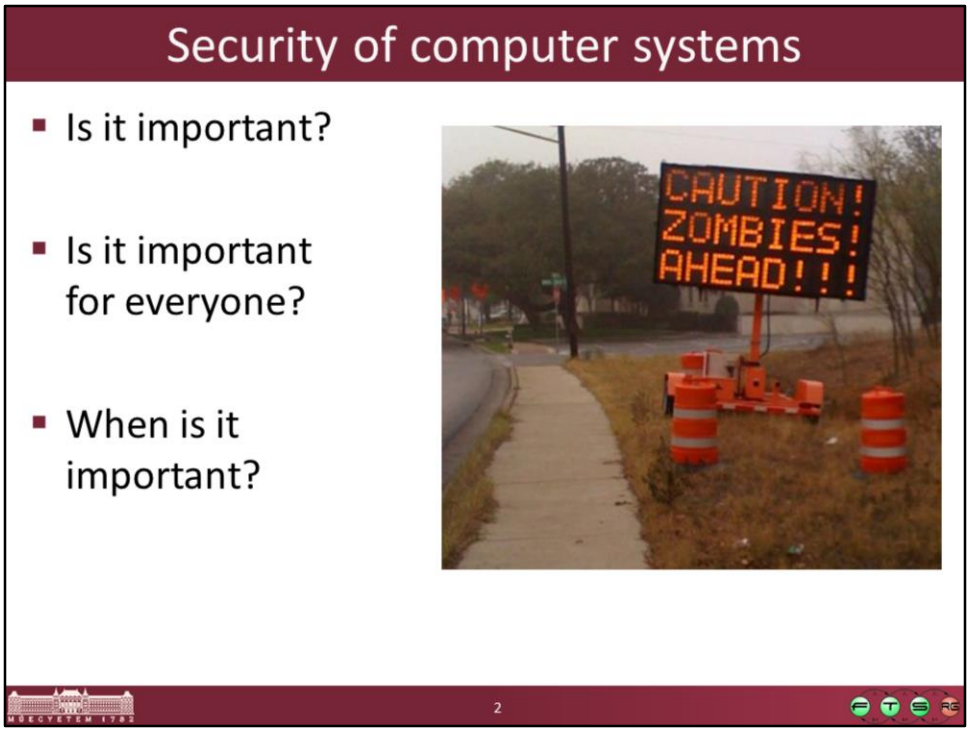
# Authentication and authorization

Dániel Tóth, Zoltán Micskei
Benedek Horváth

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék



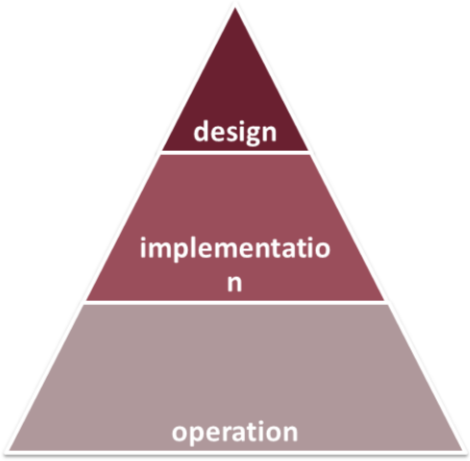

Általános kezdés:
• Nyilvánvaló, hogy banki, üzleti szférában fontos a biztonság, de máshol?
• Otthoni gépen? Személyes adatok megszerezhetőek stb. vissza lehet élni vele -> igen tényleg fontos.
• Beágyazott, zárt rendszerekben minek foglalkozni vele?! – itt jön a zombis kép

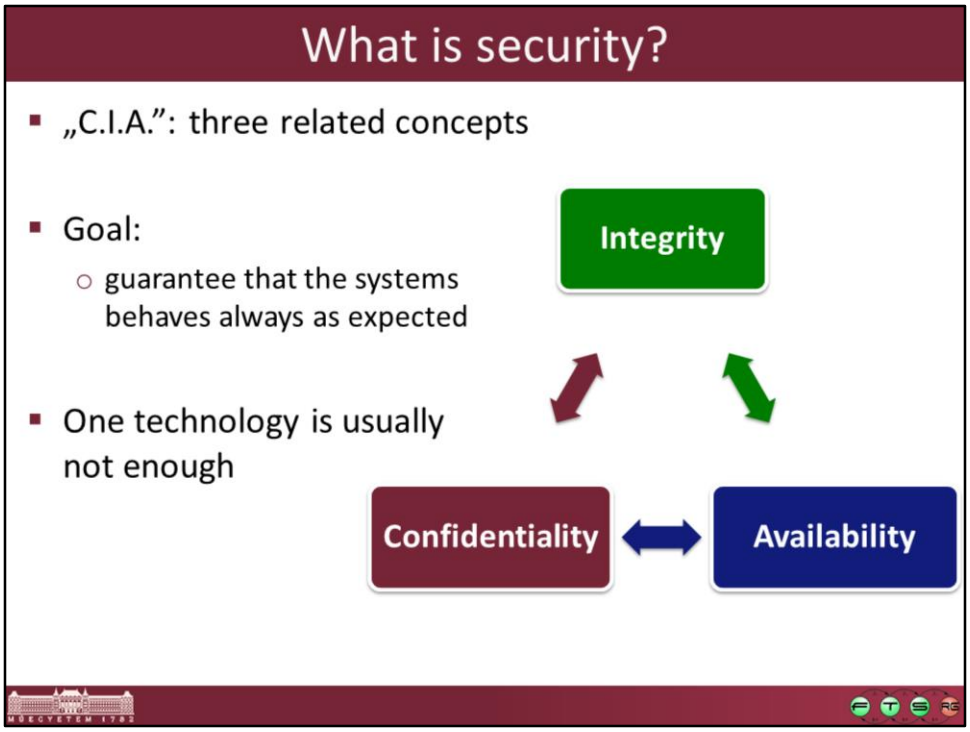
# (appended picture)

– hát azért, mert a rendszerek sosem tekinthetőek teljesen „zártnak”, úgyis megpróbálnak majd belepiszkálni. Erre veszélyesebb példa a lengyel srác esete, aki a Lodz-i villamossín váltókhoz **(tram rail switch in Lodz)** készített távirányítót és kisiklatott néhány villamost **(trams were derailed)**. **Biztonságosnak tekinthető-e egy tűzfallal védett belső hálózat? Milyen buktatói lehetnek?**
• **Egy képszerkesztőben, médialejátszó vagy dokumentum néző alkalmazásban fontos-e a biztonság?** Meglepő módon igen, mert ellenkező esetben a felhasználó által elvárttól eltérő viselkedés kényszeríthető ki belőle, egy szándékosan hibásan formázott bementtel.

# 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.

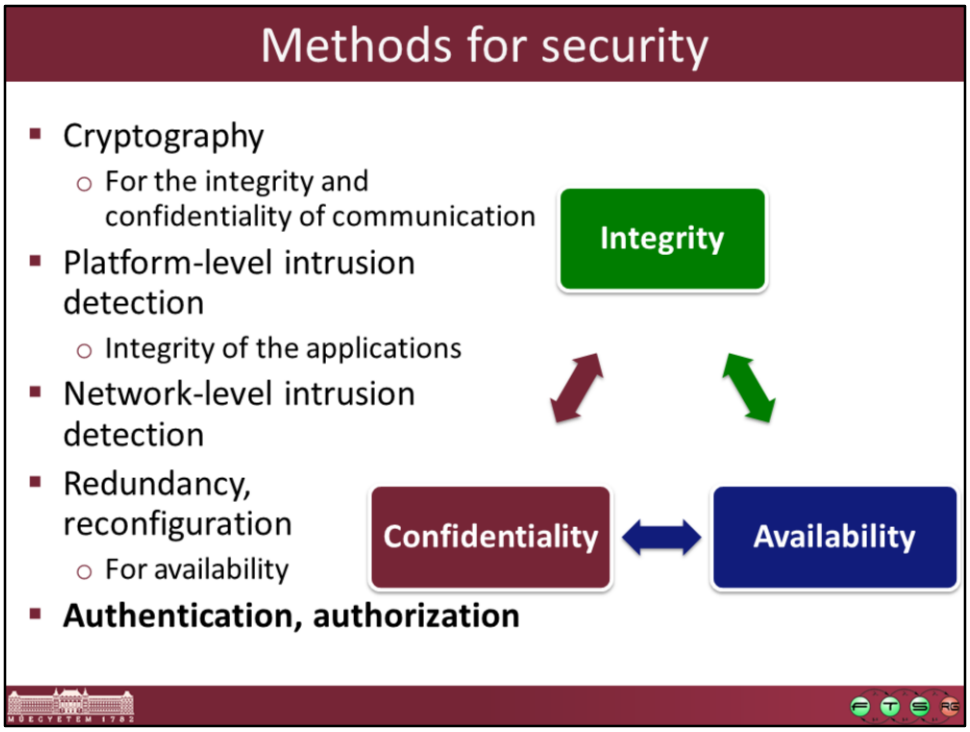





(Fontos általános megjegyzés: most a security világ szemszögéből tekintjük át, a dependability-s világ más elnevezéseket és csoportosítást használ)

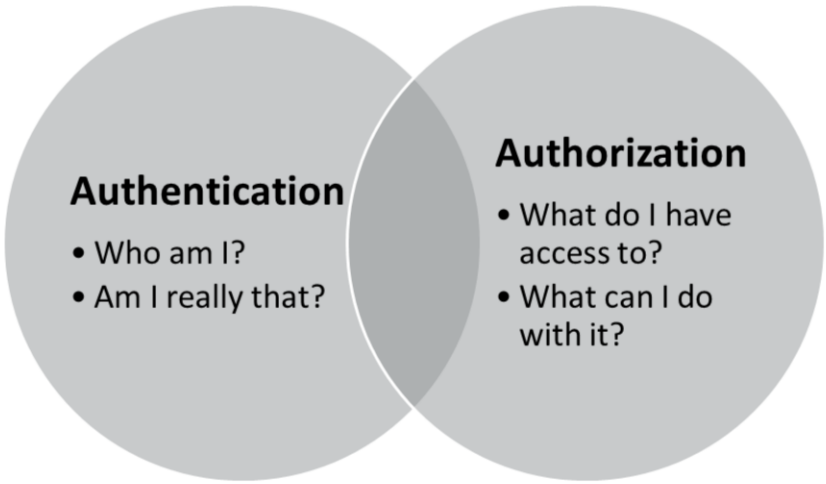
•*Bizalmasság: titoktartás harmadik féllel szemben, először mindenkinek ez jut eszébe a számítógépes biztonságról*
•*Sértetlenség: legalább olyan fontos, hogy az üzenetek feladója tényleg az legyen, aki a feladó mezőben szerepel, illetve a kapott üzenet tartalma pontosan az legyen, amit az (igazi) feladó küldeni akart. (Hitelesség, Authenticity). Néha külön kiemelik a letagadhatatlanságot (Non-deniability), vagy kifejezetten a letagadhatóságot. Továbbá a rendszer elemei is megmaradnak olyannak, amilyennek elvárjuk, rongálástól, illetéktelen módosítástól mentesnek. (Tamper resistance).*
•*Rendelkezésre állás: gyakran elfeledett része a biztonságnak, kárt lehet okozni azzal is, ha csak működésképtelenné válik egy rendszer (Denial of Service), különösen fontos „biztonságkritikus” rendszerekben, ahol emberélet függhet a rendelkezésre állástól.*

A három tulajdonságot külön-külön kell vizsgálni, nem feltétlen következménye egyik a másiknak (pl. egy digitális aláírással ellátott levél garantálhatja a sértetlenséget, de ha nincs titkosítva, akkor nem garantálja a bizalmasságot).




• Kriptográfia: Szükséges eszköz a sértetlenség és bizalmasság garantálásához, de önmagában nem elég.
• Platform szintű behatolás elleni védelem: memóriavédelem, futási szintek, NX bit, stack ill. heap randomizáció, rendszerhívás monitorozás és korlátozás – olyan megoldások, amik hibás alkalmazások esetén próbálják behatárolni egy támadás hatásait, OS illetve futtató hardver szintjén
• Hálózati behatolás elleni védelem: cím alapú szűrések, tűzfalak, forgalom megfigyelés, rögzítés, gyanús forgalom beazonosítása – mindhárom biztonsági cél esetén szükség lehet rájuk, támadás hatásait hálózat szintjén próbálja behatárolni
• Redundancia, újrakonfigurálás: véletlen meghibásodás vagy támadás esetén a működőképesség fenntartása vagy helyreállítása. A hibatűrési mechanizmusok sértetlenséget (Integrity) tételeznek fel.
• Hitelesítés, engedélyezés: erről fog szólni a most következő előadás

# 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



Operációs rendszer esetén protokollnak tekinthető az API, a folyamatok azonosítása, hozzátartozó engedélyeinek kezelése. Tágabb értelemben véve az OS szolgáltatása lehet, hogy hitelesítéssel garantálja a rá telepített programok sértetlen állapotát (aláírások ellenőrzése), OS szintű szolgáltatás lehet még a hálózati kapcsolatok hitelesítése, bizalmasságának garantálása is (pl. SSL könyvtár). Ez értelmezés kérdése.

# 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…

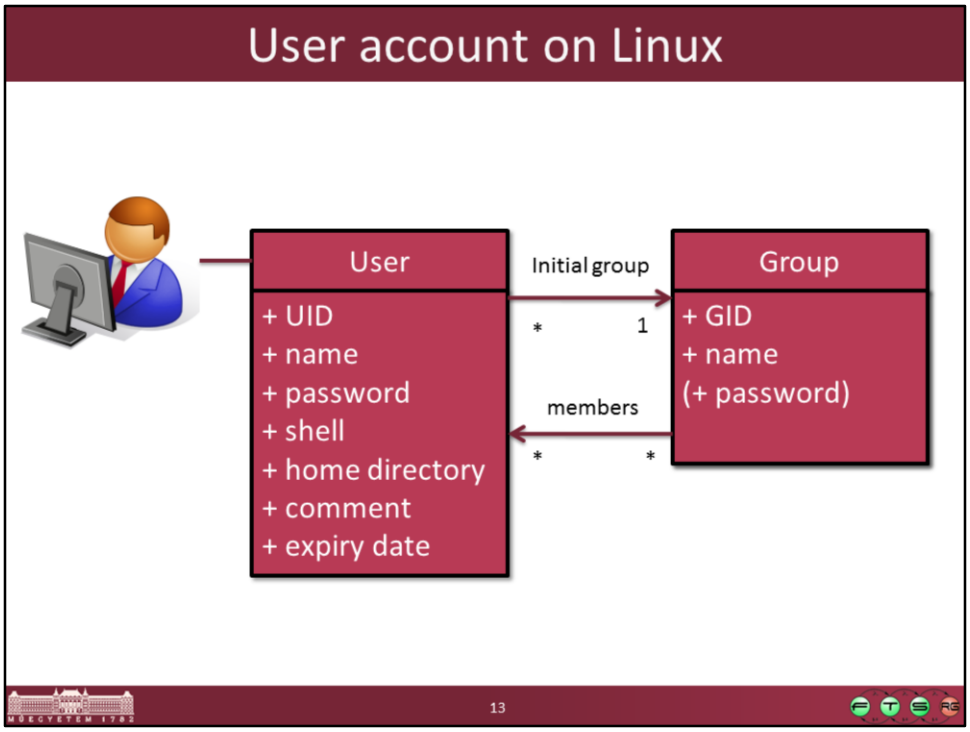
# What is a user account?

- 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`

+ Egy kis ismerkedés a man oldalakkal!




-Meglehetősen ritkán használt funkció a csoporthoz hozzárendelt jelszó, de van ilyen is.

## DEMO Process identity

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

# Authentication on Linux

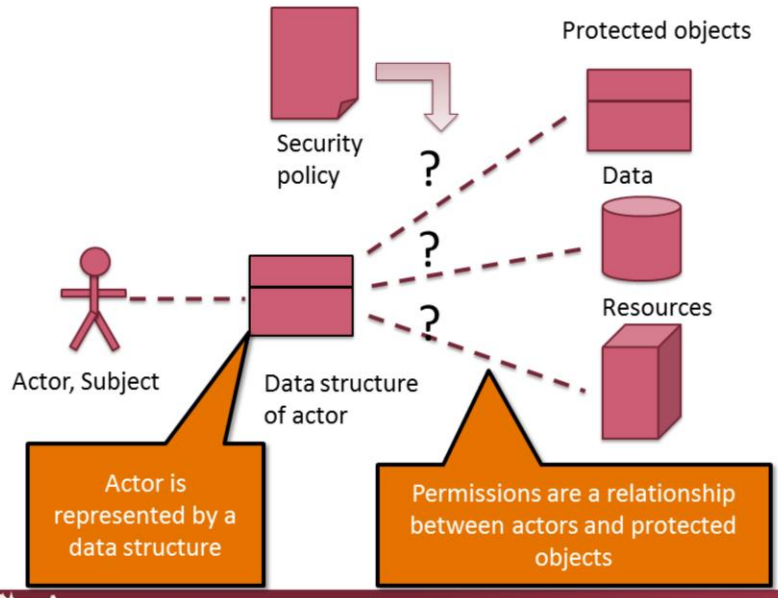
- Between machines (e.g. ssh)
  - User name / password
  - Private / public key
  - Individuality?

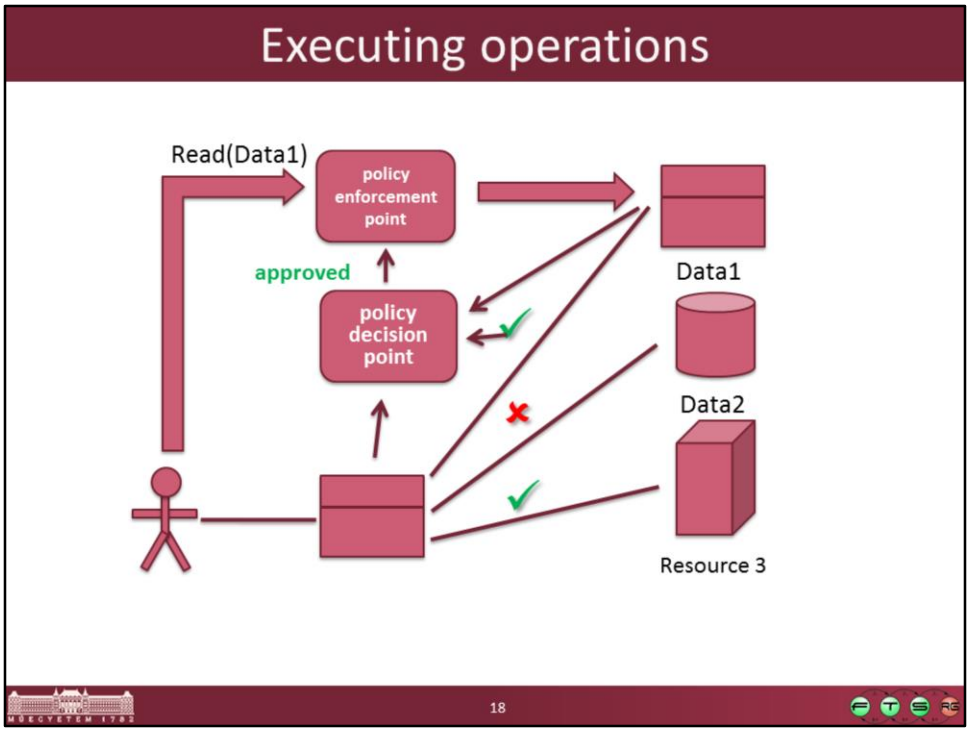

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAxNZqUyFFFoGpDt5GPn
6wxABB/nbRynaBMqy8d1Y3eW3Z
+bBucyqg6Vx1ez/Lzirm76wcIeAj6DR99wnUUumrFcUuTR
D6ejDNe8xqrMuF9RJ2+tmkCKzmiTHqyAQuh8kLZYl7psFz
vqiYkuNEVomUwkjd/GSknJ3TDtVykLDMg3wrQ
+eNd8QqI2LjUCTTg0EgovJf8MECrvrX3cKFeaOMwAhkqhU
oEbCDmxVwXgSRxWdhYCPptFB0sxrdBKpOv5REroAFgItho
qwvdPpEuobg2tBlsjqHzyqRkLbOPVjTgNHd4Hn3fV4pNwi
uIWMPpfrO7feAI9Xi8NwhaJ8f66xR2w==
```

# 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





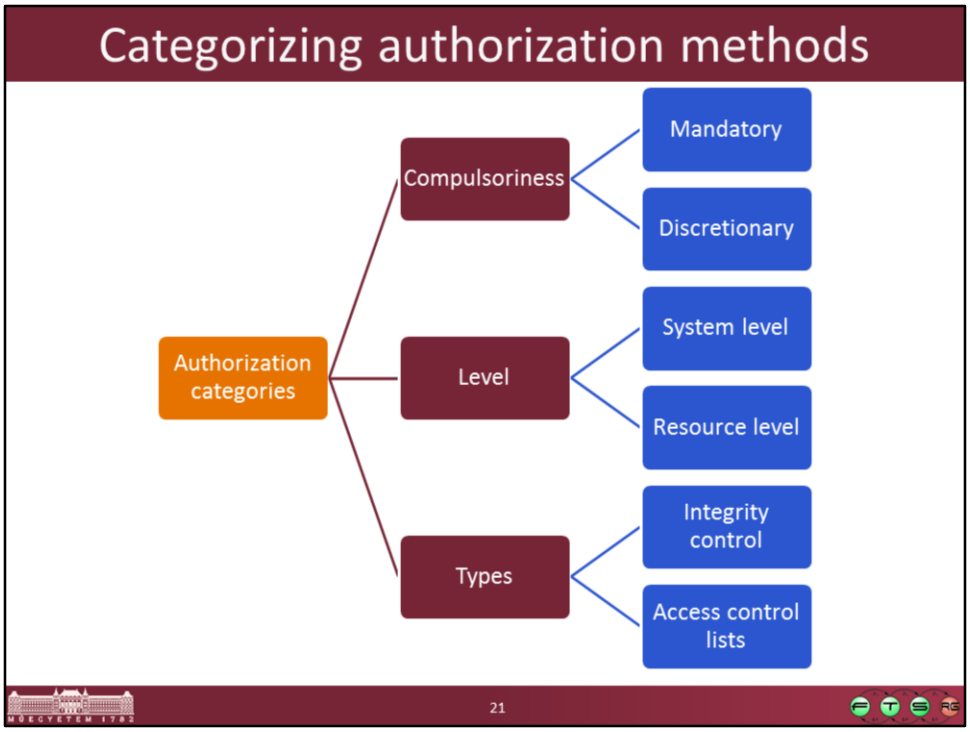

• A szereplők *műveleteket kezdeményeznek*
• A műveletek *kontextusa tartalmazza a szereplő azonosítóját, a célobjektumot és az elvégzendő művelet fajtáját*
• A jogosultsági *döntő komponens kiértékeli a kontextust, engedélyezi vagy megtiltja a műveletet*
• A jogosultsági *végrehajtó komponens biztosítja, hogy a döntő által hozott döntés érvényre jusson*

# 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!




A csoportosítás esetleges, rengeteg egyéb szempont van természetesen.

# 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)

„No read up"
„No write down"

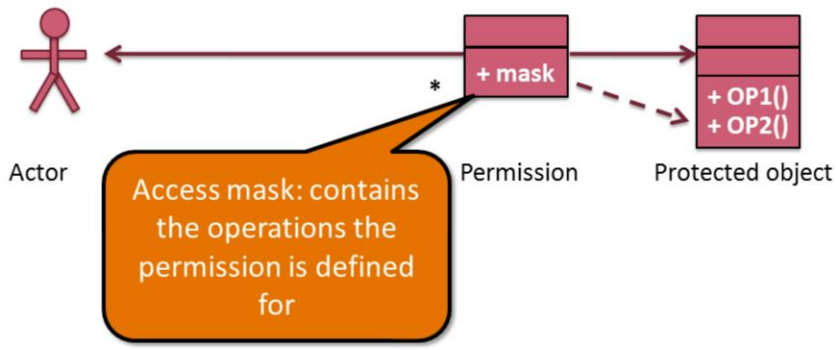
„No write up"
„No read down"

# Category: type

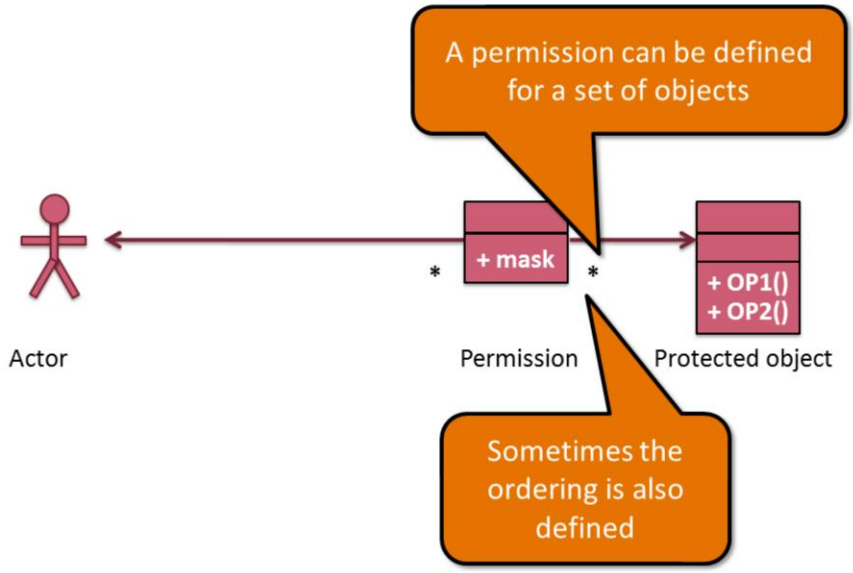
- Integrity control
  - Labeling objects
    - Integrity level: high – low, public – secret
  - Typical validation:
    - lower level actor cannot read a higher level object
  - Bell-LaPadula (confidentiality) and Biba (integrity)

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

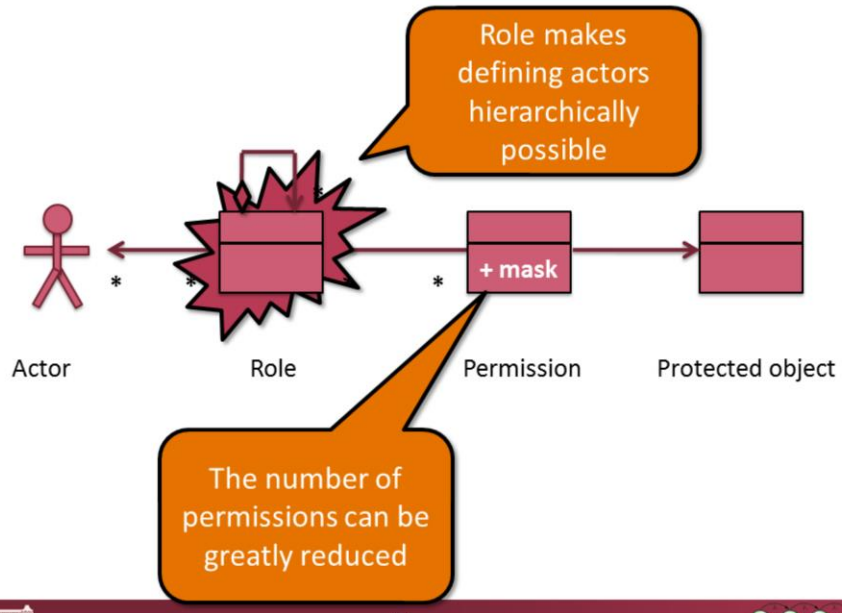
# Access control lists



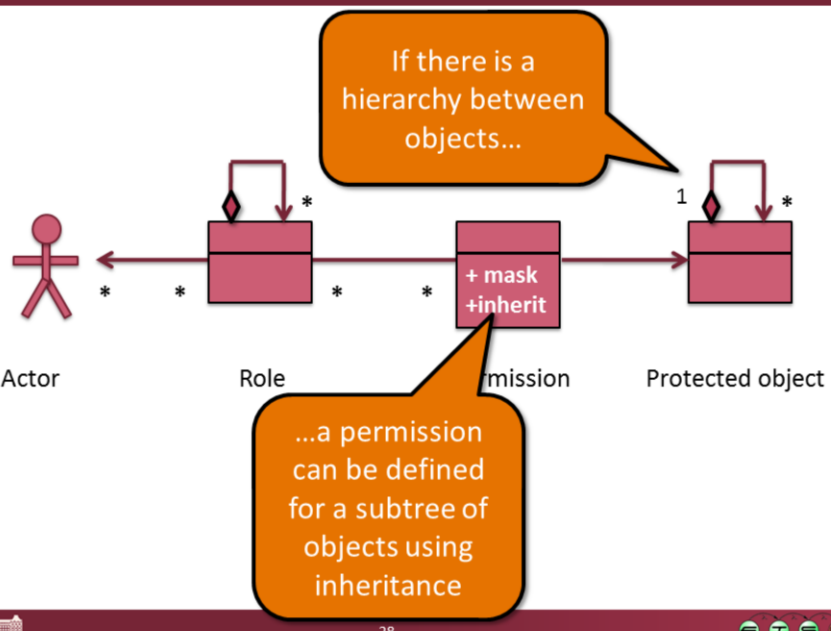
# Access control lists



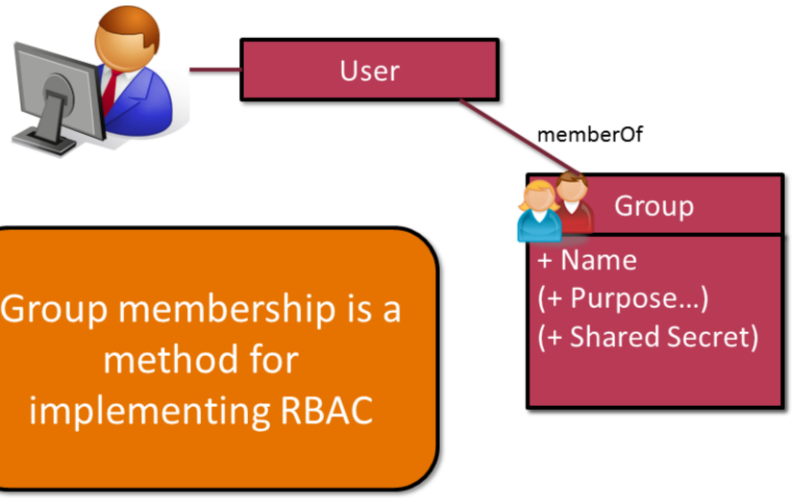
# Role-based Access Control (RBAC)



# 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
  - Actor: `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

## 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)



Capability: /usr/src/linux/include/linux/capability.h