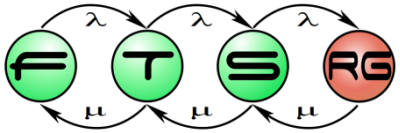
Authentication and authorization

Tóth Dániel, Micskei Zoltán

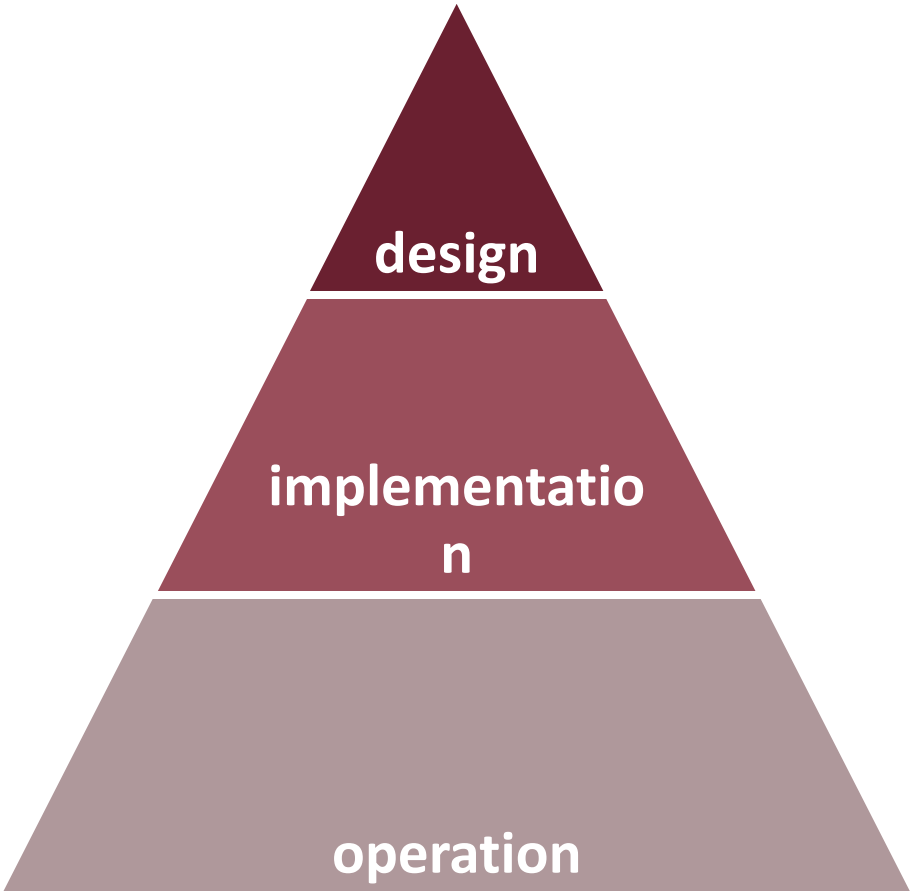


Security of computer systems

- Is it important?
- Is it important for everyone?
- When is it important?

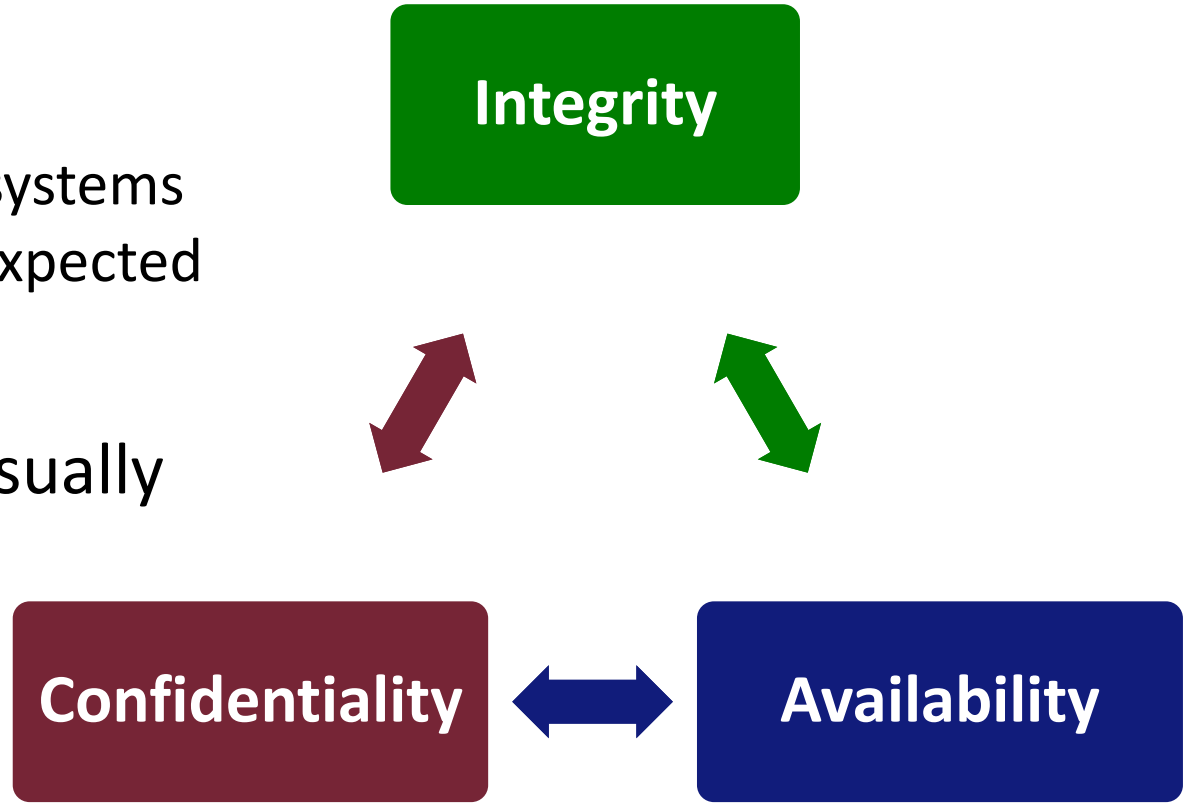
When is security important?

- In every phase of software development
- *If the system was not designed for security, it is really hard to make it secure.*
- Security is determined by the weakest link.



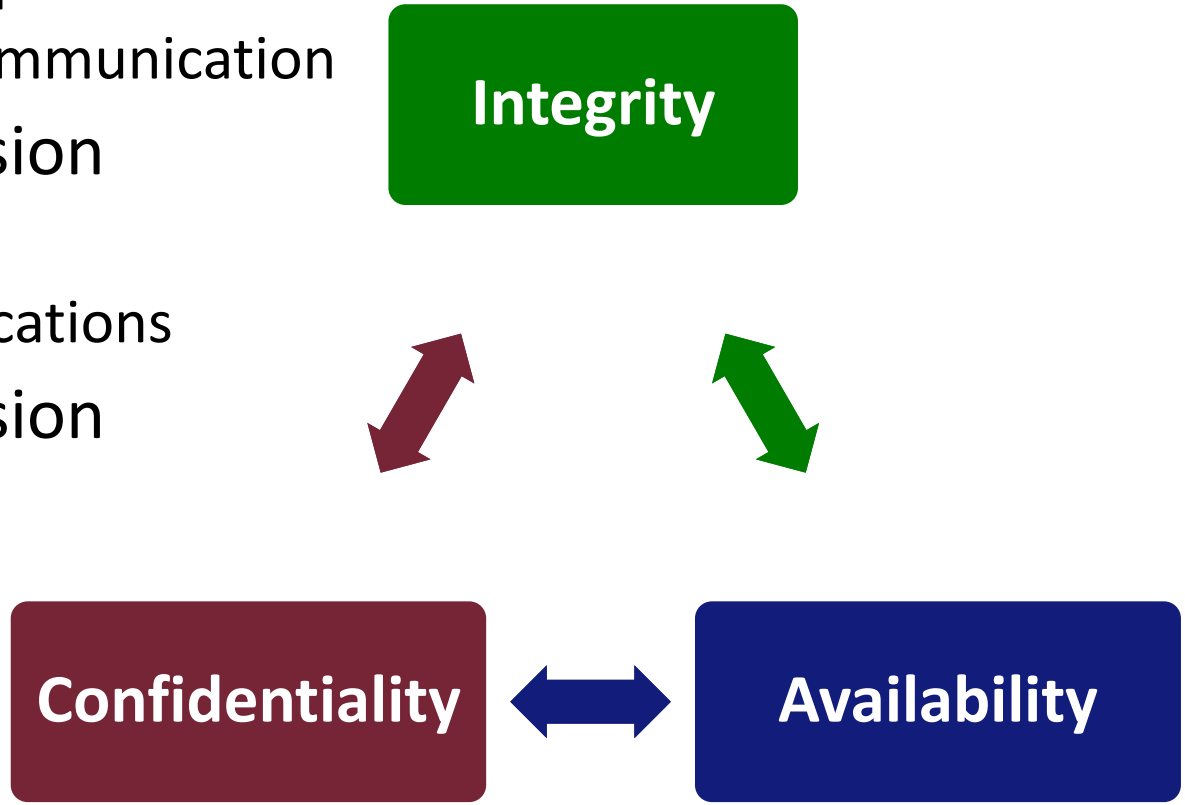
What is security?

- „C.I.A.”: three related concepts
- Goal:
  - guarantee that the systems behaves always as expected
- One technology is usually not enough

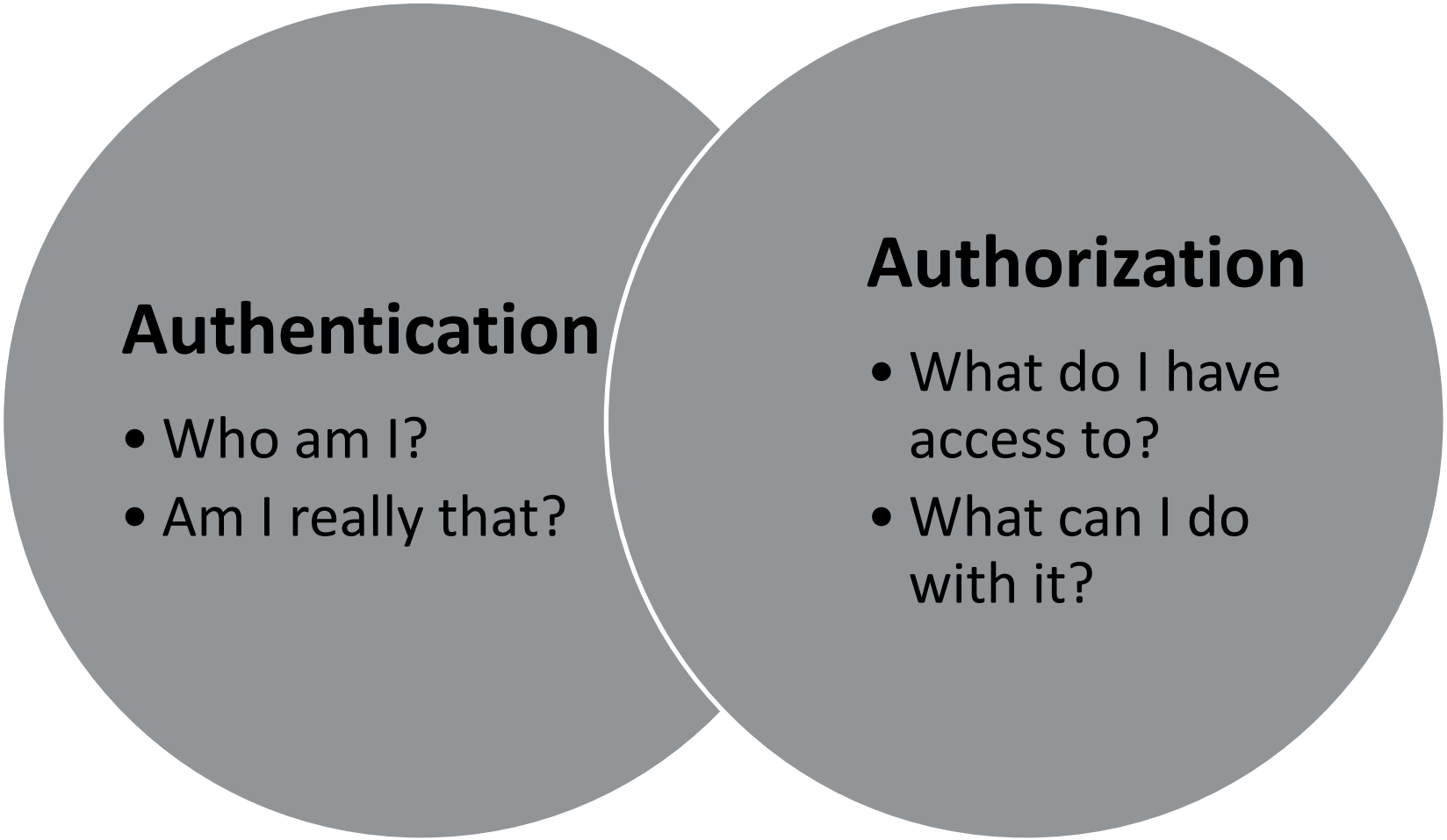


Methods for security

- Cryptography
  - For the integrity and confidentiality of communication
- Platform-level intrusion detection
  - Integrity if the applications
- Network-level intrusion detection
- Redundancy, reconfiguration
  - For availability
- **Authentication, authorization**

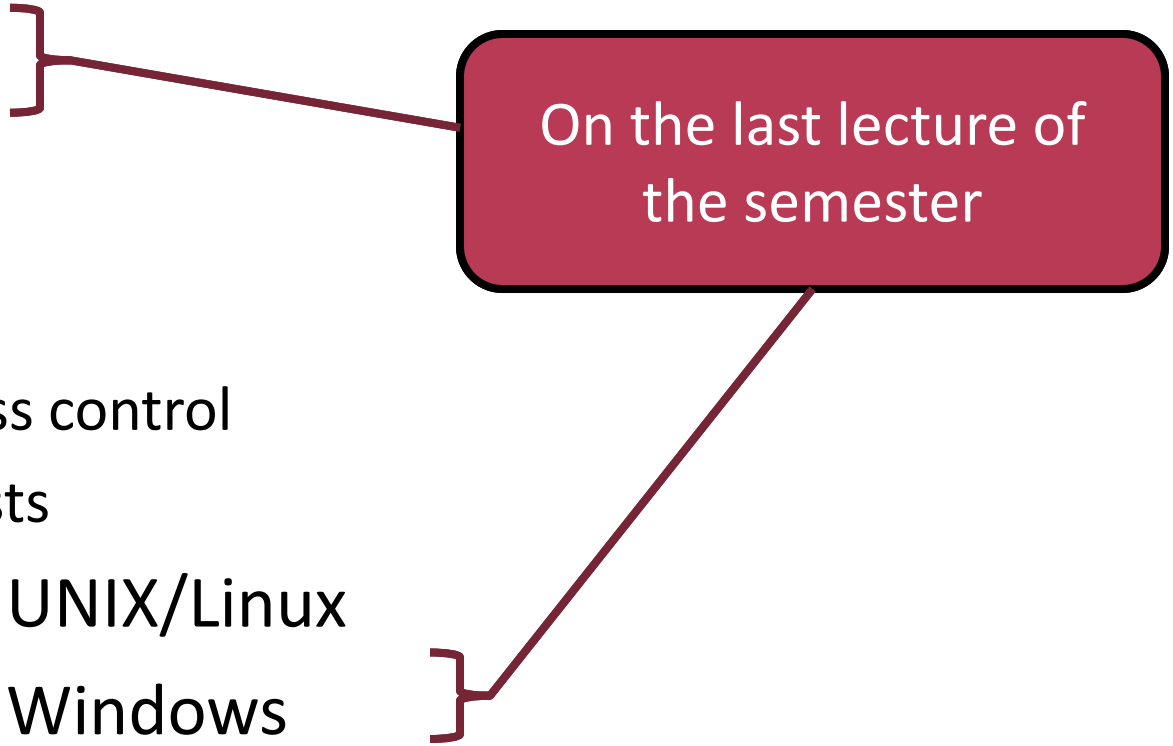


Who is “authorized”?



Content

- Short security introduction
- **User management, authentication**
  - UNIX, Linux
  - Windows
- Authorization
  - General methods
    - Role-based access control
    - Access control lists
  - Authorization on UNIX/Linux
  - Authorization on Windows



Authentication

- How can be the identity of the user decided?
  - …knows (e.g. password)
  - …has (e.g. keycard, security token)
  - …is (e.g. biometric, fingerprint)

- A (non-compromised) machine can decide the identity of the user using these methods
  - But what if the machine is compromised?
  - What to do with machine-machine communication?

Authentication

- Authentication on 3 levels:
  - Human–machine interaction
  - Machine–machine interaction over network
  - Between processes inside an OS

- Authentication protocols are needed
  - Machine–machine only the “knows” principle
  - But complex cryptographic primitives can be used

What is a user account?

User
+ ID + Name + Real Name + Personal data… + Shared Secret (Password, etc.) + Private Datastore path

For the system, the user is an object…

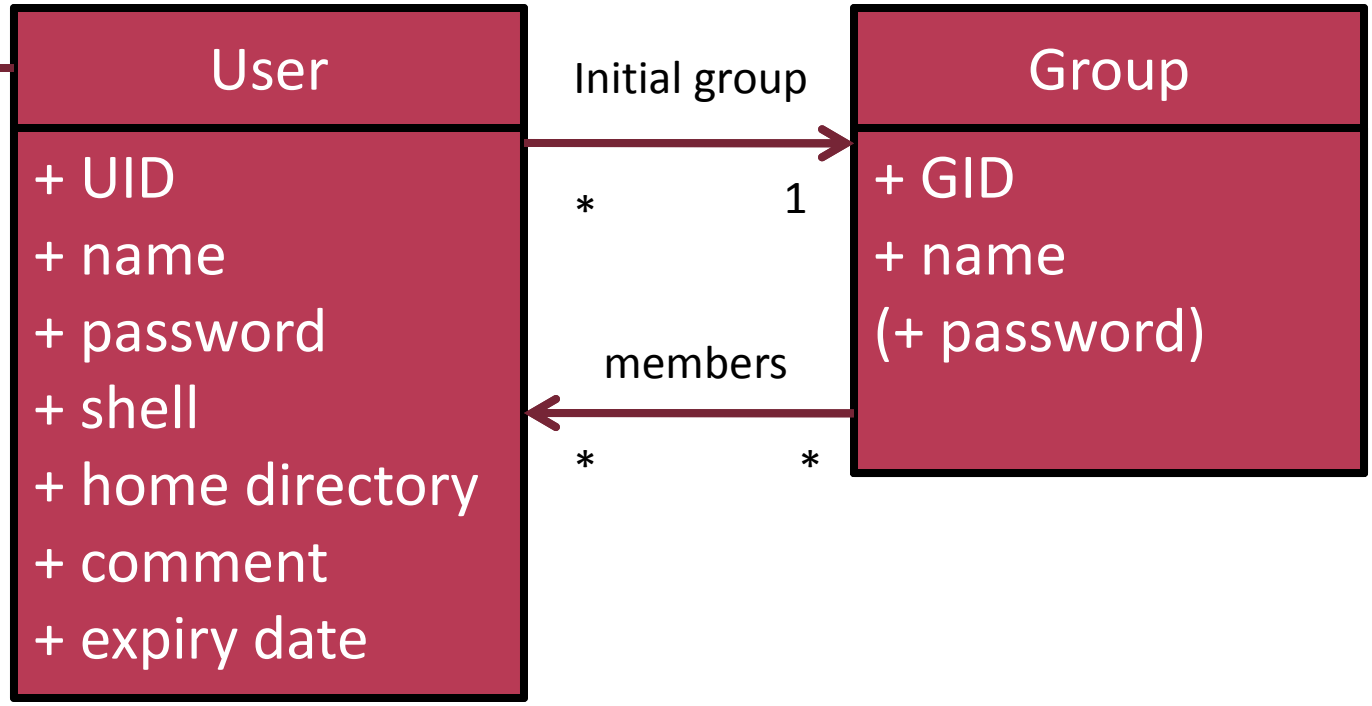
What is a user account?

- Unique identifier for an account
  - Linux, UNIX: UID (integer, root 0, users 1000-…)
- Further attributes of an account
  - Stored in /etc/passwd, /etc/shadow, /etc/groups
  - Examples
    - Login name
    - Password
    - Home directory
    - Default shell
    - Real name…

DEMO User account on Linux

- Stored in the following files:
  - `/etc/passwd`
  - `/etc/shadow`
  - `/etc/group`
- Create, delete, modify
  - `useradd, usermod, userdel`
  - `groupadd, groupmod, groupdel`
  - `passwd`

User account on Linux



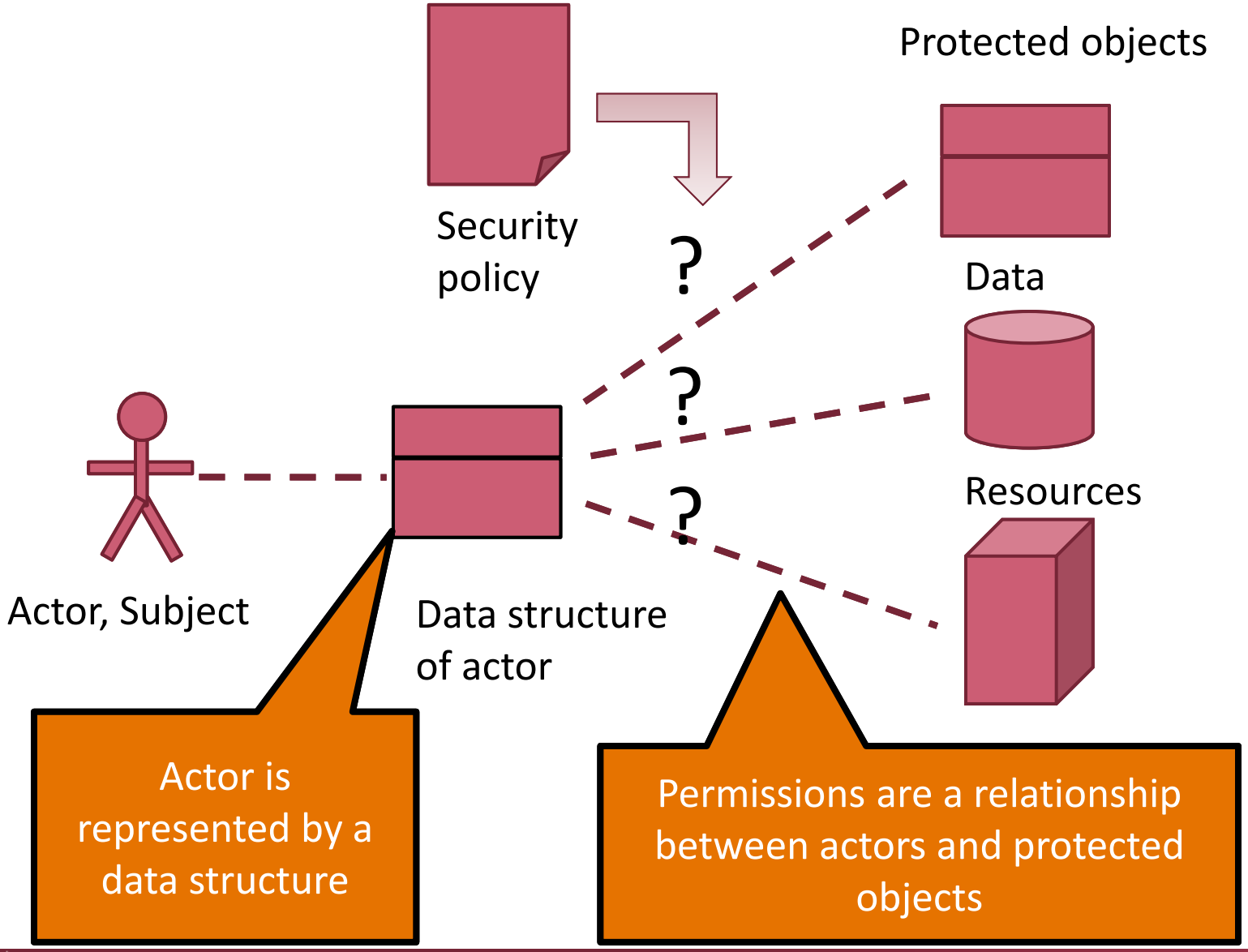
DEMO Process identity

- Identifying the identity of a process
  - `ps aux, pstree, /proc/$PID/status`
- Changing effective user és group runtime
  - `setuid, setgid`
  - `su, sudo`

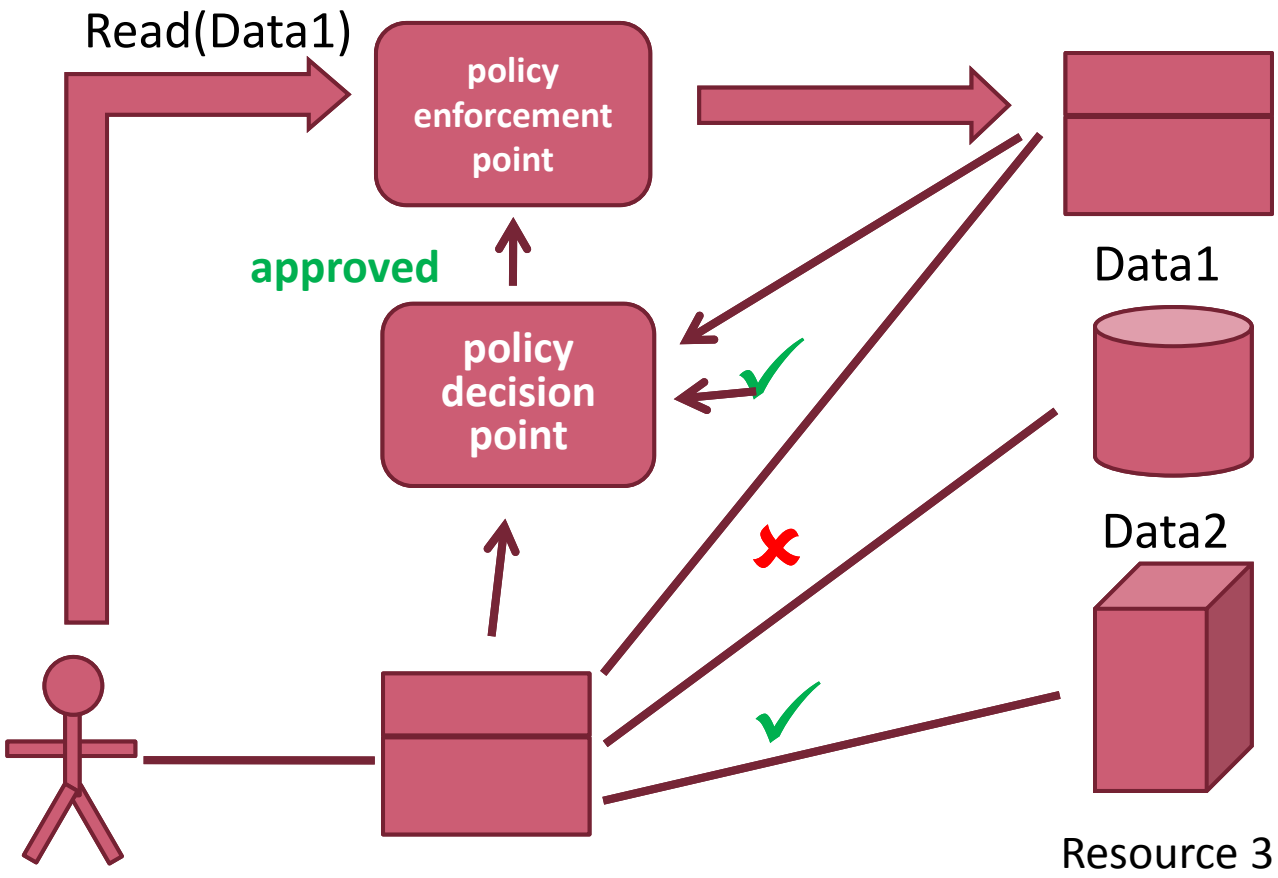
Content

- Short security introduction
- User management, authentication
  - UNIX, Linux
  - Windows
- **Authorization**
  - General methods
    - Role-based access control
    - Access control lists
  - Authorization on UNIX/Linux
  - Authorization on Windows

Authorization in general



Executing operations



General concepts

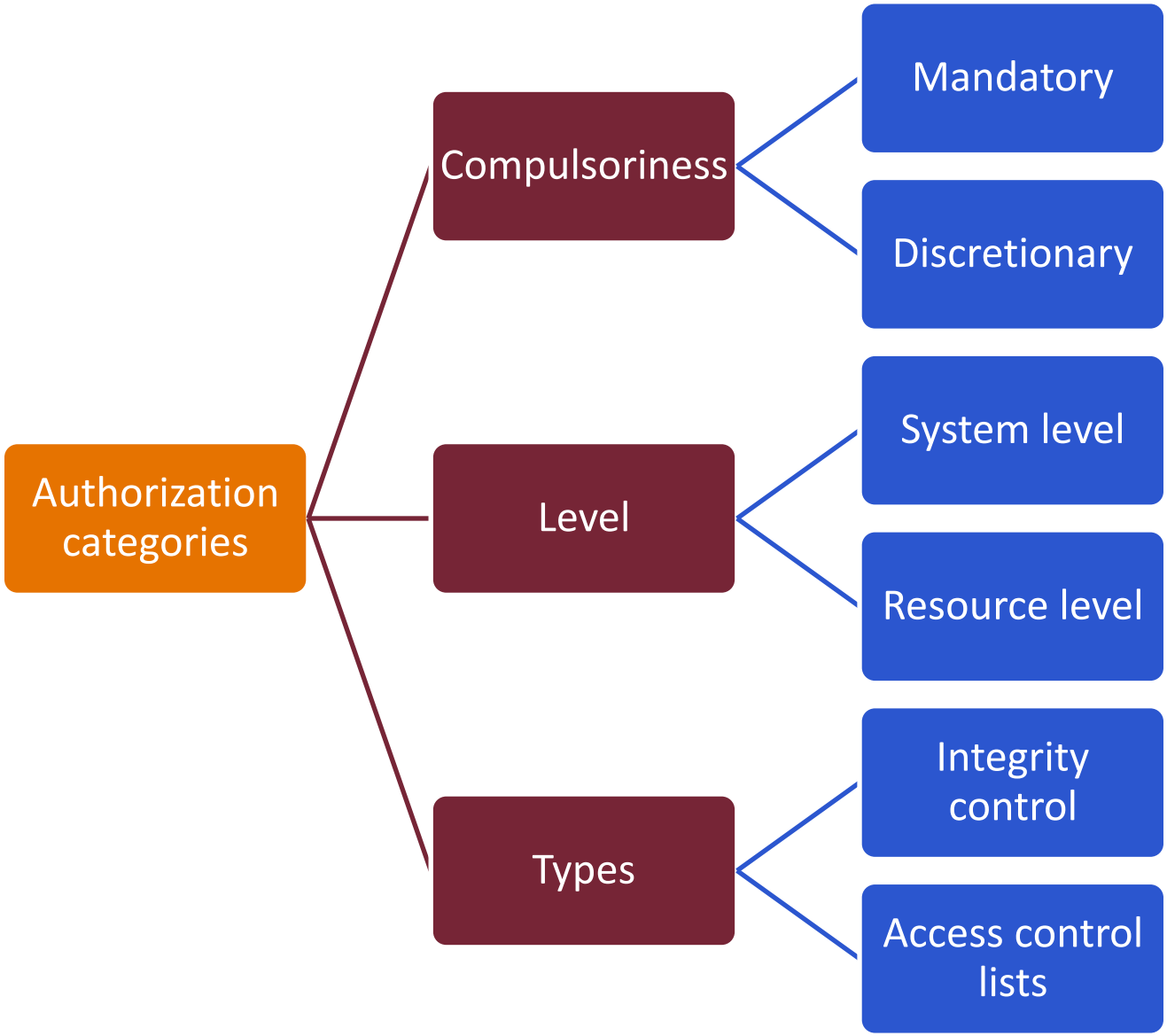
- Actors initiate *operations*
- The *context* of the operation includes the identifier of the actor, the protected object and the type of operation
- The policy *decision* component evaluates:
  - approves or denies the operation
- The policy *enforcement* component assures that the result is enforced

Challenges in authorization

- There are many actors in the system
  - Moreover: different systems identify the users differently
- There are many protected objects

- The whole relationship:
  - (Actors) X (Objects) X (Types of operation)
  - This is called *access matrix*
  - It is unmanageable, the whole matrix is huge!

Categorizing authorization methods



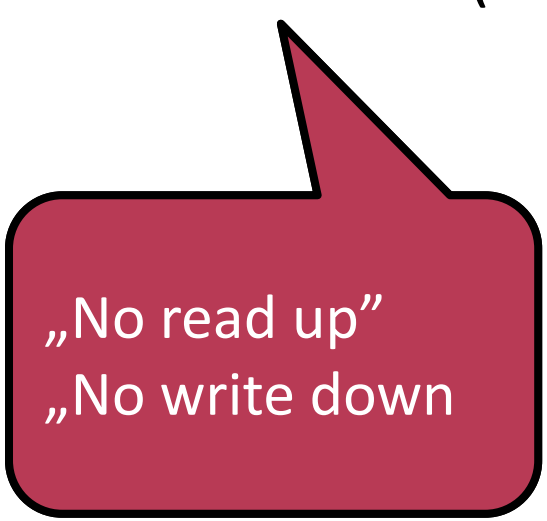
Category: Compulsoriness

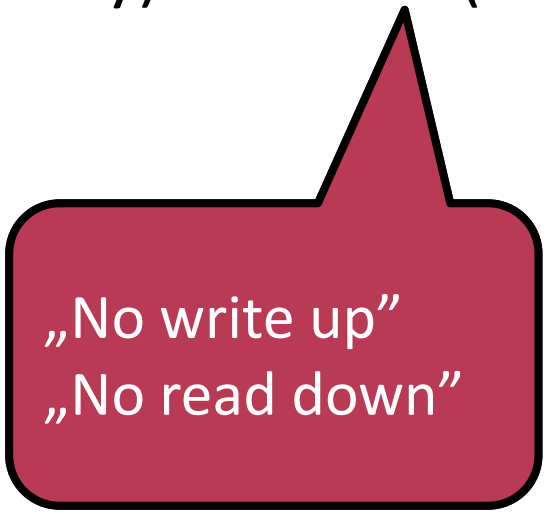
- Classical concepts (US DoD standard)

- Mandatory
  - security policy is managed centrally
  - users cannot change the policy
- Discretionary
  - the owner of the resource can change the permissions

Category: type

- Integrity control
  - Labeling objects
    - Integrity level: high – low, public – secret
  - Typical validation:
    - lower level actor cannot read a higher level object
  - Bell-LaPadula (confidentiality) and Biba (integrity)



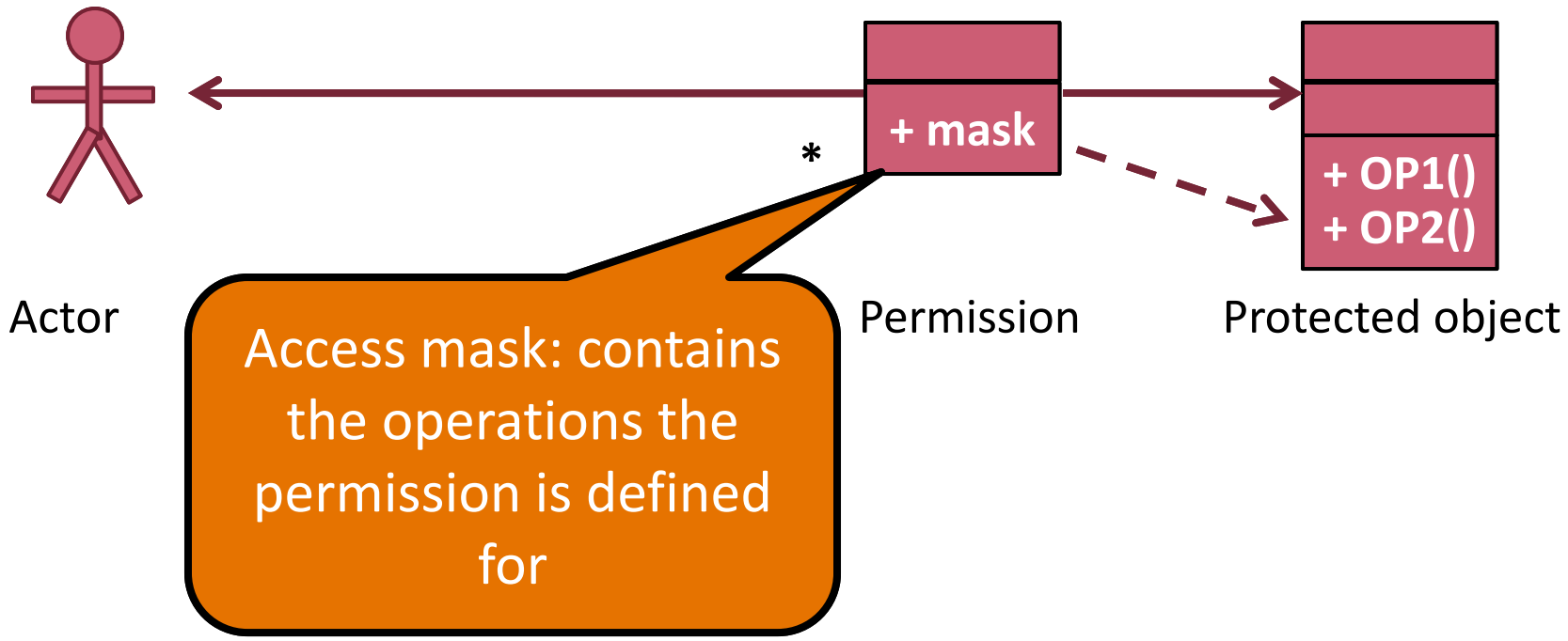


Category: type

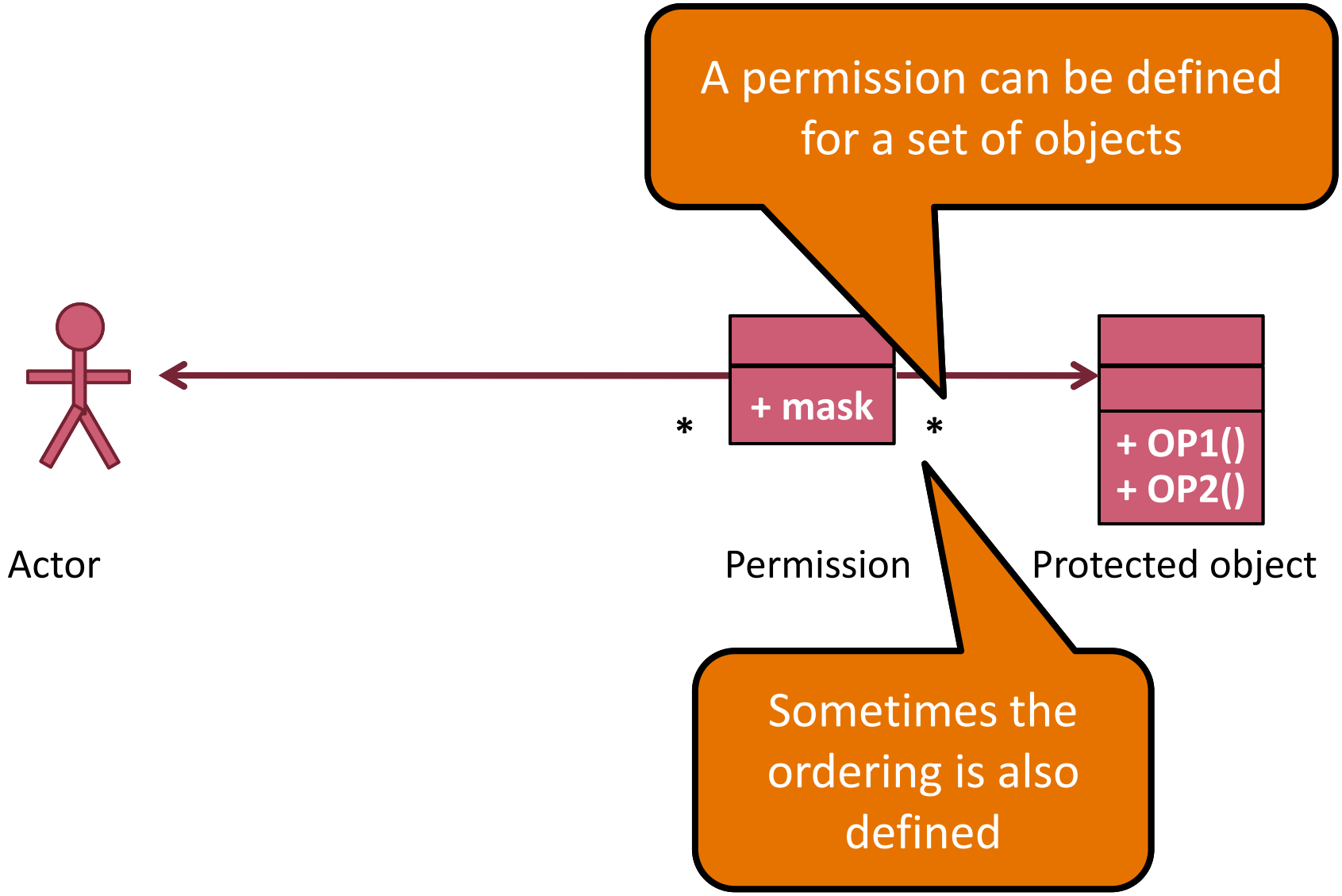
- Integrity control
  - Labeling objects
    - Integrity level: high – low, public – secret
  - Typical validation:
    - lower level actor cannot read a higher level object
  - Bell-LaPadula (confidentiality) and Biba (integrity)

- Access control lists
  - object ⟶ (actor, permissions)
    - Permission: read, write, execute…

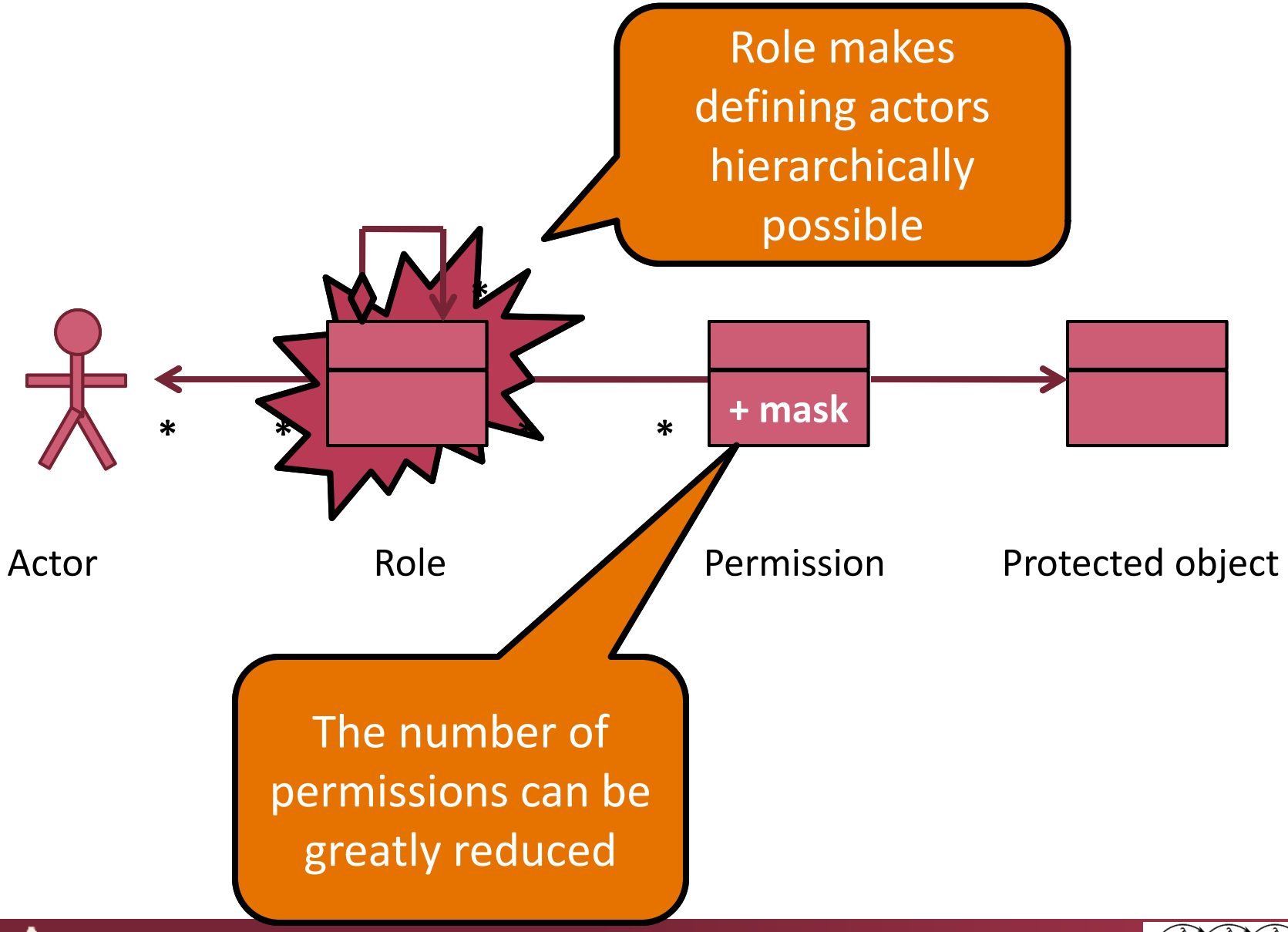
Access control lists



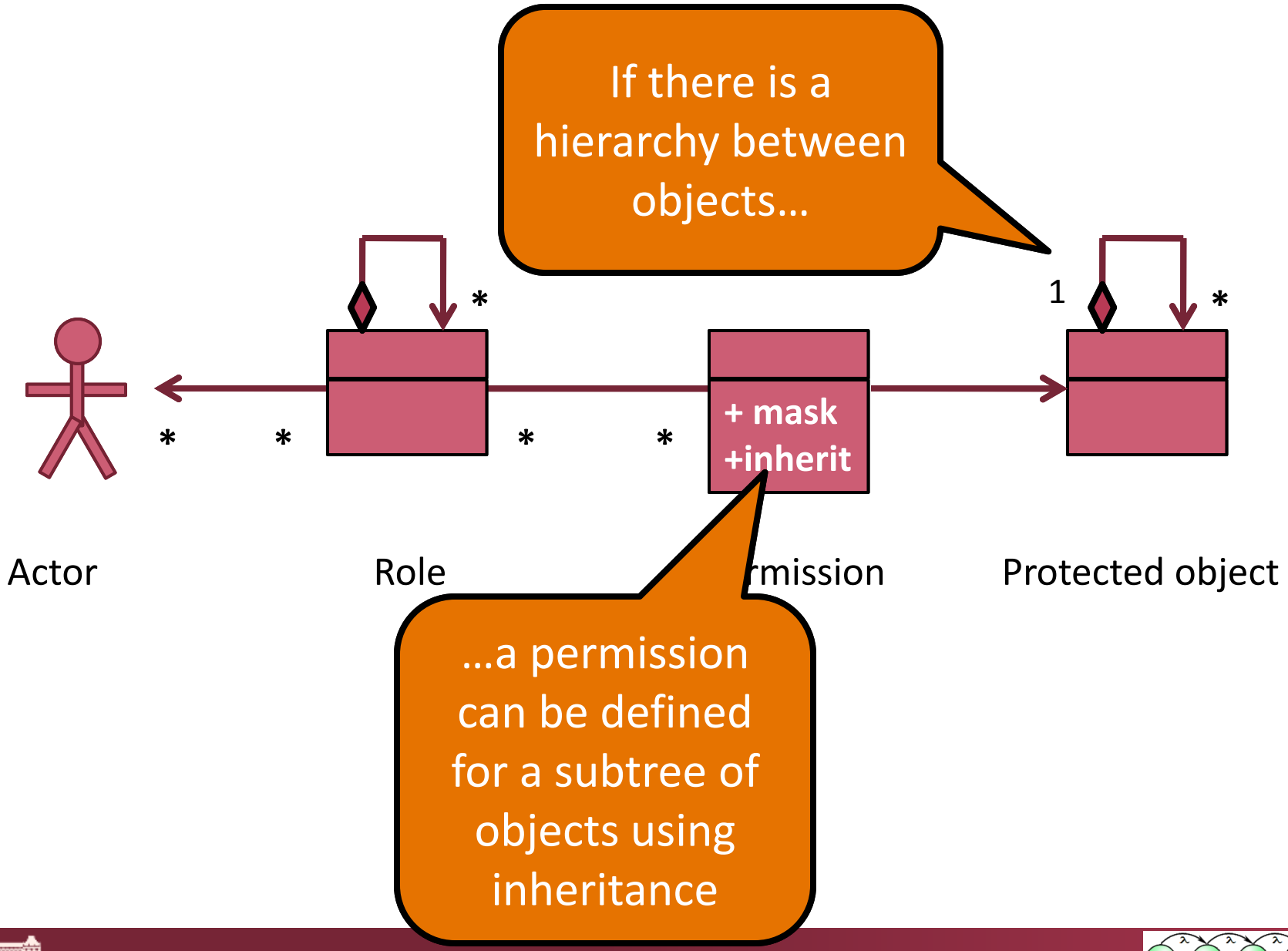
Access control lists



Role-based Access Control (RBAC)



Hierarchy between objects



Groups

User

memberOf

Group

+ Name
(+ Purpose…)
(+ Shared Secret)

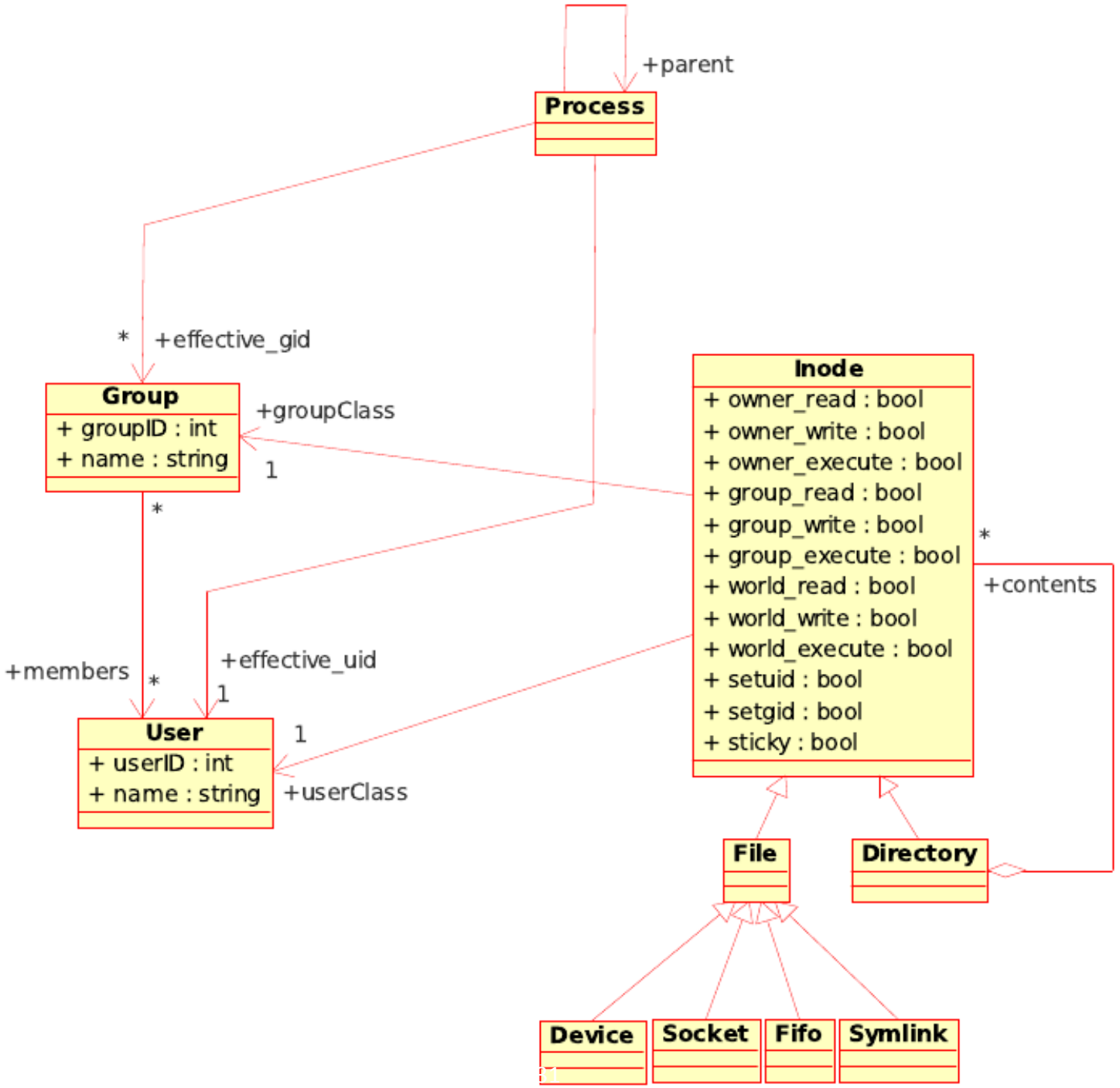
Group membership is a method for implementing RBAC

Content

- Short security introduction
- User management, authentication
  - UNIX, Linux
  - Windows
- Authorization
  - General methods
    - Role-based access control
    - Access control lists
  - **Authorization on UNIX/Linux**
  - Authorization on Windows

POSIX file system permissions

- Basic concepts
  - Acotr: `user`
  - Hierarchy of actors: `group`
  - A `user` can be member of several `group`
  - A `group` can contain several `user`
  - `Group` cannot contain an other `group`
- Permissions
  - 3x3 bit: read, write, execute (entering a directory)
    - First 3: for the owner of the object
    - Second 3: for the group of the object
    - Third 3: everyone else
  - Special bits:
    - setuid, setgid: when running changes the uid, gid to the owner
    - sticky: sets the owner of new objects


+parent
Process
*
+effective_gid
Group
+ groupID : int
+ name : string
+groupClass
1
*
+members
*
+effective_uid
1
User
+ userID : int
+ name : string
1
+userClass
Inode
+ owner_read : bool
+ owner_write : bool
+ owner_execute : bool
+ group_read : bool
+ group_write : bool
+ group_execute : bool
+ world_read : bool
+ world_write : bool
+ world_execute : bool
+ setuid : bool
+ setgid : bool
+ sticky : bool
*
+contents
File
Directory
Device
Socket
Fifo
Symlink

DEMO Linux permissions

- Changing owner: `chown`
  - can be executed only by the root
- Changing permissions: `chmod`
  - Only allowed to the owner of the object
  - Several styles for permissions:
    - 4 octal numbers
    - Changing e.g.: u+x (add execute for user), g-w (remove write for group)
- Listing:
  - `ls -l`
  - `ls -l -n`

Other privileges

- Root has special privileges:
  - Can set real-time class scheduling
  - Can access I/O devices directly (!)
  - Can listen on TCP ports below 1024
  - Can change kernel parameters, load kernel module, etc.
  - …
- But this also should be modifiable
  - Principle of least privileges
  - Method: POSIX Capabilities (method for assigning global system-level privileges)