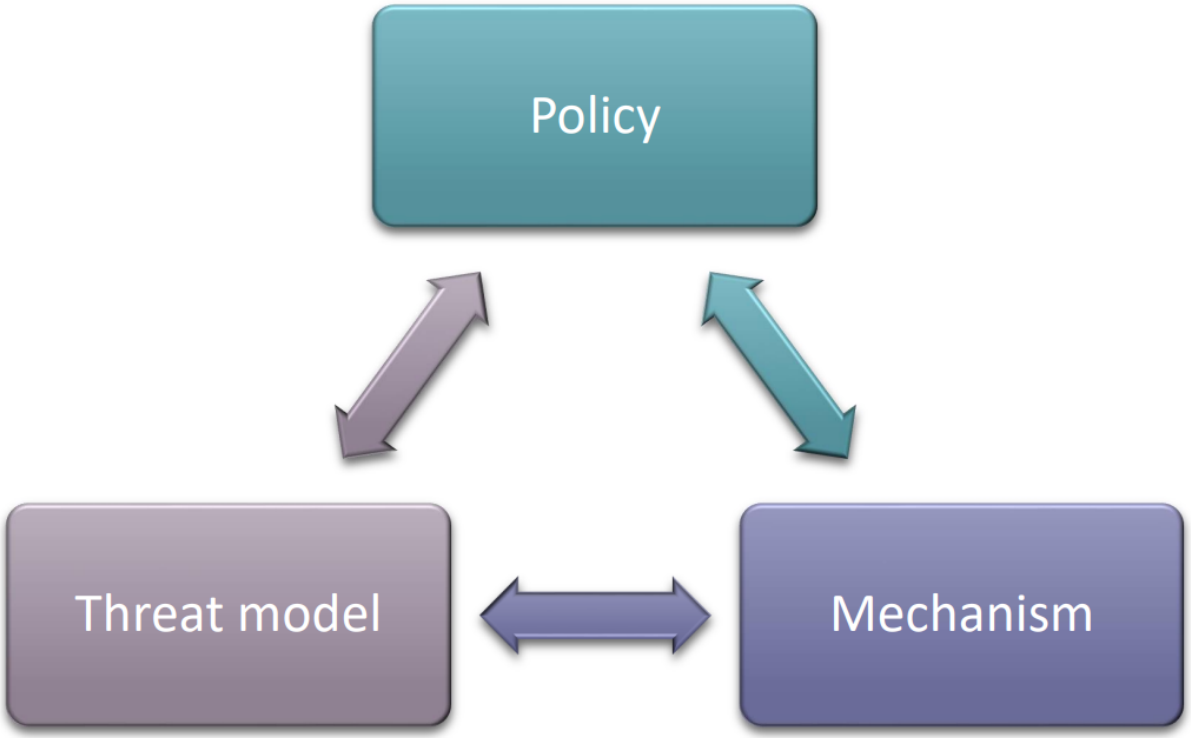
[Th 0] Introduction to security

What is security?

Security is the ability to enforce a policy in the presence of an adversary.



Why are computer systems often compromised?

Computer systems are complex

- composed of thousands of network
- protocols and software that are stratified
- heterogeneity of devices and terminals (composti da vari elementi)

Lots of details

The policy is often enforced taking care of the attacker, but it must be enforced regardless of the attacker

Example:

- easy scenario, two people can read files F1 and F2
- hard scenario, only these 2 people can read F1 and F2

There are 3 main classes of people who look for security defects.

- **clever outsiders** (who have an insufficient knowledge of the system)
- **knowledgeable insiders** (who can access on most parts of the system and have sophisticated tools)
- **funded organizations** (governments, mafia, experts)

There are also 3 types of people who leverage on the vulnerabilities:

- **black hats**
  - usually do that for bad and unethical or criminal purposes
  - they penetrate systems without permission
- **White hats**
  - usually do that for good or ethical purposes
  - they penetrate systems with permission and report the vulnerabilities
- **gray hats**
  - good purposes but illegal actions

Defects, weaknesses, and vulnerabilities

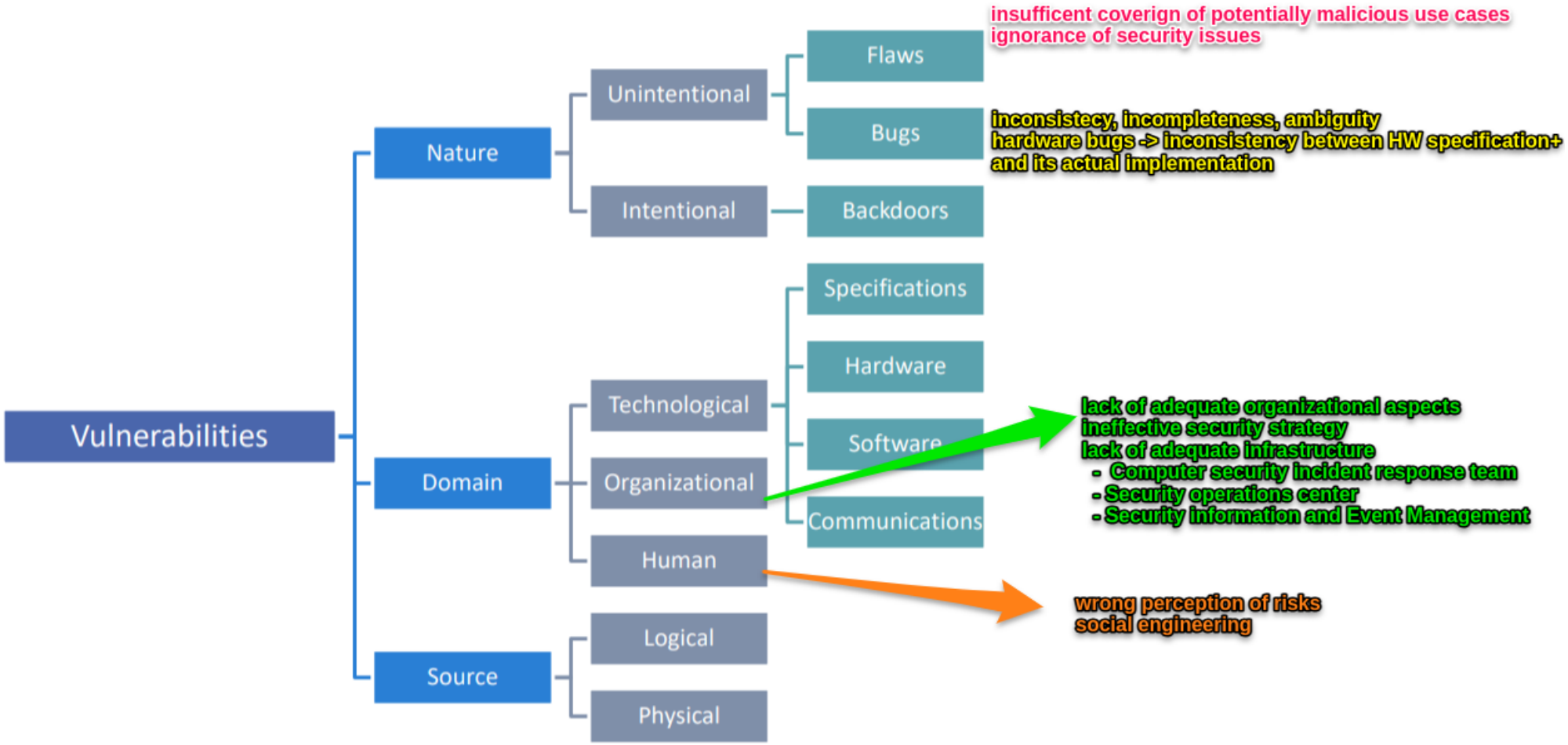
A **defect** could occur in design and implementation.

- a **flaw** is a defect in the design
- a **bug** is a defect in the implementation

A **weakness** is a characteristic that can expose the system to security risks.

A **vulnerability** is an instance of weakness that is

- **reachable** (as a part of the attack surface)
- **exploitable** by attacker



A **backdoor is an intentional channel to guarantee access or use that is outside the set of intended use cases**.
It is always a vulnerability even if the designers insert it for "good purposes".
At the hardware level, the causes of a backdoor can be:

- undocumented CPU instructions
- hardware trojans

The three components of security

POLICY

A **security policy is used basically to protect information and protect systems and resources**

We have 6 different objectives for the security policy

- **confidentiality**, this concerns the data reading, **only the authorized entities must access the data (read)**
- **integrity**, this concerns the data reading, **only the authorized entities must modify the data (write)**
- **availability,** information, **and services must be available and usable**
  - the system must be responsive to requests
- **authenticity, it should be possible to correctly identify an entity**
  - example with logins
- **accountability** (logs), **it should be possible to trace an event to an entity**
  - example using the logs
- **resilience**, **it should be possible to continue to operate under attack and to rapidly recover after a successful attack**
  - we have 4 main elements
    - manage and protect
    - identify and detect
    - respond and recover
    - govern and assure

Confidentiality, integrity, and availability are called the CIA triad.
Accountability and resilience are called additional pillars.

THREAT MODEL

It is a set of assumptions about the adversary.
It makes explicit the adversary's assumed power and has a critical importance.

- how can I assume that the attacker will not affect our system

It is a part of the **architectural modeling and risk analysis.**

Basic features

- the adversary doesn't know the password
- the adversary has no physical access to devices
-

Examples:

Network user

- **an anonymous user that can connect to a server via the internet (normal users)**
- can measure the size and the timing of requests and responses
- can run parallel sessions (more than one access)
- can provide malformed inputs (like SQL injection)
- can drop and send extra messages

Snooping user

- **a user on the same network of other users of some service**
  - someone connected to an unencrypted Wi-Fi at a coffee shop
- this user can read/measure and intercept/duplicate and modify other users messages
  - example Session hijacking ecc

Co-located Users

- **users on the same machine as users of some services**
  - example malware installed on the user's laptop
- this user can read/write the user's files
  - example attacks: password theft

SECURITY MECHANISMS

They can be **software and hardware and ensure that the policy is enforced if the threat model is correct**

Three basic kinds of security context

- **Authentication**
  - **we need to define the notion of identity and a way to connect an action with an identity**

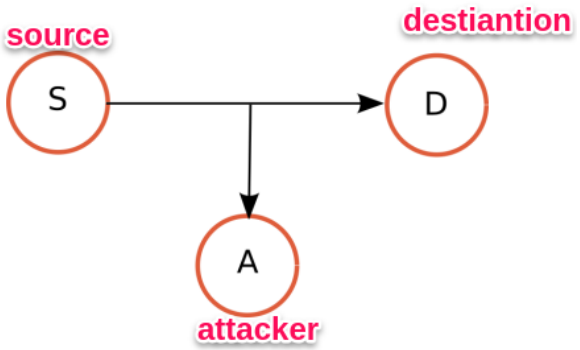
- **Authorization**
  - **defines when a principal (user correctly logged) may perform an action**
  - there are various policies that defines what a principal can do

- **Audit**
  - **maintain enough information to determine if something happened or not on the service (or server)**
  - usually, we use **logs (files) that must be safe and protected from accesses that can violate other policies**

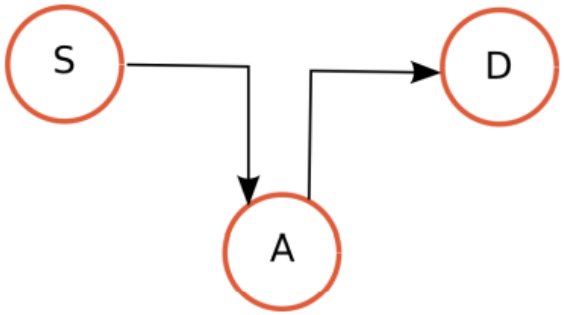
ATTACKS

We can have DAD attacks:

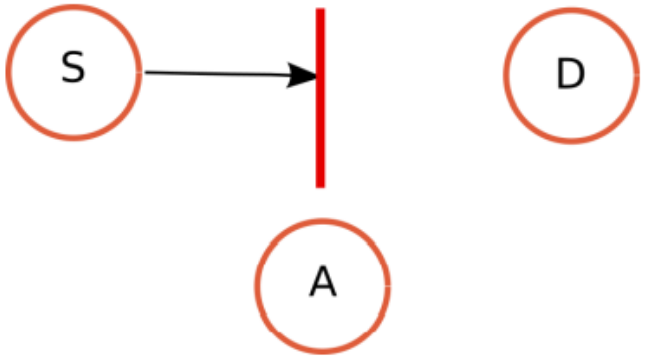
- **Confidentiality -> disclosure (or eavesdropping or stealing)**
  - **the attacker has access to information without authorization**
    - example Alice sees the bank account of Bob
  - **it breaks the confidentiality**



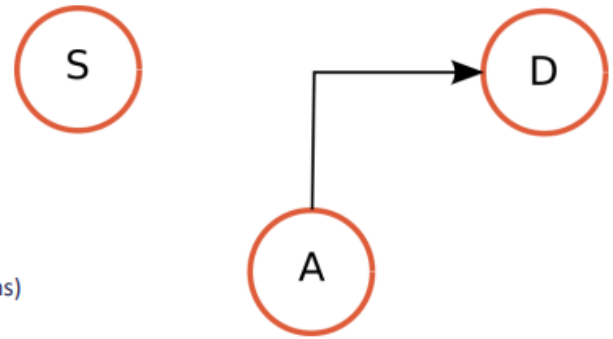
- Integrity -> Alteration (or modification or corruption)
  - the attacker modifies the information
  - it breaks integrity



- Availability -> Destruction (or interruption or inhibition)
  - the attacker stops the normal flow
  - it breaks the availability



- **Authenticity -> Forging**
  - **the attacker forges new information**
  - **breaks authenticity and accountability**
    - example a fake digital signature using an MD5 collision



sions)

POLICIES CAN GO WRONG

Example:

- "You can access your email only if you remember the password"
  - doesn't work, users can forget passwords
- you can use the email to recover it
  - doesn't work, I could have no other emails
- you can use recovery question

How we can avoid policy problems?

- **think about the implications of policy statements**
- **some policy-checking tools can help**

THREAT MODEL CAN GO WRONG

Example: We could assume that users have knowledge about phishing problems, so we underestimate the problem.

- sites not authenticated
- email and web protocols not originally designed for remote authentication

Example: We could assume that users have knowledge about spoofing problems, so we underestimate the problem.

- email not authenticated
- email and web protocols not originally designed for remote authentication

Example: cryptographic key length

- kerberos with 56 bit key DES

Example: CAPTCHA

- used to limit spam

- there are human captcha solvers in third-world countries

Example: hardware backdoors by governments (NSA)

- "clipper chip" with secret design and implementation (security through obscurity) used to "secure voice and data messages"
- it allows the federals to decode messages and voice
- obvious outcome: the chip transmitted the info to reconstruct the encryption key, and the info about devices could be attached to other devices (fake identities)

And any others

How to avoid model problems?

1. **use explicit models to understand the weakness**
2. **use simple and general models to encompass many cases**
3. **use a better design to eliminate reliance on certain assumptions**
4. **Defense in depth**
    1. provide different levels of security under different levels of assumption

MECHANISM CAN GO WRONG

In general, in mechanism, we could have bugs (in general 1 per 1000 lines of code) or we can have bugs in security-related code but also bugs in code that may seem unrelated to security.

Example:

- apple's cloud didn't enforce the same security mechanisms at all APIs to perform the login
    - in particular, find-my-phone didn't control the number of password attempts

Example Insufficient randomness

- java for android had a bug in SecureRandom class (si dimenticava di inizializzare in modo corretto il seed)
  - for this was easy to find some repeated keys
- debian accidentally "disabled" the randomness in the OpenSSL library

Example race condition:

- a program depends on timing or on the interleaving of multiple threads or processes
- so we could have interference among multiple threads or processes that share same resources
- so this can create time of check time of use bugs

Example Buffer overflow

How to avoid mechanism problems?

1. **reduce security-critical code**
2. **avoid bugs in security-critical code**
3. **use tested security mechanisms (already developed)**

GENERAL LESSONS

1. **we cannot get the right policy/threat model/mechanism on the first try**
2. **we need to iterate**
   1. design, watch attacks, update understanding of threats and policies
   2. use well-understood components and designs
   3. use public databases of vulnerability, bug bounty programs
   4. threat model change over time
3. **defender is often a disadvantage in this game**
   1. limited resources
   2. other priorities

    3. balance security against convenience
4. **a determined attacker can usually win**
    1. so we need a recovery plan

GOOD THINGS TO DO TO ACHIEVE A GOOD LEVEL OF SECURITY

- **make the cost of attack greater than the value of the information**
- **make the system less attractive**
- **find techniques that have big security payoff**
- **sometimes security increases the value for the defender**
    - VPN (employee can work from home)
    - sandboxing to have more confidence on running software we don't trust