Access Control

System Security (CM0625, CM0631) 2024-25
Università Ca' Foscari Venezia

Riccardo Focardi
www.unive.it/data/persone/5590470
secgroup.dais.unive.it

Università
Ca'Foscari
Venezia

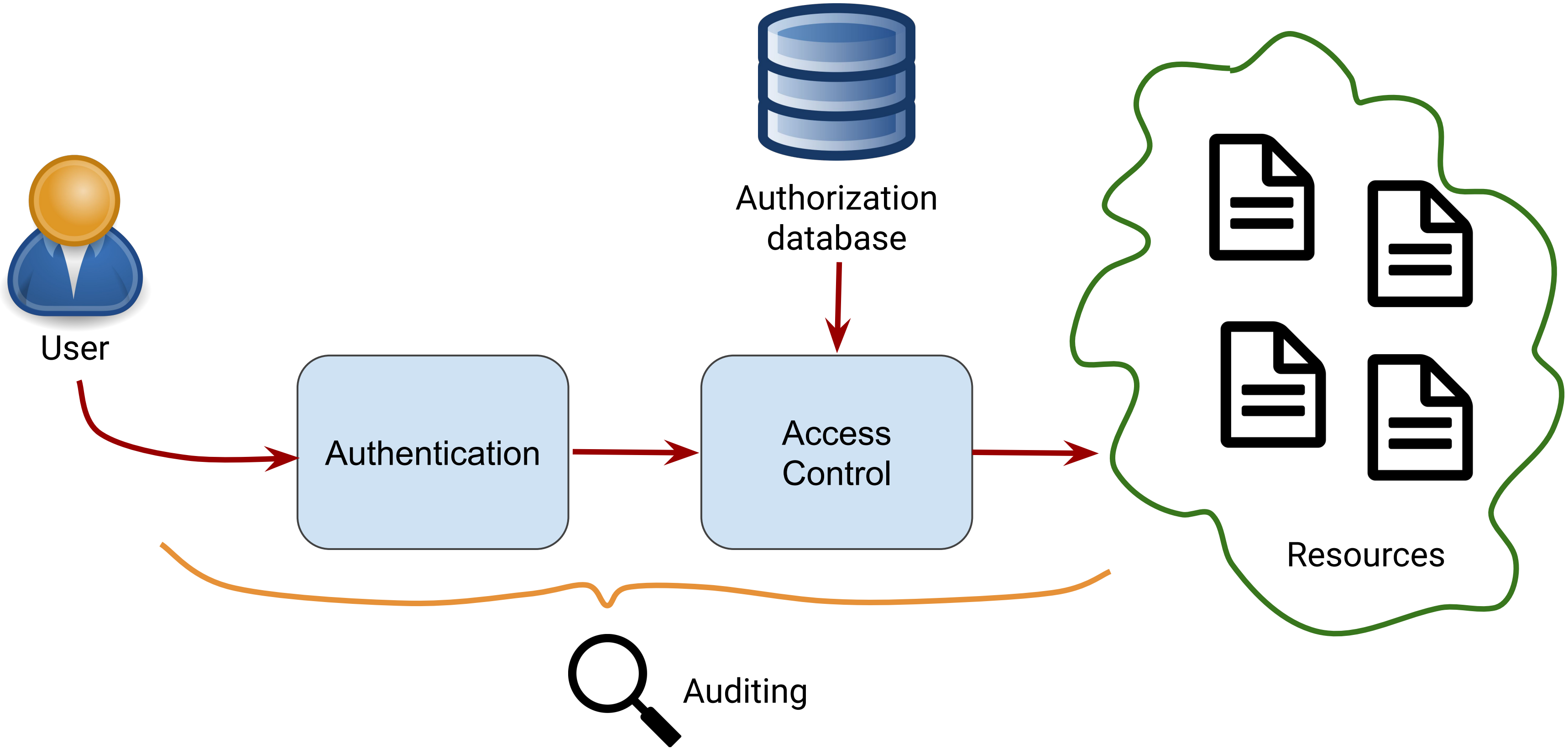
Definition

RFC 4949
Internet Security Glossary

Access Control: *Protection of system resources against unauthorized access*

- The process regulating the use of **system resources** according to a **security policy**
- Access is permitted only by **authorized** entities (users, programs, processes, or other systems) according to that policy.

Context


Authorization database
User
Authentication
Access Control
Resources
Auditing

Context

Authentication: Verification that the credentials of a user or other system entity are valid

Authorization: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

Audit: An independent review and examination of system records and activities in order to

- **test** for adequacy of controls
- **ensure compliance** with established policy and operational procedures
- **detect breaches** in security, and recommend changes in control, policy, and procedures

Subjects and objects

Subject: is an entity capable of accessing resources (objects)

- Any user or application actually gains access to an object by means of a **process**
- The process inherits the attributes of the user, such as the access rights

Object: is a resource to which access is controlled. An object is an entity used to contain and/or receive information

Examples: pages, segments, files, directories, mailboxes, messages, programs, communication ports, I/O devices.

Access rights

Read: Subject may view information in an object; read access includes the ability to copy or print

Write: Subject may add, modify, or delete data in an object

Execute: Subject may execute an object (e.g. a program)

Delete: Subject may delete an object

Create: Subject may create an object

Search: Subject may search into an object (e.g., a query giving a partial view of the content)

Note: one access right might imply another one, e.g. read ⇒ search

Access control policies

Discretionary Access Control (DAC)

Mandatory Access Control (MAC)

Role-Based Access Control (RBAC)

Attribute-Based Access Control (ABAC)

Discretionary access control (DAC)

Access matrix: access rights for each subject (row) and object (column)

	README.txt	/etc/shadow	Carol.pdf	/bin/bash
Alice	Read Write	Read Write		Read Write Execute
Bob	Read			Read Execute
Carol	Read		Read Write	Read Execute

NOTE: can be **sparse**!

Access Control Lists vs. Capabilities

Access Control List (ACL): for each object, lists subjects and their permission rights (decomposition **by columns**)

- Easy to find which subjects have access to a certain object
- Hard to find the access rights for a certain subject

Example: ACL

README.txt:

Alice: Read, Write;

Bob: Read;

Carol: Read.

/etc/shadow:

Alice: Read, Write.

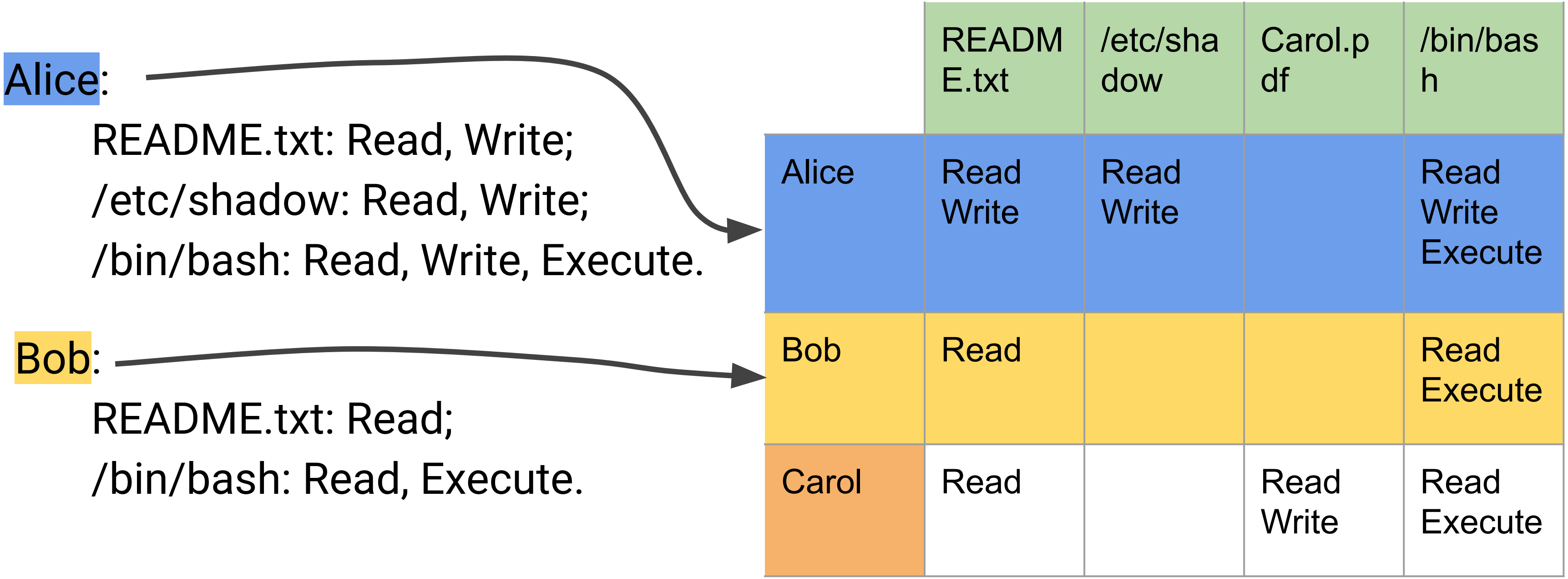
	README.txt	/etc/shadow	Carol.pdf	/bin/bash
Alice	Read Write	Read Write		Read Write Execute
Bob	Read			Read Execute
Carol	Read		Read Write	Read Execute

Access Control Lists vs. Capabilities

Capabilities: for each subject, list objects and access rights to them (decomposition **by rows**)

- Easy to find the access rights for a certain subject
- Hard to find which subjects have access to a certain object

Example: Capabilities



Alice:

README.txt: Read, Write;
/etc/shadow: Read, Write;
/bin/bash: Read, Write, Execute.

Bob:

README.txt: Read;
/bin/bash: Read, Execute.

	README.txt	/etc/shadow	Carol.pdf	/bin/bash
Alice	Read Write	Read Write		Read Write Execute
Bob	Read			Read Execute
Carol	Read		Read Write	Read Execute

Authorization table

IDEA: store an entry for each **subject**, **access right**, and **object**

- Querying by subject gives **capabilities**
- Querying by object gives **ACLs**

Subject	Access right	Object
Alice	Read	README.txt
Alice	Write	README.txt
Alice	Read	/etc/shadow
Alice	Write	/etc/shadow
Alice	Read	/bin/bash
Alice	Write	/bin/bash
Alice	Execute	/bin/bash
Bob	Read	README.txt
Bob	Read	/bin/bash
Bob	Execute	/bin/bash
...	...	...

DAC is … discretionary

A subject can give access to the object it **owns**

In some systems, access rights can be given with a **copy flag** so that non-owners can pass the right to other subjects

NOTE: programs typically **inherits** user's access rights

1. **Attack scenario**: A malware program executed by Alice can leak Alice's sensitive data by simply giving read access to (malicious) Bob
2. Alice might **erroneously** give read access to her sensitive files

⇒ Discretionary Access Control is too flexible

Access control policies

Discretionary Access Control (DAC)

Mandatory Access Control (MAC)

Role-Based Access Control (RBAC)

Attribute-Based Access Control (ABAC)

Mandatory Access Control (MAC)

MAC imposes rules that subjects cannot change

Example: Alice has clearance *secret* that allows her to own and access secret files but does not allow her to make those files accessible to *unclassified* users, such as Bob.

MAC prevents:

1. **Leakage due to malware** that would run with clearance *secret* too, and won't be able to communicate towards *unclassified* users
2. **Leakage due to errors**: Any file created by Alice would automatically have level *secret*

Example 1: Bell - La Padula (BLP)

Security levels: define the level of security wrt a certain property, e.g. Confidentiality.

Example: inspired from military

1. *top secret*
2. *secret*
3. *confidential*
4. *restricted*
5. *unclassified*

Subjects and objects are assigned to security levels

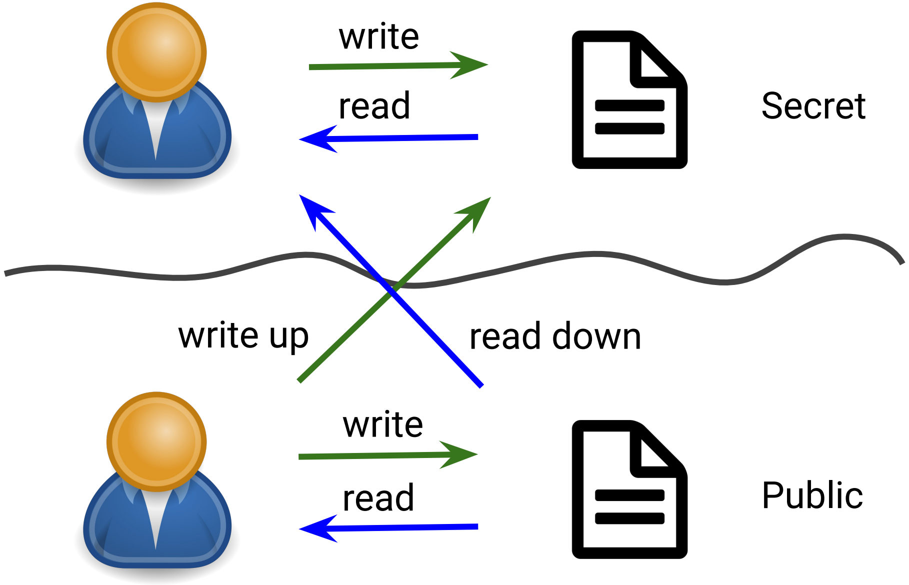
- **Clearance**: the security level of subjects
- **Classification**: the security level of objects

BLP (confidentiality)

Definition: Information should never flow from a level to lower ones

- **Simple security**: Subjects cannot read from objects at a higher level
- ***-property**: Subjects cannot write into objects classified at a lower level

… plus **standard DAC**!

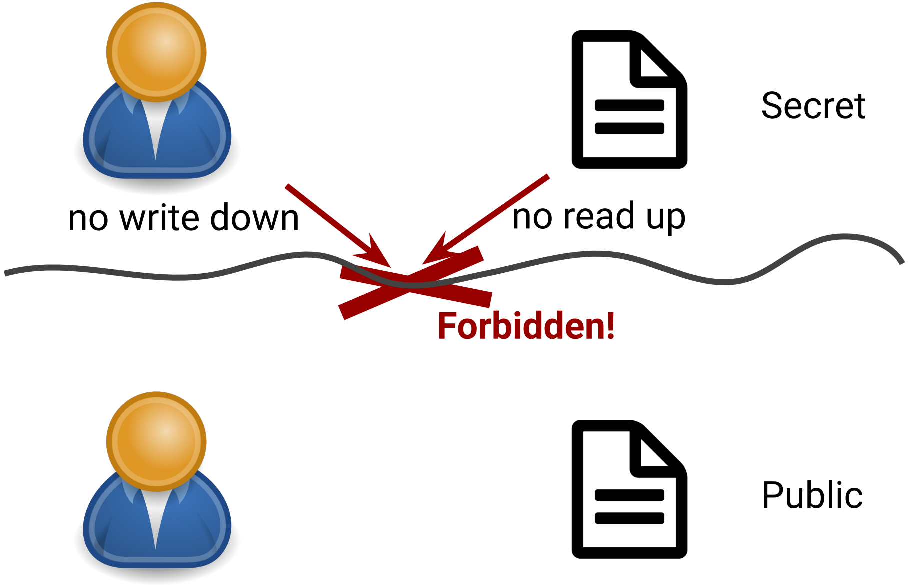


BLP (confidentiality)

Definition: Information should never flow from a level to lower ones

- **Simple security**: Subjects cannot read from objects at a higher level
- ***-property**: Subjects cannot write into objects classified at a lower level

… plus **standard DAC**!

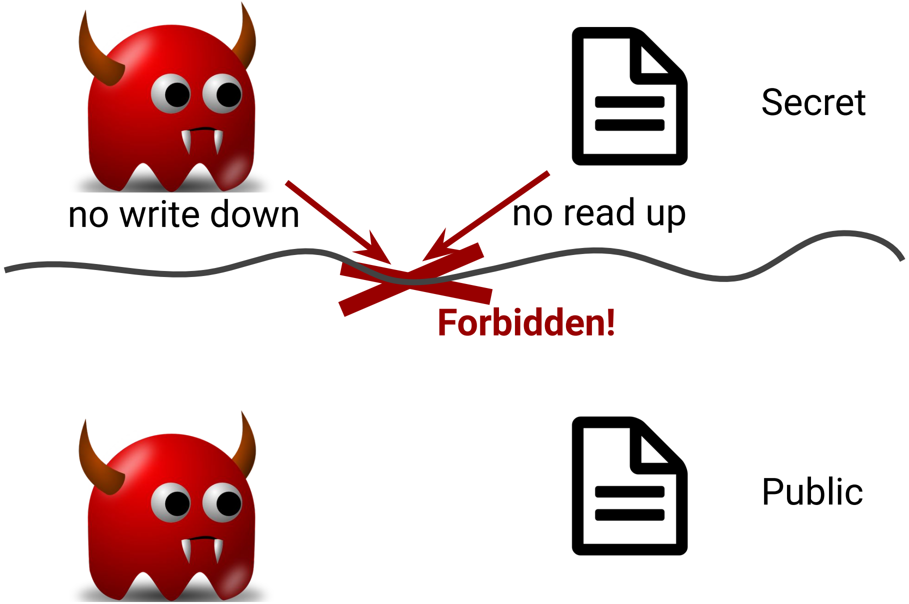


BLP (confidentiality)

Definition: Information should never flow from a level to lower ones

- **Simple security**: Subjects cannot read from objects at a higher level
- ***-property**: Subjects cannot write into objects classified at a lower level

… plus **standard DAC**!

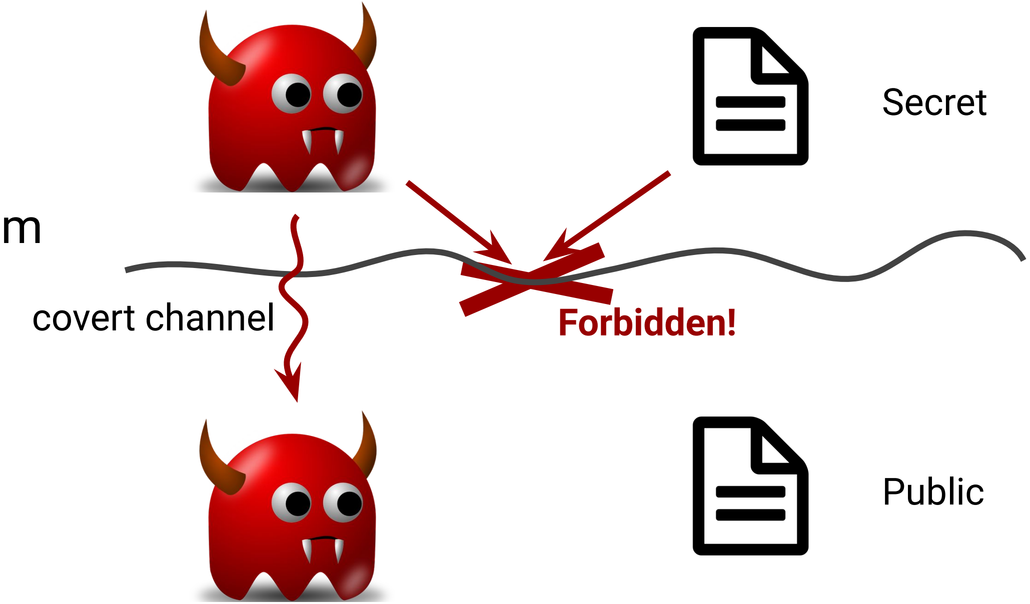


Problem: covert channels

Definition: A way to indirectly transmit information

Example: A shared resource that is slowed down by a malicious program might be used to encode bits:

- Slow ⇒ 0
- Fast ⇒ 1



Ex 2: Chinese wall

Goal: prevent conflicts of interest

- Objects belongs to **company datasets**
- The company datasets belong to **conflict of interest classes**

Idea: Subjects cannot access objects from different companies that belong the same conflict of interest class

Example:

- Bank A,
  Oil company B,
  Oil company C
- B and C objects are in conflict

Subject S accesses an object from B:

- S can access more B's objects
- S **cannot** access C's objects
- S can access A's objects

Chinese wall policy (read)

Simple security: read access is granted if object

- is in the same company dataset as an already accessed object

or

- belongs to an entirely different conflict of interest class

Problem with write access:

- Bank A,
  Oil company B,
  Oil company C
- B and C are in conflict

1. Subject S’ reads from C
2. Subject S reads from B and write into an A
3. Subject S’ reads from A
   ⇒ **Conflict!**

Chinese wall policy (write)

***-property**: write access is granted if

- access is permitted by simple security property

and

- no object can be read which is in a different company dataset to the one for which write access is requested

NOTE: This rule is **very restrictive**: read/write permission is only possible on single company datasets

In the original paper authors propose the idea of **sanitized information**, i.e., company information that does not require protection

Relaxed *-property:
and contains unsanitized information

Access control policies

Discretionary Access Control (DAC)

Mandatory Access Control (MAC)

Role-Based Access Control (RBAC)

Attribute-Based Access Control (ABAC)

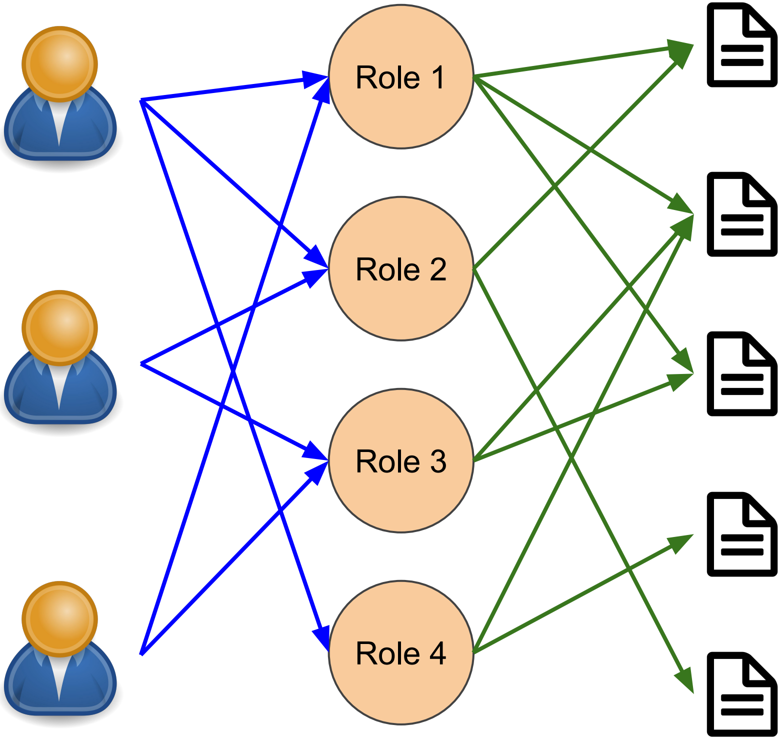
Role-Based Access Control (RBAC)

DAC specifies access rights for each subject and object

RBAC adds a new layer: **roles**

- Subjects are assigned to roles
- Roles have access rights to objects

NOTE: RBAC can express DAC and MAC policies



RBAC access matrix

Access matrix: access rights for each role (row) and object (column)

	README.txt	/etc/shadow	Carol.pdf	/bin/bash
Administrator	Read Write	Read Write		Read Write Execute
Student	Read			Read Execute
Professor	Read		Read Write	Read Execute

RBAC role assignment

Role assignment: for each subject (row) and role (column)

	Administrator	Student	Professor
Alice	x		x
Bob		x	
Carol			x

Note: we can have multiple roles per user and multiple users per role

Hierarchies and exclusive roles

Users establish sessions with the **roles they need** to accomplish a task (least privilege principle)

Roles can be organized as a **hierarchy**:

Example:

Professor → Department Dean
Professor → Rector

Roles might be **mutually exclusive** to enforce *separation of duties*

Separation of duties: if one task requires two users to be performed

Examples:

- creating vs. authorizing an account
- auditing vs. performing a task

Access control policies

Discretionary Access Control (DAC)

Mandatory Access Control (MAC)

Role-Based Access Control (RBAC)

Attribute-Based Access Control (ABAC)

Attribute-Based Access Control (ABAC)

IDEA: Access regulated through attributes

Subject attributes: name, title, age, …

SA_1, …, SA_K

Object attributes: author, category, …

OA_1, …, OA_M

Environment attributes: date, setting, connection, …

EA_1, …, EA_N

For each subject s, object o and environment e:

$$\mathrm{ATTR}(s) \in SA_1 \times SA_2 \times \ldots \times SA_K$$
$$\mathrm{ATTR}(o) \in OA_1 \times OA_2 \times \ldots \times OA_M$$
$$\mathrm{ATTR}(e) \in EA_1 \times EA_2 \times \ldots \times EA_N$$

```
can_access(s,o,e) =
 f(ATTR(s),ATTR(o),ATTR(e))
```

ABAC example

Access to online streaming

```
can_access(s,o,e) =
 (
  (Membership(s) == Premium)
  ∨
  (Membership(s) == Regular ∧
   Type(o) == OldRelease)
 )
 ∧
 ( ExpireDate(s) >= Time(e) )
```

👍 ABAC is more flexible than RBAC

👍 ABAC can express DAC, MAC, and RBAC

👎 Access decision is more complex

⇒ On the Web and Cloud is more and more **popular** (performance is already limited by network latency)

Exercise: define BLP with ABAC

BLP is no read-up no write-down

What attributes?

- Use security levels! `clearance(s)` and `classification(o)` are the security levels of s and o
- `access_right(e)` in `{read,write}`

```
BLP_can_access(s,o,e) =
    access_right(e) == read  AND clearance(s) >= classification(o)
  OR
    access_right(e) == write AND clearance(s) <= classification(o)
```