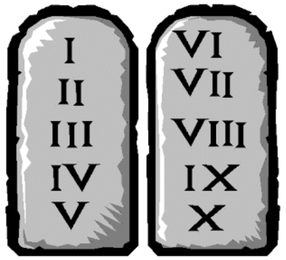
Fundamentos de Segurança Informática (FSI)

2024/2025 - LEIC

Systems Security (Part 1)

Hugo Pacheco
hpacheco@fc.up.pt

Fundamental Principles



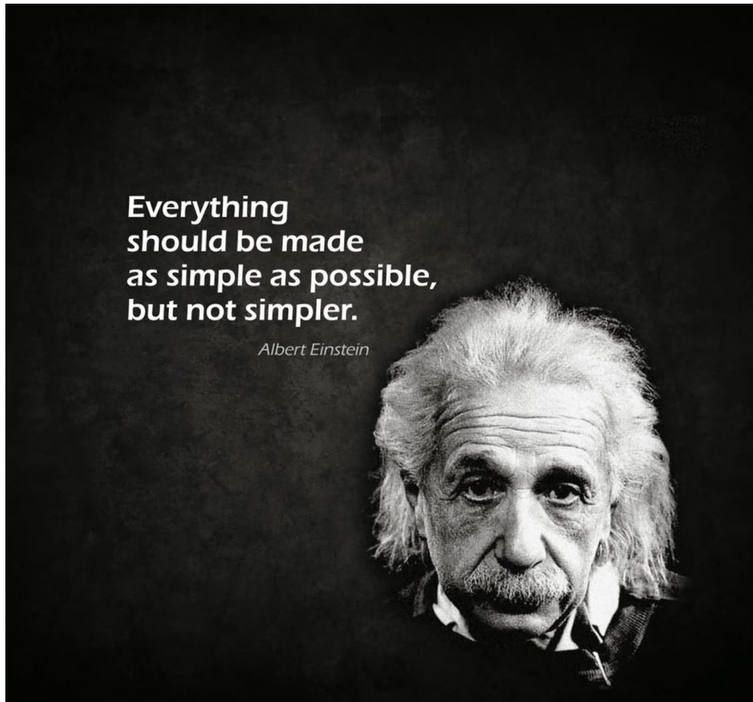
- When considering security of complex systems, the following principles must always be present [Saltzer, Schroeder 75]

- All remain valid, even for **computer security**, but some **less technical**

  - ***Economy of mechanism***
  - ***Fail-safe defaults***
  - ***Complete mediation***
  - ***Open design***
  - ***Separation of privilege***
  - ***Least privilege***
  - ***Least common mechanism***
  - ***Psychological acceptability***
  - ***Work factor***
  - ***Compromise recording***

Economy of Mechanism

- Keep it simple:
  - A system shall have **only the necessary functionalities** (e.g., running services)
    ⇒ avoid “nice to have” functionalities
  - The **simplest security mechanisms** shall be adopted
- Eases implementation, usability, **validation**, etc.



Fail-safe Defaults 

- The configuration of any system shall, by **default**, enforce a **conservative level** of protection

- E.g., a new user must, by omission, have minimum permissions
- ***“Fail closed”*: on failure, the system reverts to its conservative lock**
- ***“Fail open”*: on failure, the system recovers to its permissive state**

Railway semaphore signals. "Stop" or "caution" is a horizontal arm, "Clear to Proceed" is 45 degrees upwards, so failure of the actuating cable releases the signal arm to safety under gravity.

FortiGate VPN Default Config Allows MitM Attacks

The client's default configuration for SSL-VPN has a certificate issue, researchers said.

https://threatpost.com/fortigate-vpn-default-config-mitm-attacks/159586/

Default Key Algorithm In Thomson And BT Home Hub Routers

Mon, 14 Apr 2008 08:00:33 GMT
by pagvac

Yes, we're back with more embedded devices vulnerability research! And yes, we're also back with more security attacks against the BT Home Hub (most popular DSL router in the UK)!

Confirmed suspicions

Many of us involved researching the security of wireless home routers have always suspected that routers that come with default WEP/WPA keys follow predictable algorithms for practical reasons. Yes, I'm talking about routers that come with those stickers that include info such as S/N, default SSID, and default WEP/WPA key. Chances are that if you own a wireless router which uses a default WEP or WPA key, such key can be predicted based on publicly-available information such as the router's MAC address or SSID. In other words: it's quite likely that the bad guys can break into your network if you're using the default encryption key. Thanks to Kevin, our suspicion that such issue exists on the BT Home Hub has been confirmed (keep reading for more details!). Our advice is: **use WPA rather than WEP and change the default encryption key now!**

https://www.gnucitizen.org/blog/default-key-algorithm-in-thomson-and-bt-home-hub-routers/

Open Design 

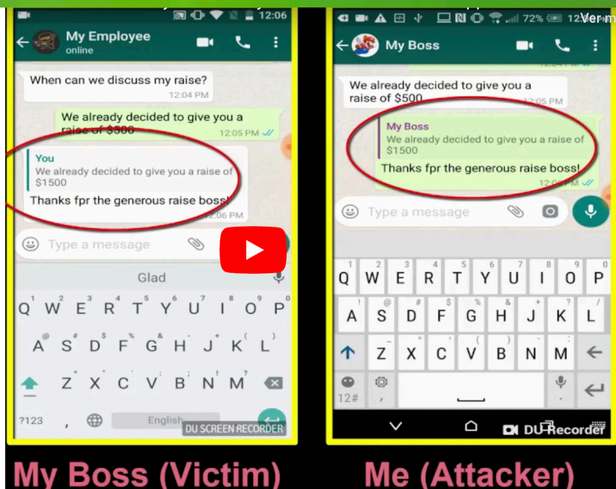
- The security architecture and the inner working of a system **shall be public**:
  - The security argument should not be based on hiding the security mechanisms
- Secrets are (a small number of) configurable system parameters:
  - cryptographic keys, passwords, etc.
- **Rationale:**
  - Allow scrutiny ⇒ More likely to find out if and where we are vulnerable
  - Secrets can be adjusted without changing the system

WhatsApp vs Signal

- Proprietary (WhatsApp) vs Open Source (Signal)
  - e.g., WhatsApp VOIP vulnerability, used to inject malware by NSO/Pegasus
  - e.g., vulnerability that allowed impersonation attacks: malicious users could “repost” as if they posted from trustworthy sources
- 2014: WhatsApp ⇒ Signal E2EE Protocol

WhatsApp urges users to update app after discovering spyware vulnerability

The spyware, developed by Israeli cyber intelligence company, used infected phone calls to take over the functions of operating systems



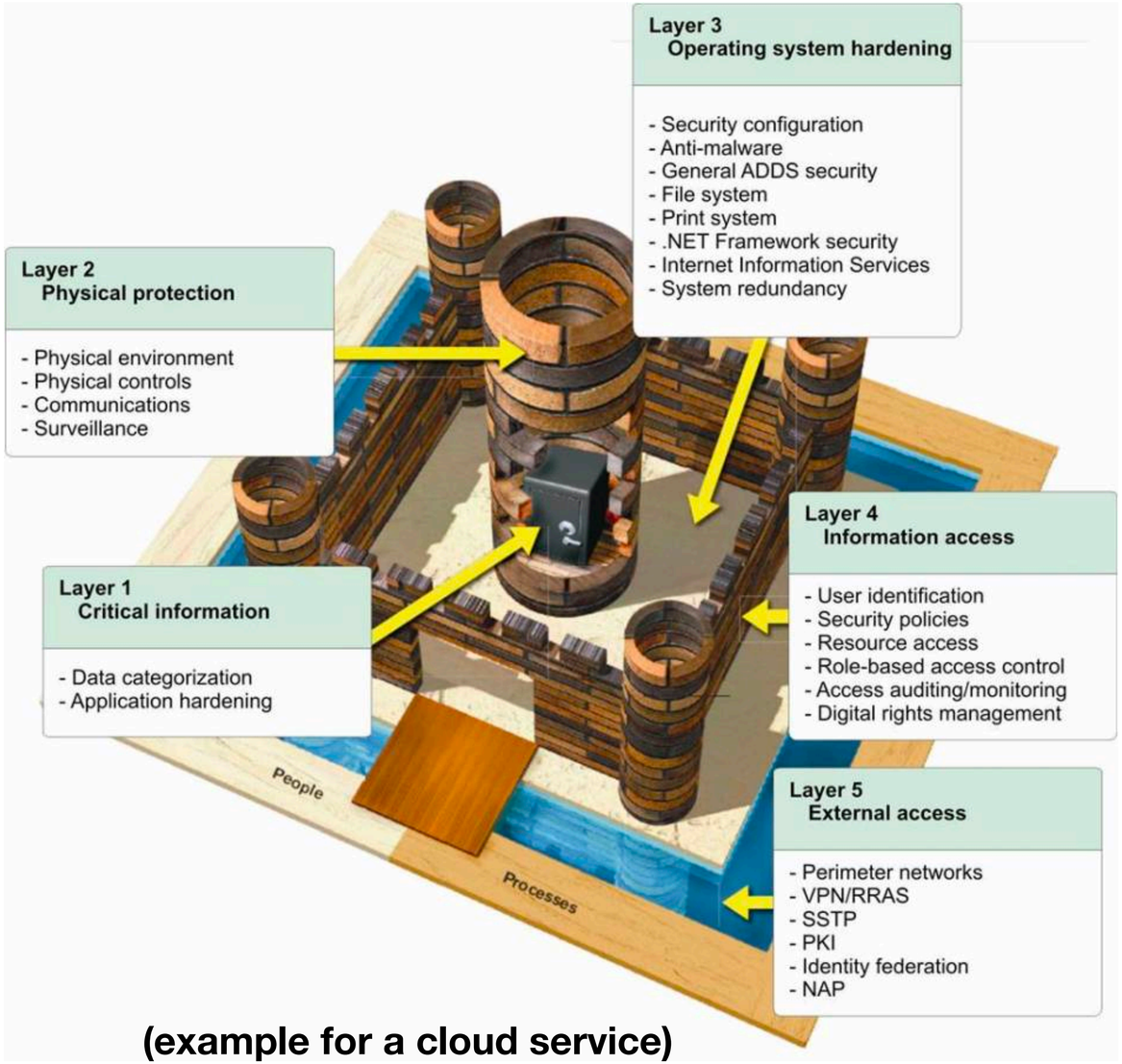
https://www.theguardian.com/technology/2019/may/13/whatsapp-urges-users-to-upgrade-after-discovering-spyware-vulnerability

https://research.checkpoint.com/2019/black-hat-2019-whatsapp-protocol-decryption-for-chat-manipulation-and-more/

Security through Obscurity

Defense in Depth

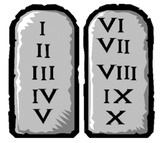
- Also known as the “Castle Approach”
- We have seen before that all security mechanisms can fail
- We **cannot deposit all trust in one single mechanism**
  - e.g., blinding the perimeter but assuming there exist no internal threats



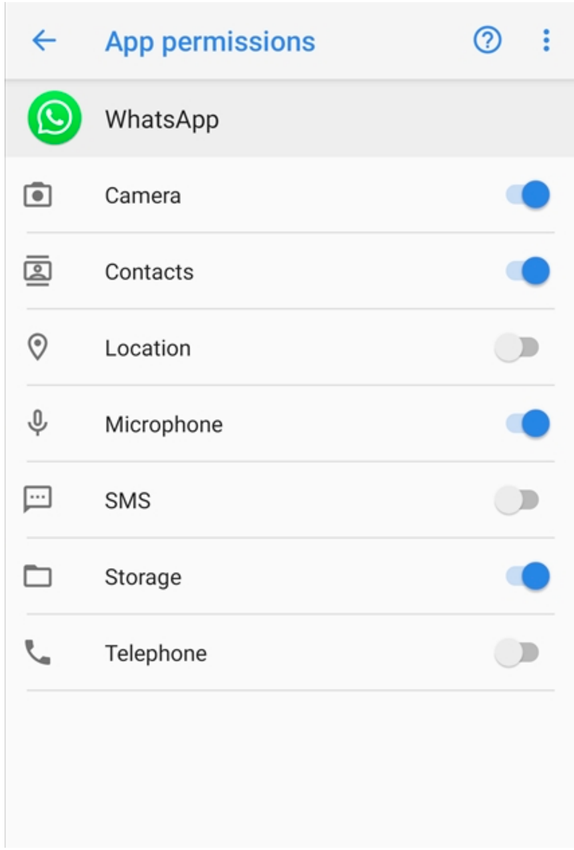
(example for a cloud service)

Least Common Mechanism

- Sharing of resources of access mechanisms shall be minimised, especially when users have different levels of privilege
- Examples:
  - don't reuse passwords across accounts in different services
  - store each user’s data on a different file system
  - a web server and a database server should not share interfaces

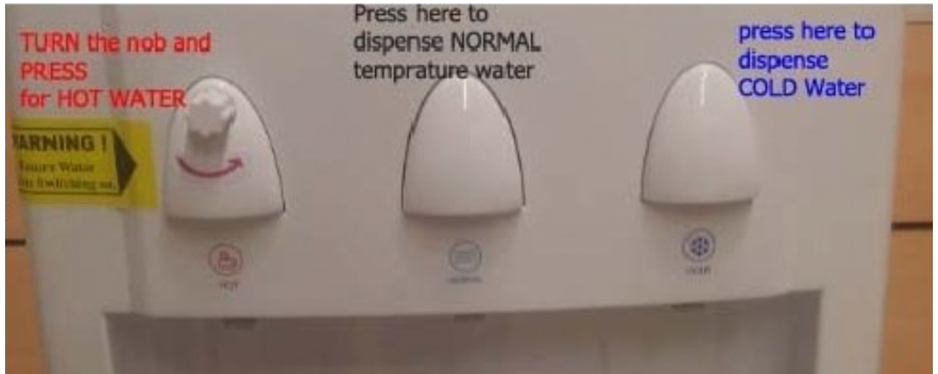
Least Privilege 

- Each user, compartment, program, etc.
  - Shall have only the **least privileges/permissions** to accomplish its role
  - E.g., Android/iOS, Linux
  - **Rationale:**
    - Contradicting this principle unnecessarily amplifies the potential impact of a security breach
  - **Problem:** running all services as root?



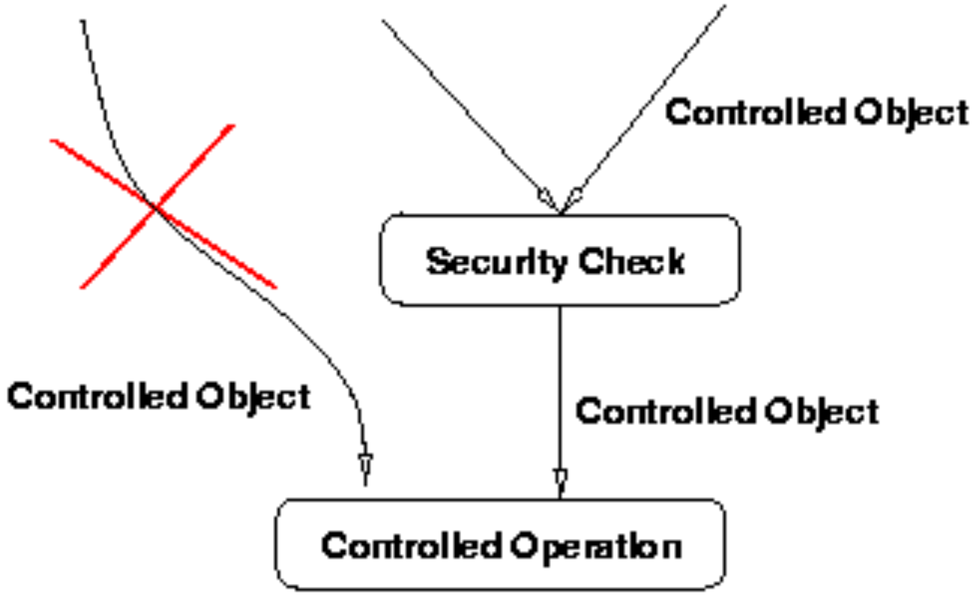
Separation of Privilege 

- Functionalities and resource usage **shall be compartmentalised**:
  - Each compartment must be isolated from others, in a separate trust domain
- Combined with Defense in Depth  , Least Privilege  and Least Common Mechanism  :
  - Compromising a compartment shall have localised consequences
  - A compartment shall only have access to its necessary resources



Complete Mediation

- A system manages resources: files, HW devices, memory, etc
- For all resources:
  1. **Define** a clear **security policy**
  2. **Validate all accesses** to the resource w.r.t. the security policy
- E.g. **Web** = IDOR (*https://site.com/account?id=123)*
- E.g. **Linux** = virtual memory is physically shared
  - All process accesses to memory are mediated by the OS
- We will see various forms of access control = mediation



Quiz 

- Installing a computer virus when opening a mail attachment is a **violation** of which security principle?
- Least Privilege 🕎: the virus runs with the user's permissions?
- Defense in Depth 🕎: no antivirus check?
- Economy of Mechanism 🕎: do we really need executable attachments?

Quiz 

- Using OAuth 2.0 for distributed authorisation **adheres** to which security principles?
- Separation of Privilege 🕎: various types of authorisation roles
- Least Common Mechanism 🕎: user authorisation is delegated to the entity that registers its account

Access Control

Access Control

- **Access control** denotes a family of security mechanisms that instantiate some of the previous principles:
  - **Least Privilege**  : assign strictly necessary privileges to the entities that interact with the system
  - **Complete Mediation**  : guarantee that all accesses to a resource are performed by entities with adequate privileges
  - **Separation of Privilege**  **:** related to the previous two

Access Control

- Access control encompasses:
  - **Actor** (entity that performs an action)
  - **Resource** (on which the action is performed)
  - **Operation** (the concrete action that is performed)
- A pair (**resource**,**operation**) is often called a **permission**
- UNIX: A **process** shall not be able **read/write memory** from other processes
- Mobile: An **app** shall only be able to **edit** its own **data**
- Web: A **domain** shall not be able to **read** the **cookies** from other domains

Access Control Matrix

- Describes all possible accesses:
  - **Actor**, **Resource**, **Operation**
  - E.g., read, write, execute
- Advantages: clarity, effectiveness
- Problem: number of combinations
- Centralised? How does it scale for large-scale systems?

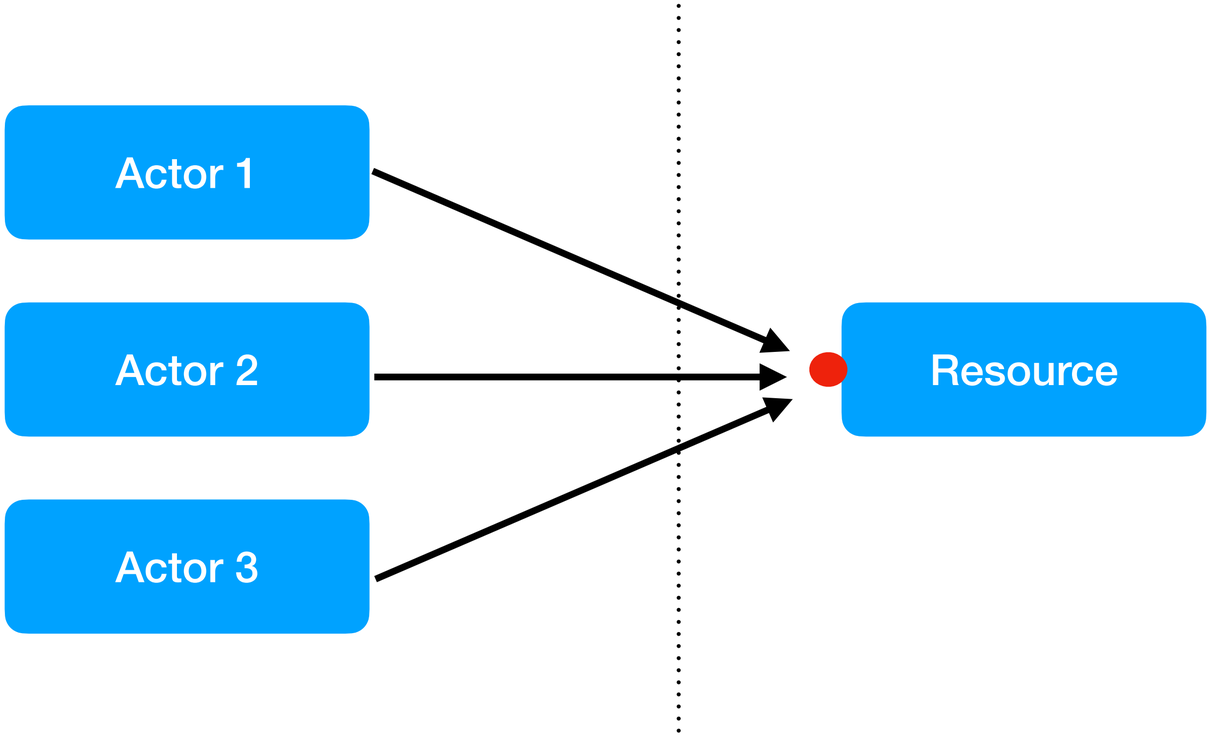
	R1	R2	R3
A1	r	rw	n
A2	rw	n	r
A3	r	r	r

Access Control Lists (ACL)

- For each **resource**, all **permissions** of **actors** on them (omission = no permission, intuition = guest lists)
- **Advantages:**
  - **Locality: permissions** of each **resource** stored individually next to the resource
  - **Isolation:** a **resource** can only be accessed by a limited number of **actors**
- **Disadvantages:**
  - Not possible to determine the **permissions** of an **actor** without checking all **resources** (e.g., to remove the actor)

	R1	R2	R3
A1	r	rw	n
A2	rw	n	r
A3	r	r	r

Access Control Lists (ACL)



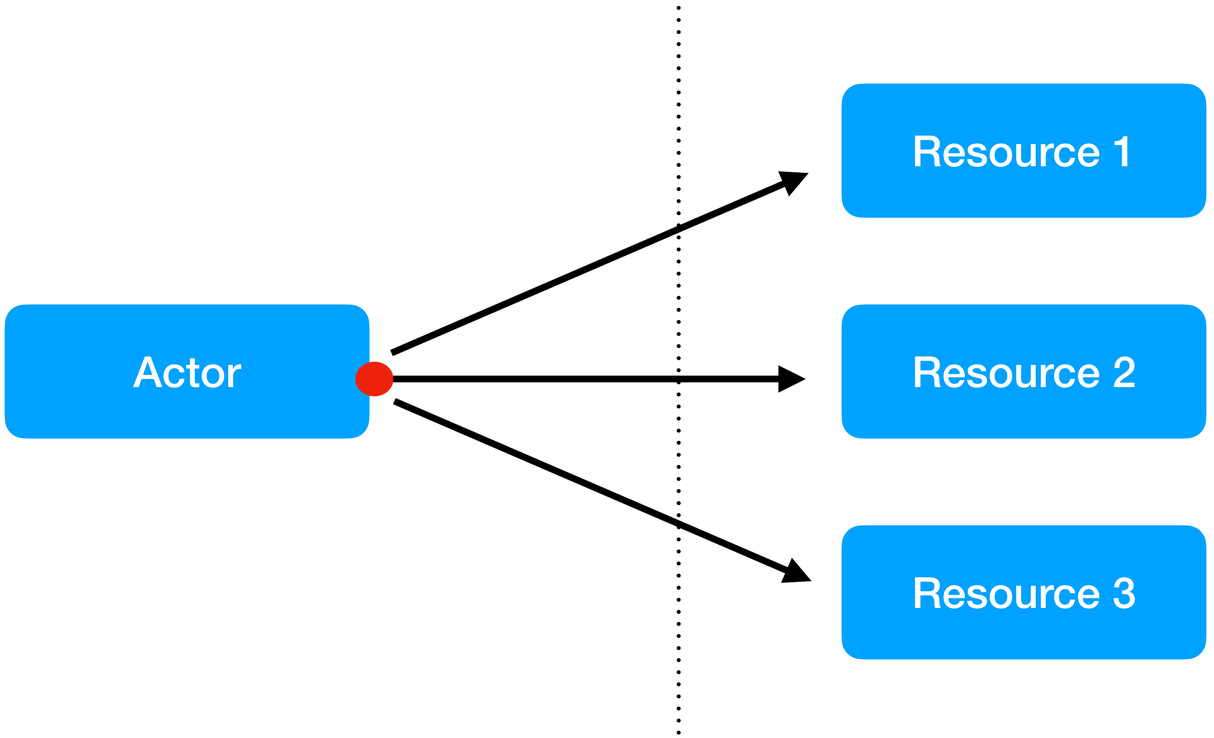
Inbound Access Control: oriented towards **what** is accessed

Capabilities

- For each **actor**, all **operations** that he can perform on each **resource** (omission = no permission, intuition = ticket)
- **Advantages:**
  - **Locality:** **permissions** of each **actor** stored individually next to the actor
  - **Isolation:** an **actor** can only access a limited number of **resources**
- **Disadvantages:**
  - Not possible to determine the **actors** that can access a **resource** without checking all actors (e.g., to remove the resource)

	R1	R2	R3
A1	r	rw	n
A2	rw	n	r
A3	r	r	r

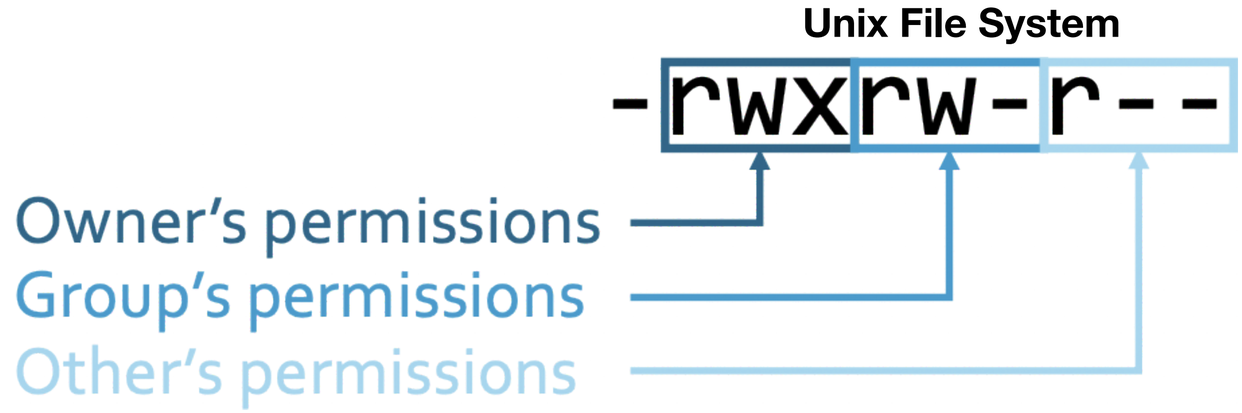
Capabilities



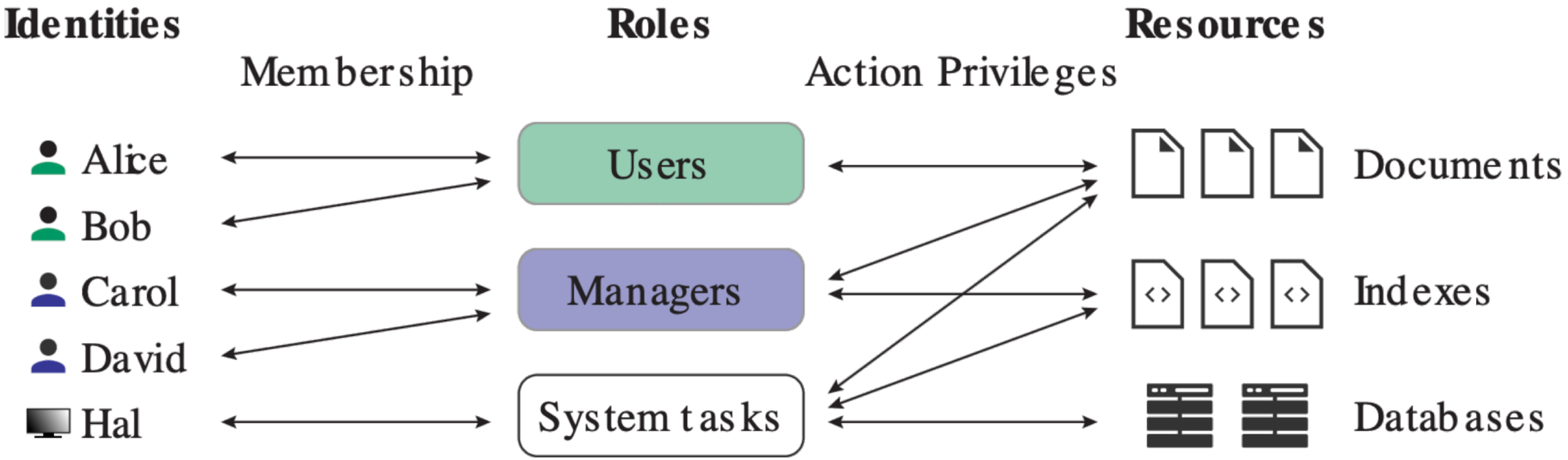
Outbound Access Control: oriented towards **who** accesses

Role-Based Access Control (RBAC)

- One of various models designed to separate the management of **resources** from the management **actors**:
  - Aggregation: relating **actors** to **roles** and **roles** to **permissions**
- The relation between **roles** and **permissions** is usually very stable:
  - Administered by who manages **resources** ⇒ similar to ACL
- The **roles** of **actors** can be more dynamic:
  - Administered by who manages **actors** (e.g., users) ⇒ similar to Capabilities



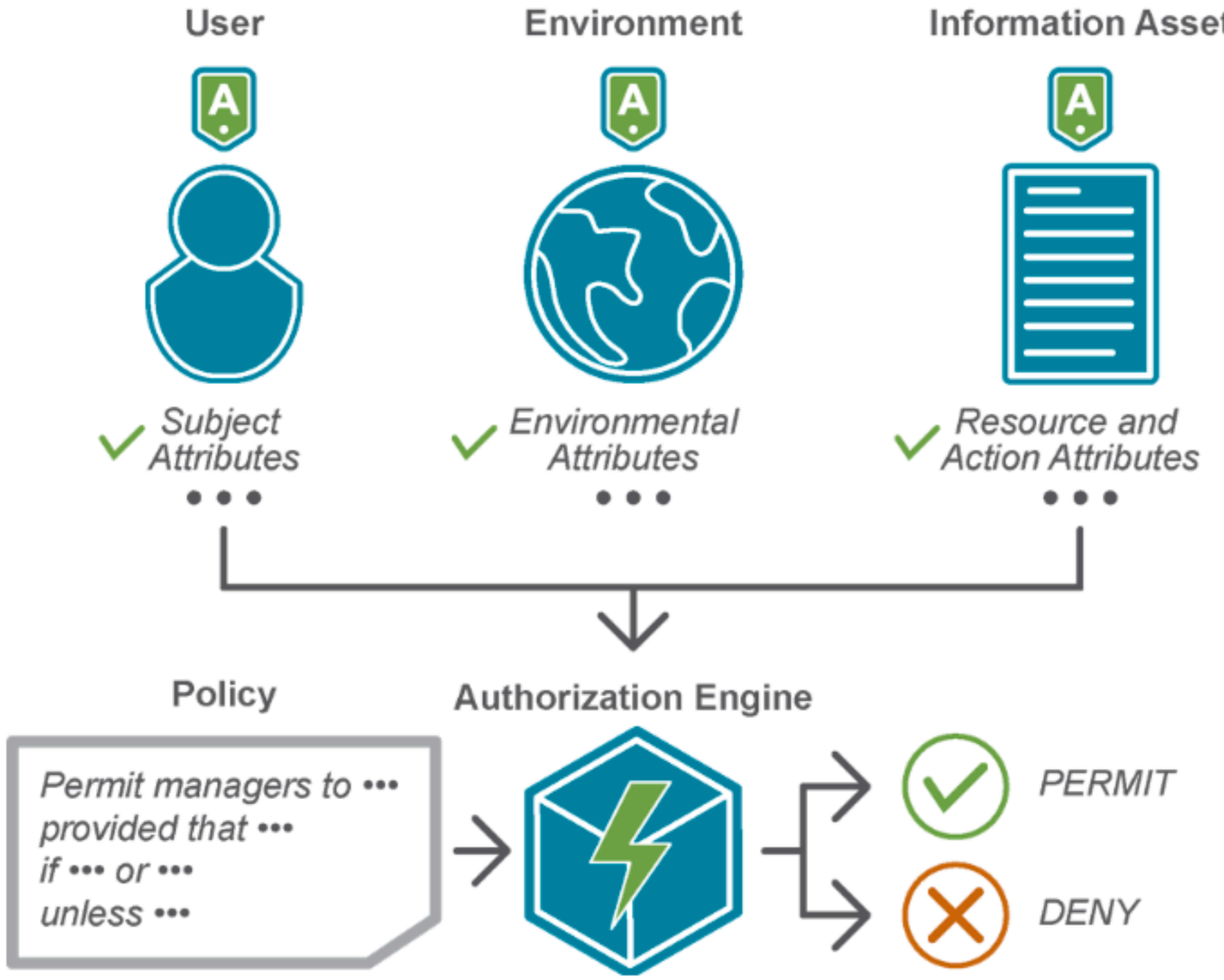
Role-Based Access Control



Attribute-based Access Control (ABAC)

- RBAC is a particular case of ABAC:
  - **Actors** and **resources** have associated **attributes**
  - Access matrix describes **permissions** based on **attributes**:
    - E.g., to access a **resource** with **attribute A**, the **actor** must have **attribute B**
    - A **dynamic access control engine** receives **attributes A and B** and decides on acceptance or not according to the security policy
  - Allows for more expressive policies ⇒ geographical or temporal context, hierarchical systems

Attribute-based Access Control

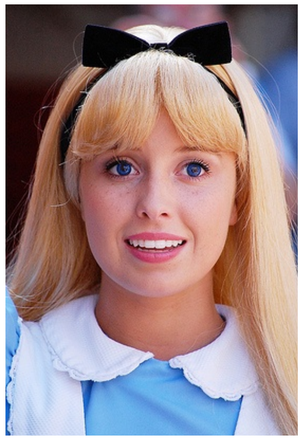


Operating Systems Security

Operating System

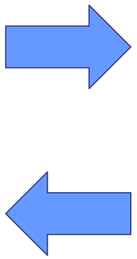
- Computer interface between **users** and **hardware**:
  - Manages access to **resources** by **applications**
    - Storage, processor, memory, I/O, network, etc.
    - Sharing of **resources** across various **users** and **applications**
    - Software that handles low-level operations and offers convenient **abstractions** for developing applications
  - Very complex and manages critical security aspects

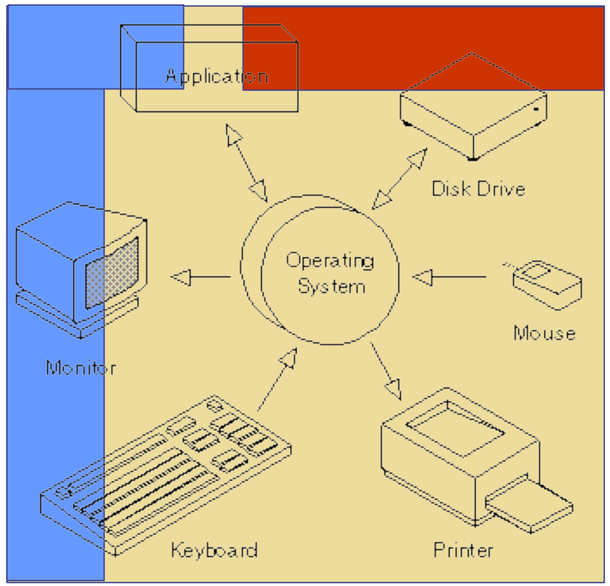
Systems Security



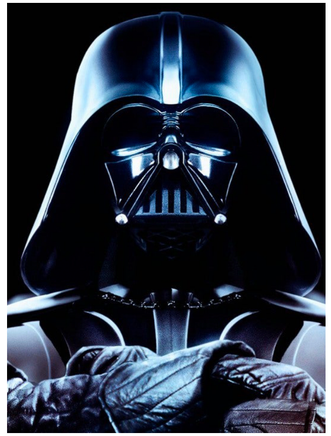
Alice

Bob





Mallory



Darth

- **Honest users/processes (Alice,Bob)** run “isolated” applications
- **Malicious attackers (Mallory,Darth)** attempt to impersonate users/processes to escalate permissions, break isolation and/or usurp resources
- Attackers may control users, processes and/or malicious files

Systems Security Model

- Multiple users with different access levels:
  - Administrators, frequent users, sporadic users, etc.
  - Different needs and privileges w.r.t. the resources under use
  - The OS has to guarantee that these requirements are respected and, at the same time, to forbid abusive behaviour
  - **Users are always potential threats**, which threat model?
  - **Resources are assets to be protected**, which properties?

Systems Security Model

- Multiple applications/services running simultaneously:
  - **Applications are also potential threats** (remember previous classes)
  - Must be protected from interference of others ⇒ **isolation**
  - This protection shall also hold for non-running applications
    - The application's state is usually stored in shared resources (e.g., disk, RAM)

Systems Security Model

- Modern Operating Systems (OSes) seek to guarantee:
  - Virtual **isolation between users**, applications and processes …
  - … despite they are **sharing the system's resources** …
  - … by enforcing **access mediation mechanisms** 
- Administering an OS ⇒ configuring such mechanisms
  - Must apply the fundamental principles  : least privilege, separation of privilege, etc ⇒ good practices

Acknowledgements

- This lecture's slides have been inspired by the following lectures:
  - CSE127: System Security I
  - CS155: Security Principles and OS Security
  - CS343: Access Control