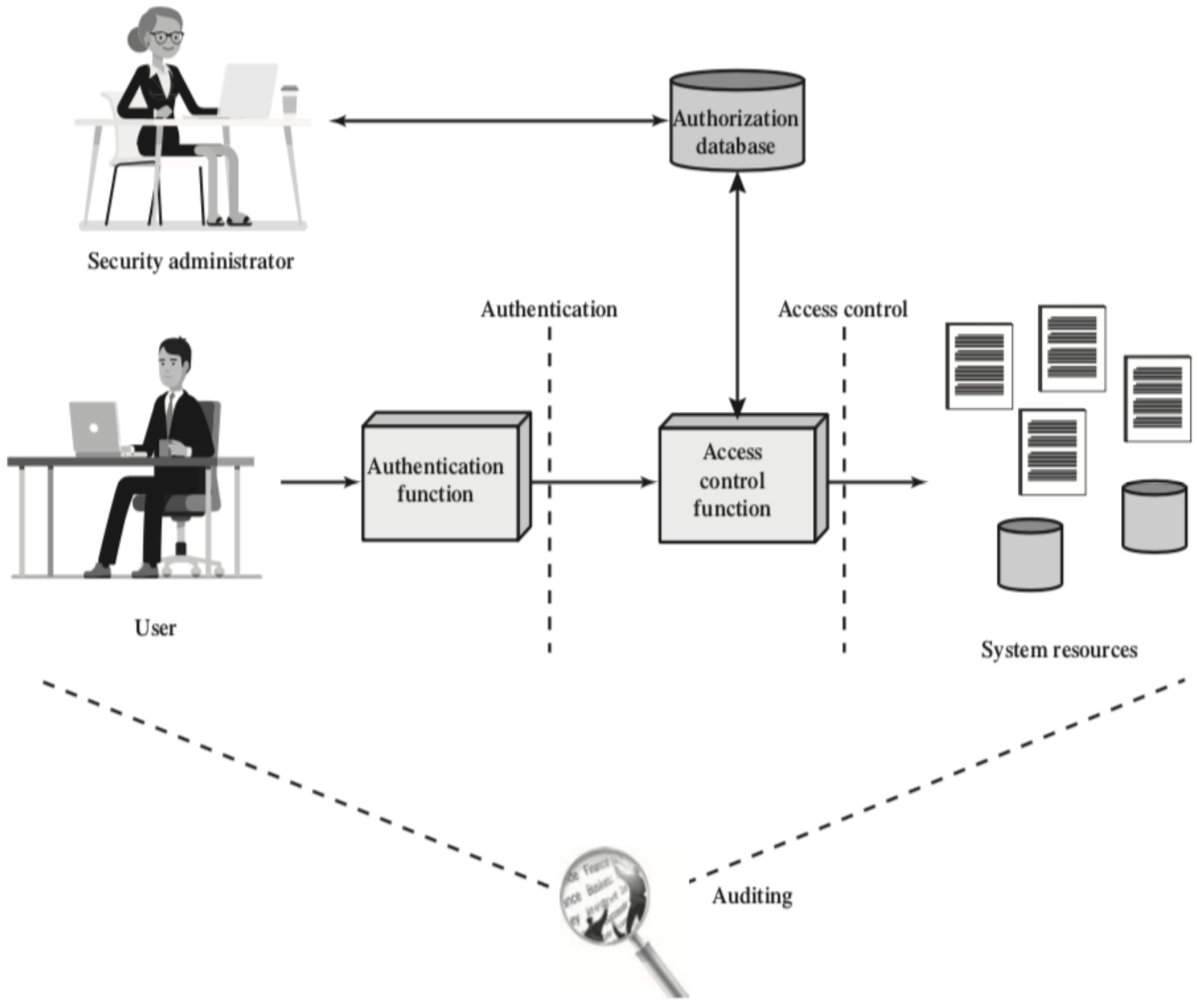
Access Control

Mauro Barni
University of Siena

Access control: a crucial building block

- **NISTIR 7298** defines access control as the process of granting or denying specific requests to:
  - obtain and use information and related information processing services
  - enter specific physical facilities.
- **RFC 4949, Internet Security Glossary**, defines access control as a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities according to that policy.

Involved entities



The authentication function determines whether the user is permitted to access the system at all.

The access control function determines if the specific requested access by this user is permitted.

The auditing function monitors and keeps a record of user accesses to system resources.

Access control policies: DAC

- **Discretionary access control**

  Controls access based on the identity of the requestor and on access rules 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.

Access control policies: MAC

- **Mandatory access control**

  Controls access 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.

Access control policies: RBAC, ABAC

- **Role-based access control**

  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**

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

Subjects, objects, access rights

- **Subject.** An entity capable of accessing objects. The concept of subject equates with that of process, ex:
  - *Owner* - the creator of a resource, such as a file. For system resources, ownership may belong to a system administrator.
  - *Group* - a named group of users may also be granted access rights, such that membership in the group is sufficient to exercise these access rights.
  - *World* – users who are able to access the system but are not included in the categories owner and group
- **Object:** Records, blocks, pages, segments, files, portions of files, directories, directory trees, mailboxes, mes- sages, and programs.

Subjects, objects, access rights

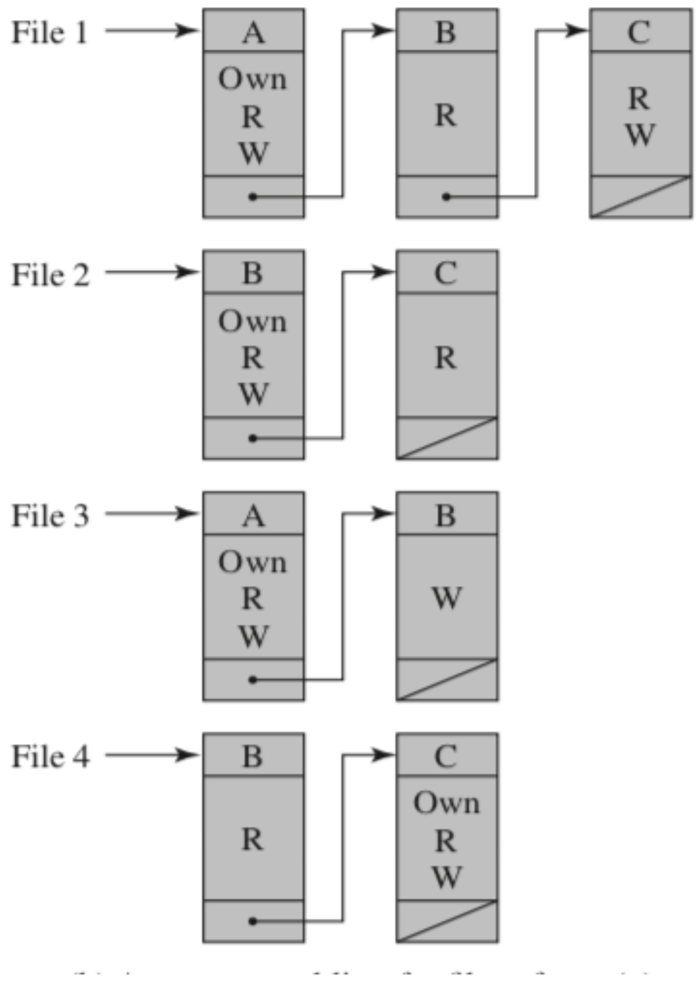
- **Access right** describes the way in which a subject may access an object.
  - *Read*: user may view information in a system resource, it includes the ability to copy or print
  - *Write*: user may add, modify, or delete data in system resource
  - *Execute*: user may execute specified programs.
  - *Create*: user may create new files, records, or fields
  - *Search*: user may list the files in a directory or otherwise search the directory.

Discretionary Access Control

- **Access control is granted in a discretionary way by an administrator or a user.**
- It is often described by means of an **access matrix**

users / objects	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5	Obj 6
User A		OWR				R
User B	OWR	WR			OWR	OWR
User C		WR		OWR		R
User D	R	WR	OWR			R

Access control lists

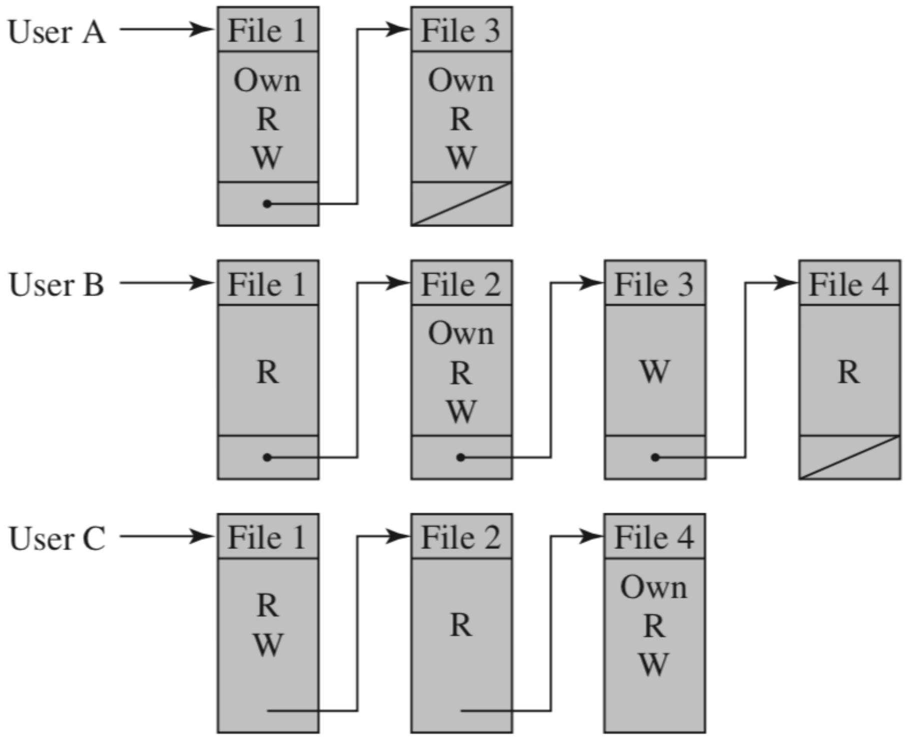


ACL may contain default entries for users which are not explicitly listed

ACLs are convenient to determine the access rights related to a specific object

ACLs are difficult to use to check the rights of a specific user

Capability tickets



Decomposing the access matrix by rows gives a capability ticket

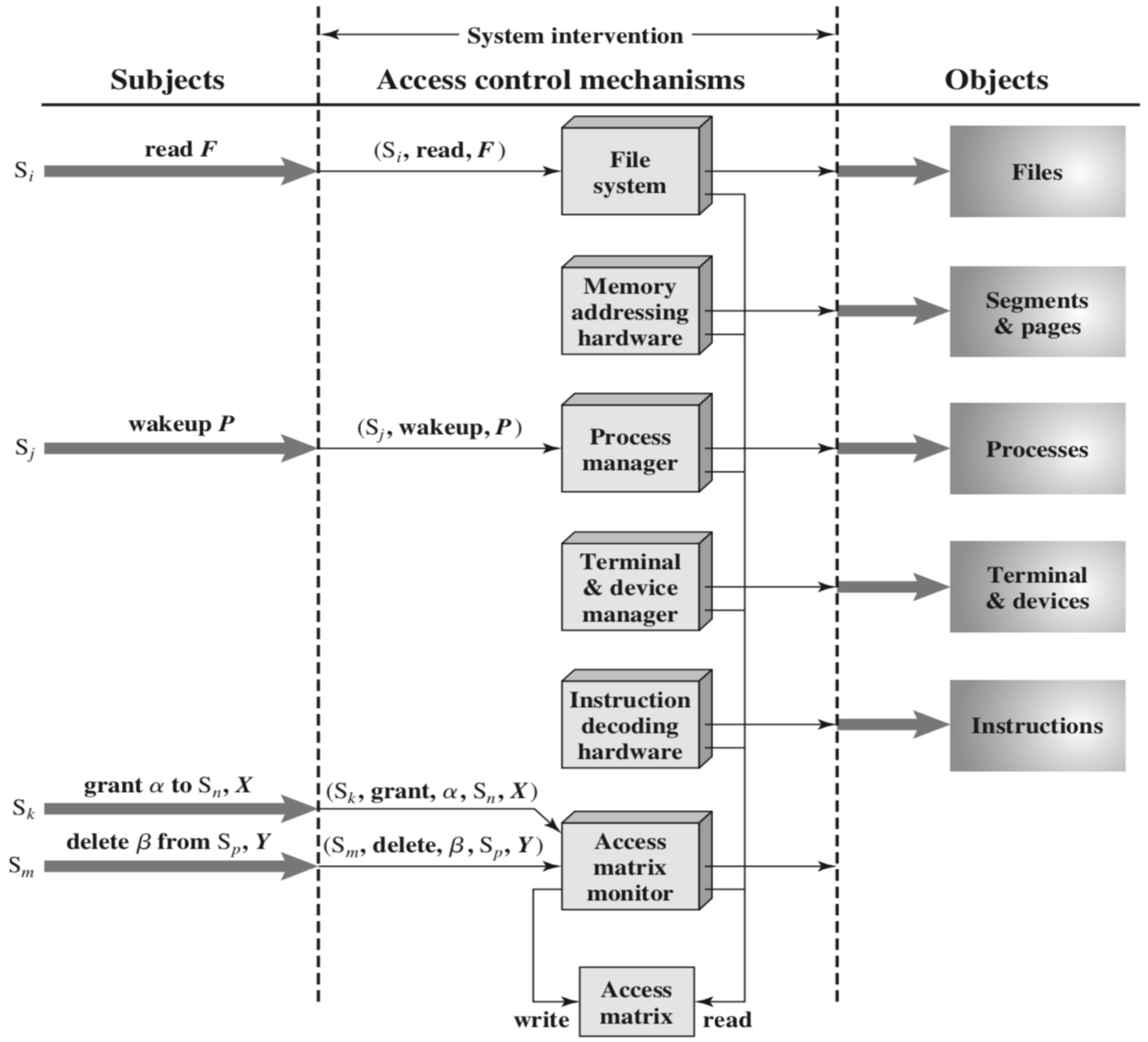
CTs are given directly to users and are then spread all across the system

In some cases users may loan or distribute CTs

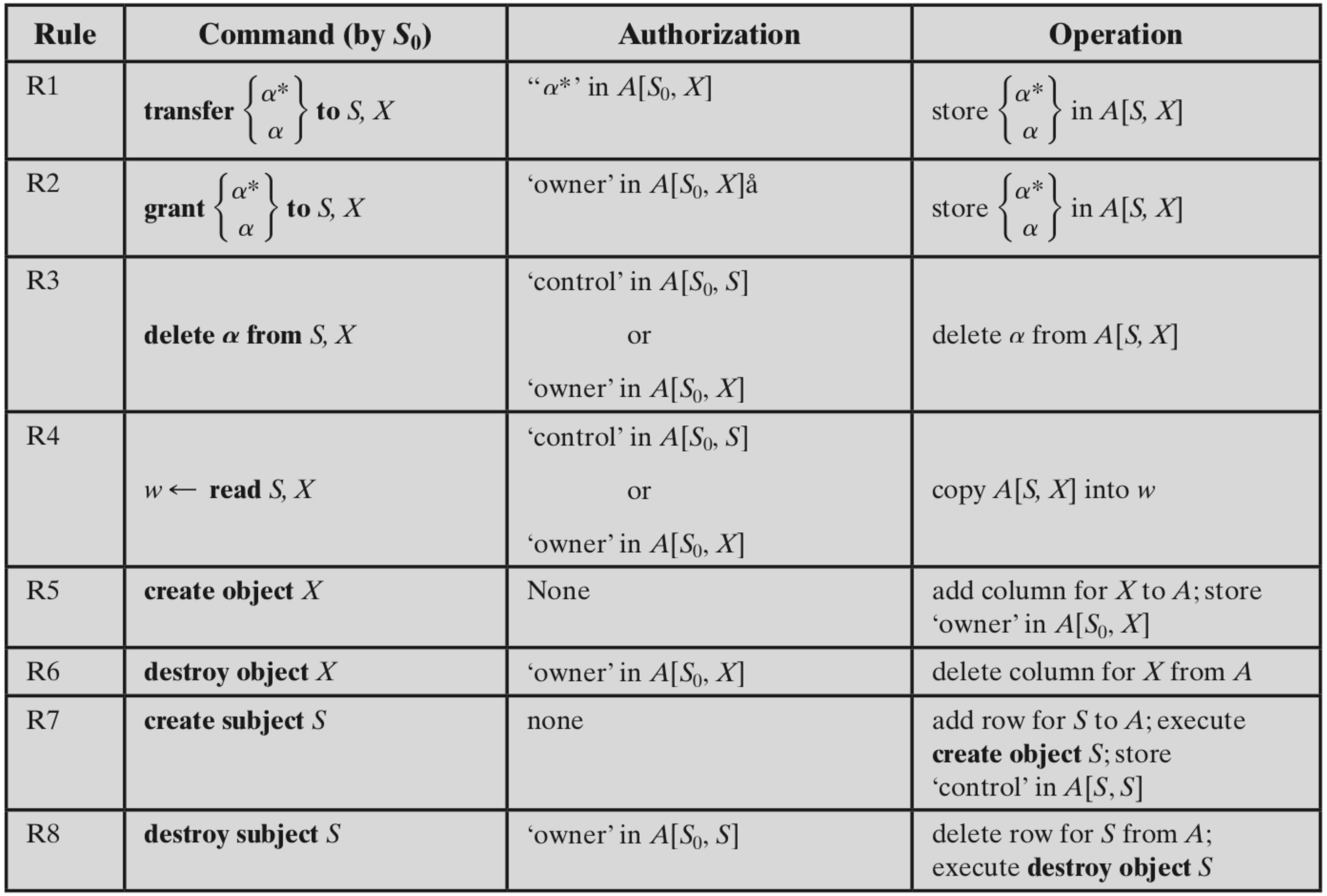
CTs are sometimes protected by MAC or crypto

DAC Access control model

- An access control model has three main functionalities.
  - represent the protection state of the system
  - enforce access policy
  - allow subjects to alter the protection state in certain ways



Rules to modify the access matrix

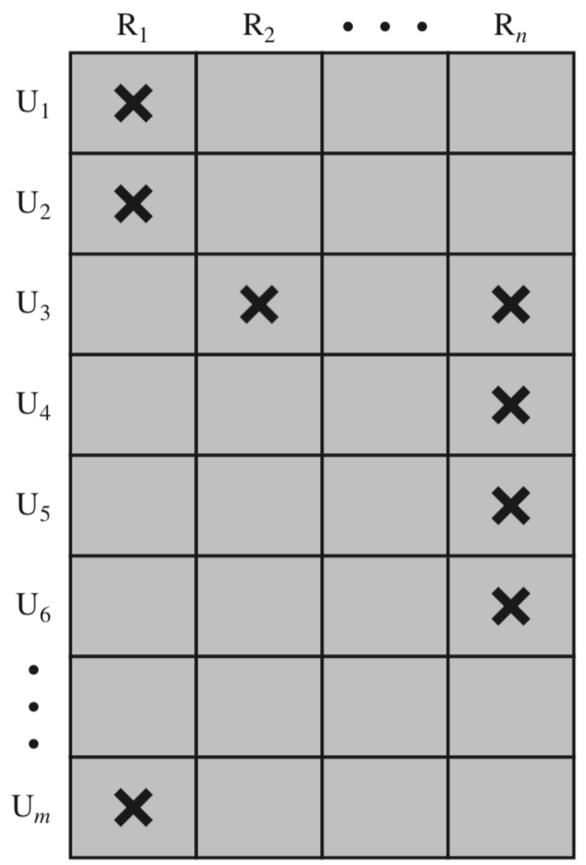


Rule	Command (by S_0)	Authorization	Operation
R1	**transfer** $\left\{\begin{matrix}\alpha^* \\ \alpha\end{matrix}\right\}$ **to** S, X	"α^*" in $A[S_0, X]$	store $\left\{\begin{matrix}\alpha^* \\ \alpha\end{matrix}\right\}$ in $A[S, X]$
R2	**grant** $\left\{\begin{matrix}\alpha^* \\ \alpha\end{matrix}\right\}$ **to** S, X	'owner' in $A[S_0, X]$å	store $\left\{\begin{matrix}\alpha^* \\ \alpha\end{matrix}\right\}$ 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

Role-Based Access Control: RBAC

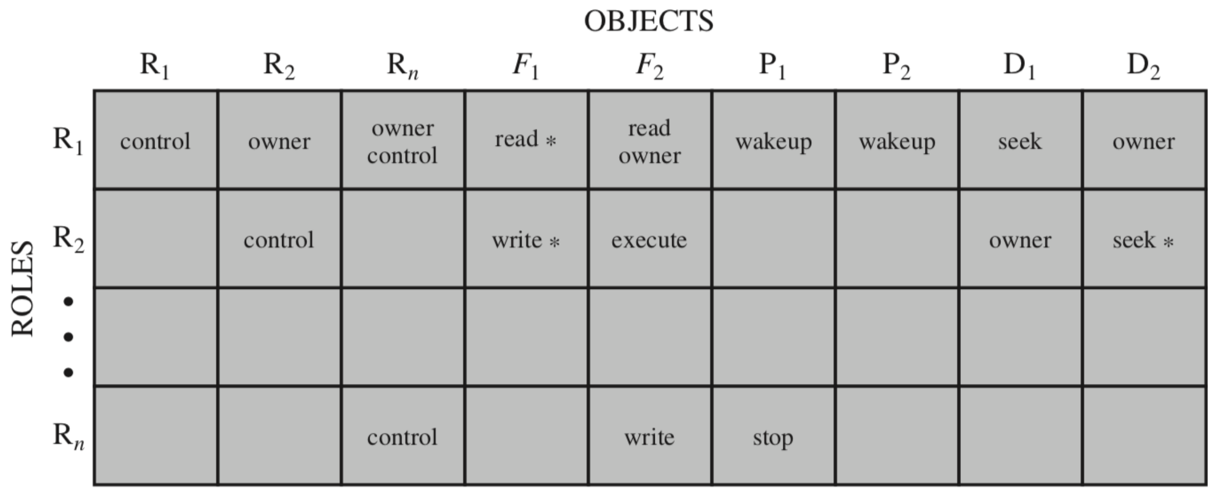
- Access rights are assigned on the base of roles rather than to users directly
- Users are assigned to roles either statically or dynamically according to their responsibilities in the system
- Relationship between users and roles is *many to many*
- RBAC is getting more and more popularity due to its flexibility with respect to DAC

Representation of protection state



	R_1	R_2	• • •	R_n
U_1	✗			
U_2	✗			
U_3		✗		✗
U_4				✗
U_5				✗
U_6				✗
• • •				
U_m	✗			

Users vs roles table



		OBJECTS								
		R_1	R_2	R_n	F_1	F_2	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			

Access control Matrix

RBAC permits to easily implement the principle of least required privilege. Each role should contain the minimum set of access rights needed for that role. A user is assigned to a role that enables to perform only what is required for that role.

Attribute-Based Access Control: ABAC

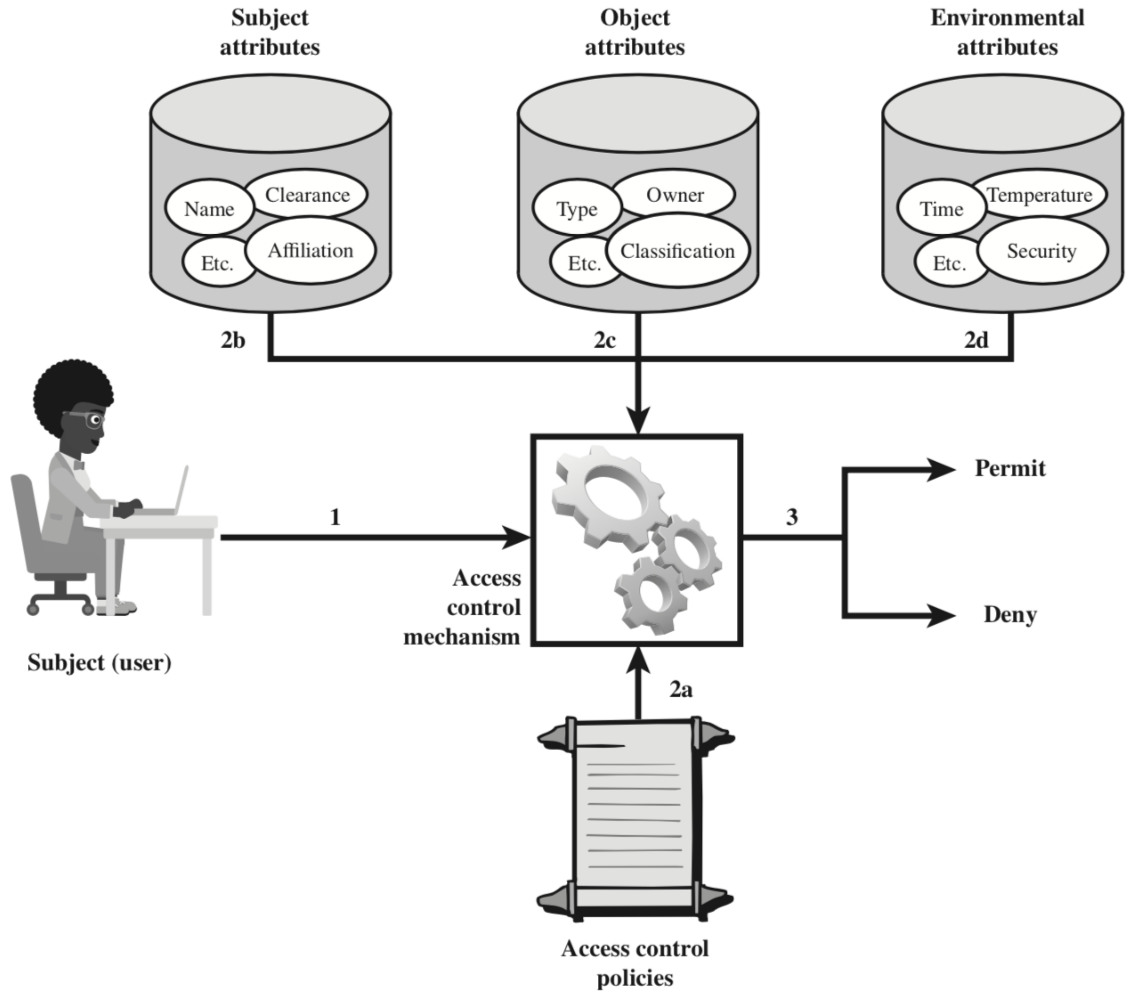
- New approach, which is gaining consensus due to its extreme flexibility (at the expense of complexity)
- Can implement DAC, MAC and RBAC as special cases
- Access control is granted based on rules applied to attributes of both the objects and the subjects requesting access

- Research still going on

Attributes

- *Subject attributes:* each subject has associated attributes that define the identity and characteristics of the subject. They 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*: objects have attributes that can be lever- aged to make access control decisions. 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. Examples include current date and time, the network’s security level etc …

ABAC access control architecture



Policy rules: example

- An online entertainment service wants to enforce the following access rule

Movie rating	Access rule
Adult (R)	Age 18 or older
Young (PG-13)	Age 14 or older
Everybody (G)	Everybody

- With RBAC we could define three roles and state in an access control matrix which roles can view each movie

Policy rules: example

- With ABAC we can define the following attributes and policy rule

Subject/ Object	Attributes	Values
User	Age	[1-100]
Movie	Rating	{R, PG-13, G}

$$
\begin{aligned}
&\mathrm{R1{:}can\_access}(u, m, e) \leftarrow \\
&\quad (\mathrm{Age}(u) \geq 17 \land \mathrm{Rating}(m) \in \{\mathrm{R}, \mathrm{PG\text{-}13}, \mathrm{G}\}) \lor \\
&\quad (\mathrm{Age}(u) \geq 13 \land \mathrm{Age}(u) < 17 \land \mathrm{Rating}(\mathrm{m}) \in \{\mathrm{PG\text{-}13}, \mathrm{G}\}) \lor \\
&\quad (\mathrm{Age}(u) < 13 \land \mathrm{Rating}(m) \in \{\mathrm{G}\})
\end{aligned}
$$

Policy rules: example

- Suppose now that only premium members can access new movies
- With RBAC, we would need to define 6 roles:
  - (<14, regular, <14 premium … >17 premium
- It is easily seen that the complexity of RBAC access matrix grows exponentially with the number of attributes

Policy rules: example

- With ABAC everything is much simpler

Subject/Object	Attributes	Values
User	Age	[1-100]
User	Subscription	{Regular, Premium}
Movie	Release	{New release, old release}
Movie	Rating	{R, PG-13, G}

$$\begin{aligned}
&\texttt{R2:can\_access}(u,\ m,\ e) \leftarrow \\
&\quad (\texttt{MembershipType}(u) = \texttt{Premium}) \lor \\
&\quad (\texttt{MembershipType}(u) = \texttt{Regular} \land \texttt{MovieType}(m) = \texttt{OldRelease}) \\
&\texttt{R3:can\_access}(u,\ m,\ e) \leftarrow \texttt{R1} \land \texttt{R2}
\end{aligned}$$

- Think about a new rule according to which regular users can access new movies in promotional periods

References

- W. Stallings, L. Brown, “*Computer security: principles and practices*”*,* Pearson, 4-th edition. Chapter 4.
- Lectures notes (these slides)