# Fundamentos de Segurança Informática (FSI)

2024/2025 - LEIC

# Security Model

**Hugo Pacheco**
**hpacheco@fc.up.pt**

# Correctness ⊂ Security

**Bad behaviour is large and hard to quantify**

- Correctness: good input ⇒ good output
  - e.g. “*valid* user and date”
    ⇒ “user history in that date”
- Security: bad input $\not\Rightarrow$ bad output
  - Integrity: “data remains trustworthy”
  - Confidentiality: “sensitive data is not revealed”

# Two models: Binary

## secure vs insecure

- Typical of cryptography and dependable systems
- Formally define the capabilities of attackers **X** and the security goals **Y**
- No attacker limited to **X** can break security goal **Y**
- Common terminology: security proof, secure by design

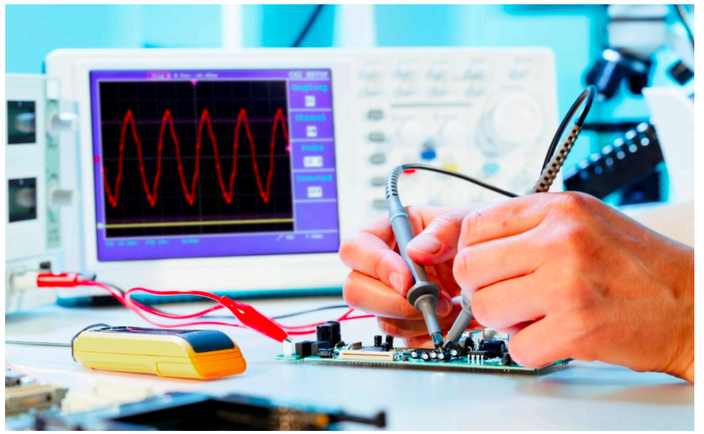
# Two models: Risk Management

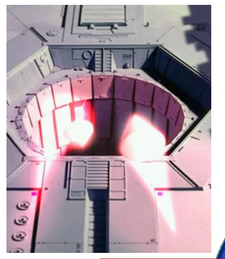
## more secure vs less secure

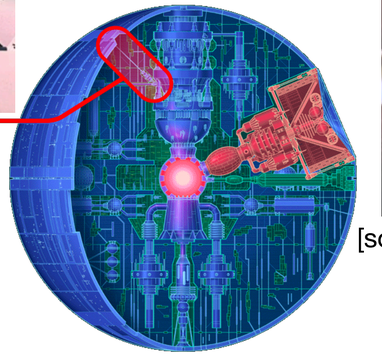
- Typical of software engineering and security of real-world systems
- Minimize risk w.r.t. the more likely threats
- Optimize the cost of security measures vs eventual losses
- Common terminology: resilience, mitigation, risk, security engineering

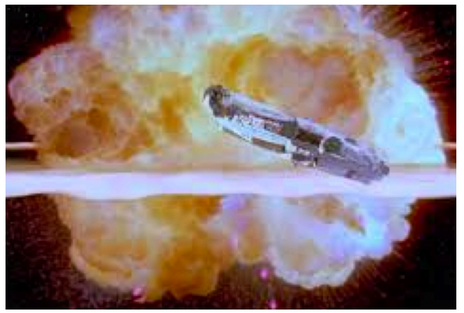
# Both models have limitations

- Binary:
  - Does not scale for complex systems
  - Abstract model ≠ concrete system
  - Formal models may ignore real problems, e.g., side-channels
- Risk management:
  - The risk analysis may be wrong
  - We are never really secure (if that exists!)
  - A wrongly-catalogued threat may mean severe exposure





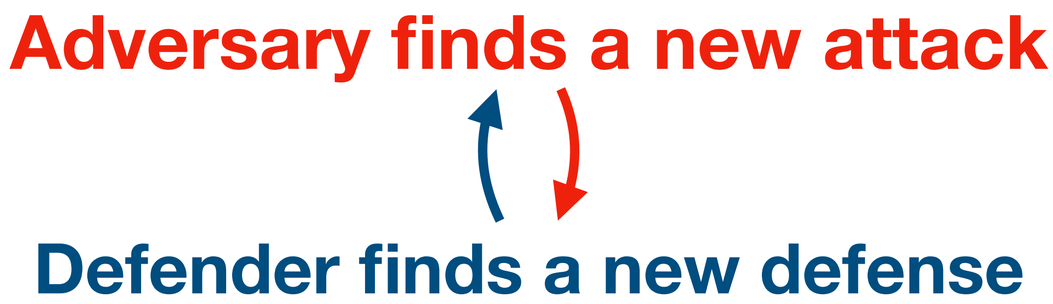


[source: Star Wars: Episode IV]

# The Security Lifecycle

- Security is a continuous process
  - even when the system is no longer operational! (e.g., electronic voting)



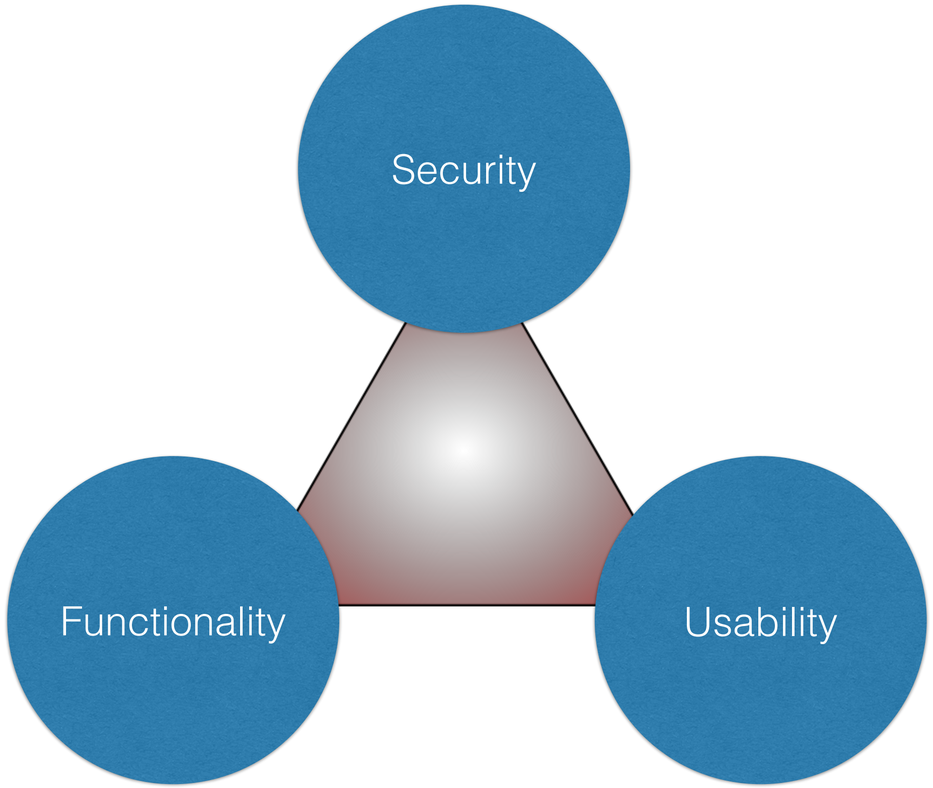

- The best that we can strive for is a balance over time
  - binary vs risk management = longer cycles vs smaller cycles

# Working in Cybersecurity

Paranoia

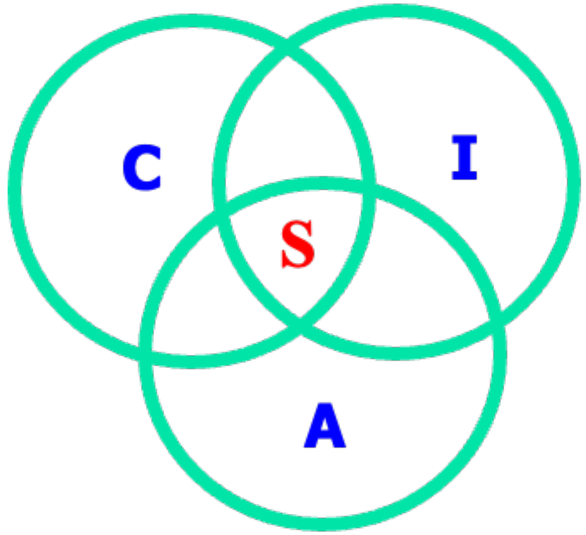
Rationality

# A triad of compromises



# Security = CIA

- 3 essential security axes:
  - **Confidentiality** (secrecy, privacy)
  - **Integrity** (non-tampering, reliable data, authenticity of origin)
  - **Availability** (existence, liveness)




**S** = **Secure**

# CIA: Always a compromise

- Absurd Example 1: C vs IA
  - **+C**: Disconnect system from the Internet
  - **-A:** none, **-I:** out-of-date updates
- Absurd Example 2: I vs CA
  - **+I:** Extensive digital and manual data validation
  - **-C:** more agents can access data, **-A:** validation process “freezes” data
- Absurd Example 3: A vs CI
  - **+A:** geo-distributed cloud service
  - **-C:** accessible from the Internet, **-I:** in-transit data may be tampered

# Risk Management

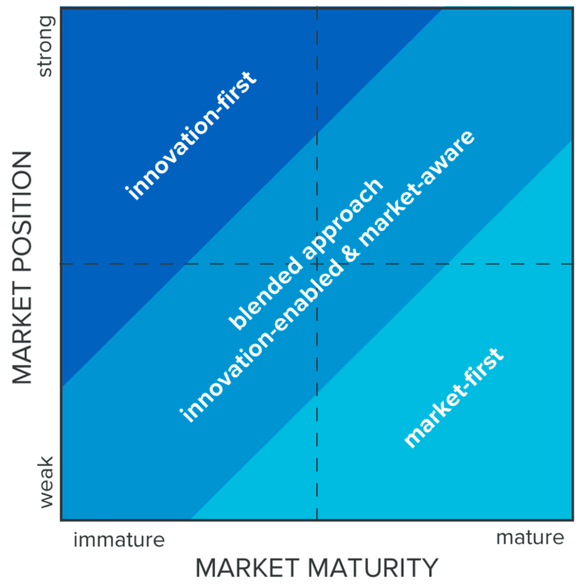
- Security is a risk vs cost management process

**How much would an attack cost?**

**How much would a defense cost?**

- It may be cheaper to accept the risk of the attack occurring (e.g., credit card fraud vs nuclear attack)

- Balance with go-to-market approaches (security ∈ innovation)



# Risk Management: Assets

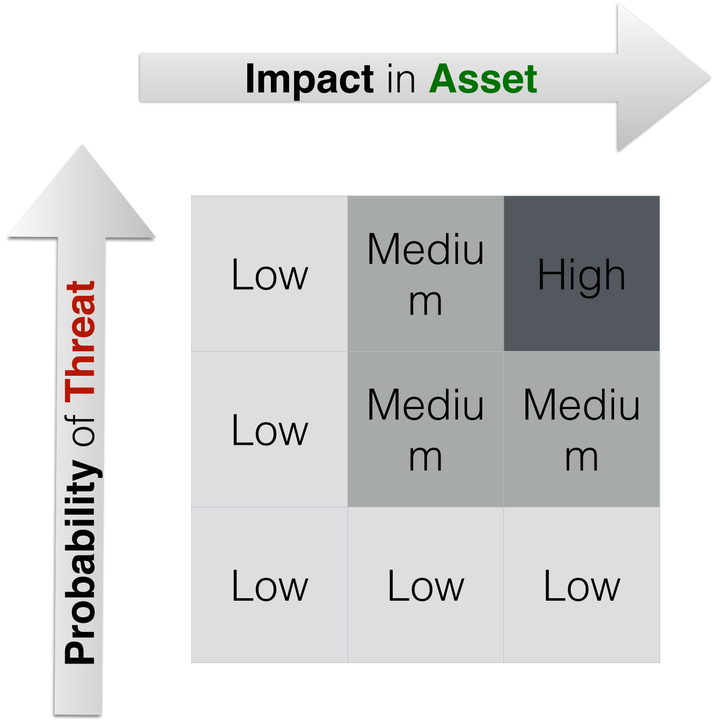
- We **manage the risk** of relevant **assets** being improperly used or unavailable
- An **asset** is a resource that holds value for an **actor** of the system:
  - information
  - reputation/marketing
  - resource with an intrinsic monetary value
  - Infrastructure
  - etc.

# Risk Management: Assets

- We manage the risk of an **asset** losing its value during system operation
- CIA ⇒ Risk of **loss of value** due to:
  - Confidentiality (unauthorised accesses)
  - Integrity (unauthorised modifications)
  - Availability (service downtime)

# Risk Analysis Matrix

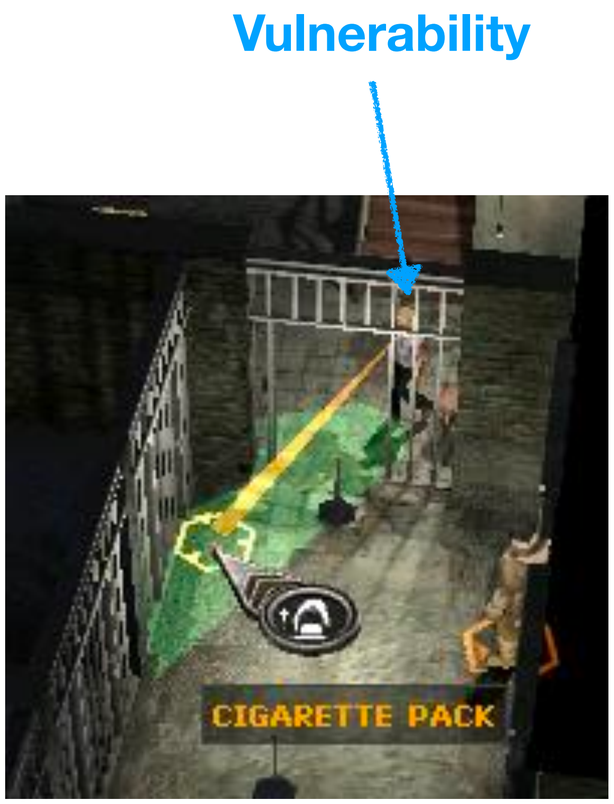
- 2 axes: **assets** + **threats**
- **Risk** depends on two critical factors:
  - Potential impact in the **asset**
  - Probability of the **threat** occurring
  - The mitigating action depends on the assessed risk
- Threat Modelling, e.g., STRIDE methodology



# More terminology…

# Vulnerabilities

- A **vulnerability** is a latent failure that is accessible to an adversary
- Often non-intentional, originating from design flaws:
  - Bad quality software
  - Inappropriate requirements analysis
  - Misconfiguration
  - Wrong usage




[source: Commandos game series]

# Attacks

- An **attack** occurs when someone tries to explore a vulnerability
- Kinds of attacks:
  - Passive (e.g., eavesdropping)
  - Active (e.g., guessing passwords, Denial-of-Service)
- When an **attack** is successful, the system *has been compromised*

Active **Attack**

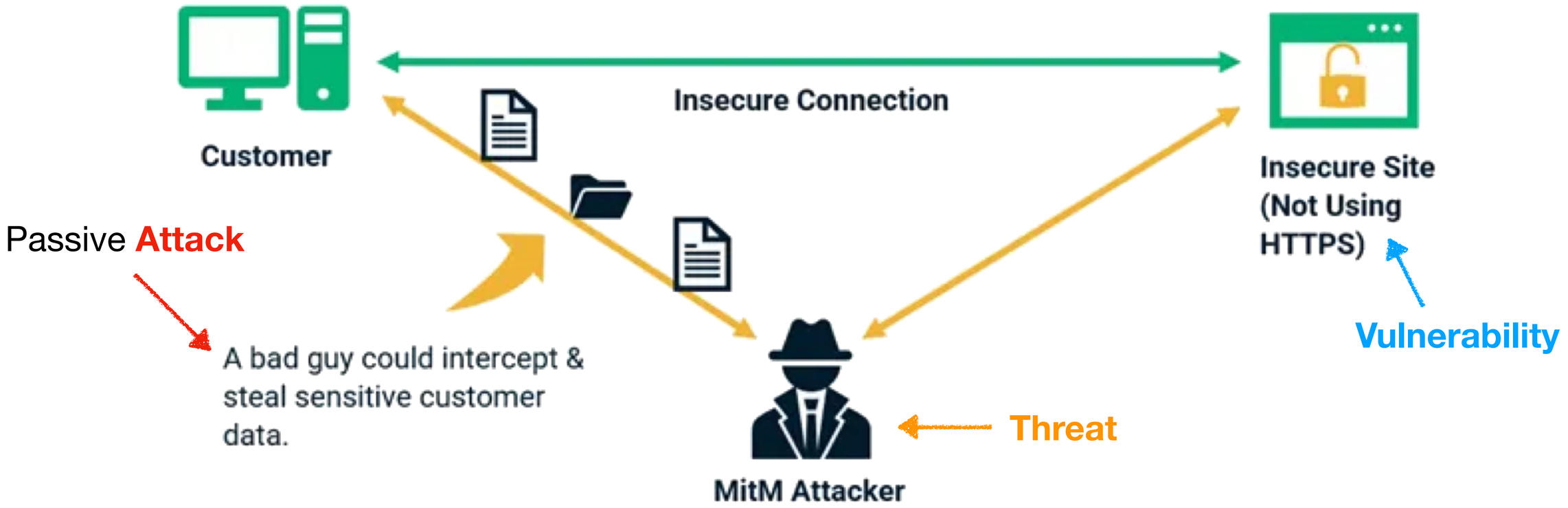

[source: Commandos game series]

# Threats

- A **threat** is a possible incident that may bring negative consequences for the system, person or organisation
- Typically defined according to their kind and origin:
  - **Kind**: physical damage, loss of service, breach of information, technical failures, abuse of functionalities
  - **Origin**: deliberate action, negligent action, accident, environmental event, flaw, external actor
- Identified and classified according to their relevance
  - Security profiles vary across communities (e.g., finance vs water supply)

[source: Commandos game series]

# Vulnerability, Attack, Threat?



# Structure of an attack


- An **attack** needs MOM:
  - **Motive**/Threat: DoS, tampering, theft of information, natural catastrophe, …
  - **Opportunity**/Vulnerability: something that can be explored
  - **Method**/*Exploit*: way to explore the vulnerability

# Reporting

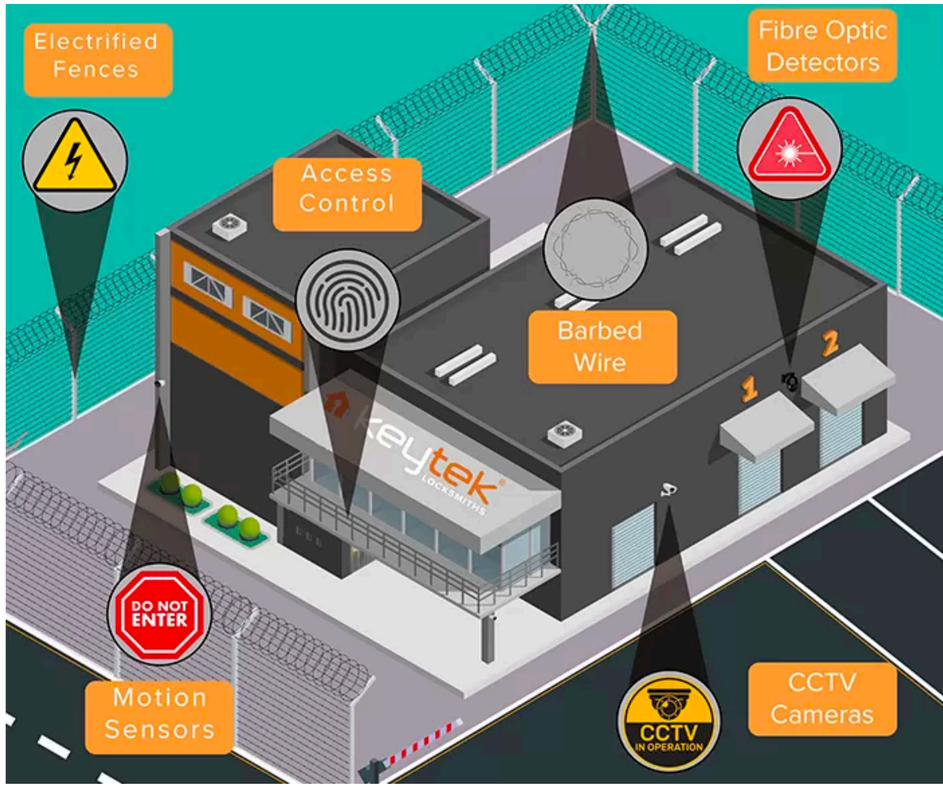

- New vulnerabilities appear every day
- Vulnerability Reporting
  = community process of managing vulnerabilities
  - Identify, classify, disclose, detect, mitigate, eliminate
  - **Threat**: CWE (https://cwe.mitre.org)
  - **Vulnerability**: CVE (https://cve.mitre.org/)
  - **Exploit**: https://www.exploit-db.com/, https://www.rapid7.com/db/, etc.

# Threat Model

- Security goals: assets, what are we protecting from whom?
- Who are the adversaries?
  - Motivation
  - Capabilities
  - Kind of access
- Adversarial thinking: What kind of attacks do we need to protect against?
- Scope: What kind attacks can we ignore?

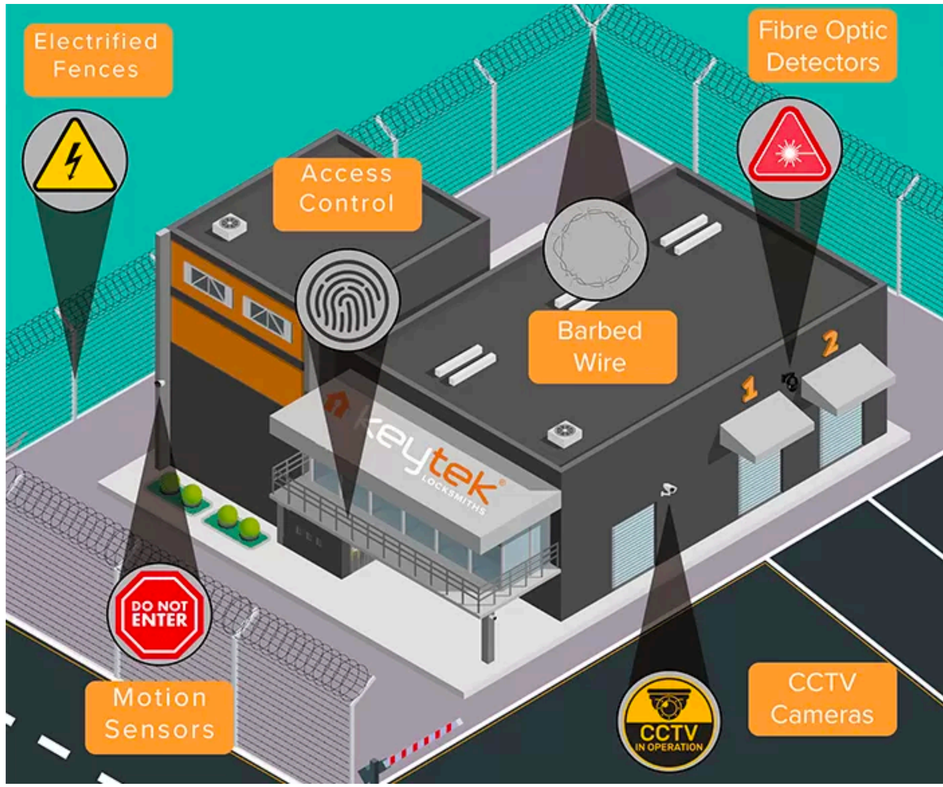
# Threat Model

- Security perimeter
  - Frontier that delimits a context with the same security level
  - Any external inputs are suspicious
- Attack surface
  - Points of contact with the exterior within the security perimeter



# Security Mechanism

- A **security mechanism** is a method, tool or procedure that allows to implement (part of) a **security policy**
- May not be technical, e.g., require personal identification at a corporate building entrance
- Part of our **trust model** consists in trusting that a (set of) **security mechanism**(s) properly implement a **security policy**

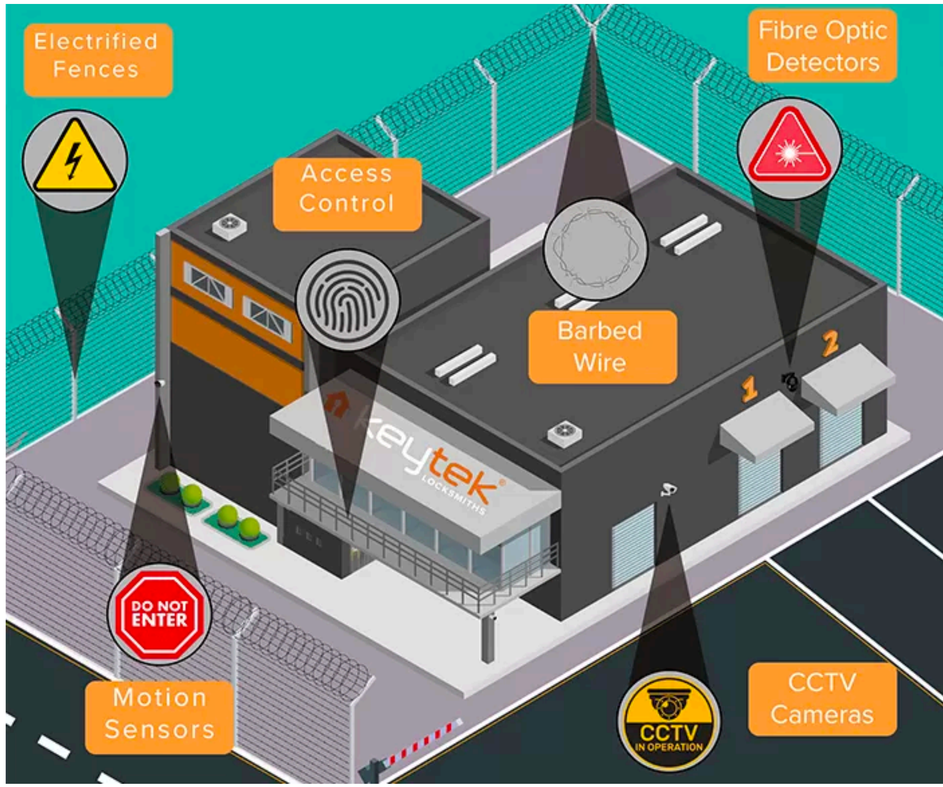


# Security Mechanism Examples

- Identification/authentication mechanisms (e.g., biometry, one-time passwords)
- Access control (e.g., RBAC)
- Cryptography (e.g., ciphers, MACs, signatures)
- Physical control (e.g., vaults, tourniquets)
- Auditing (e.g., penetration testing)

# Security Policy

- A **security policy** determines:
  - a set of processes/mechanisms that must be implemented
  - To guarantee security according to a defined threat model
- May have as goals:
  - prevention, detection and/or recovery

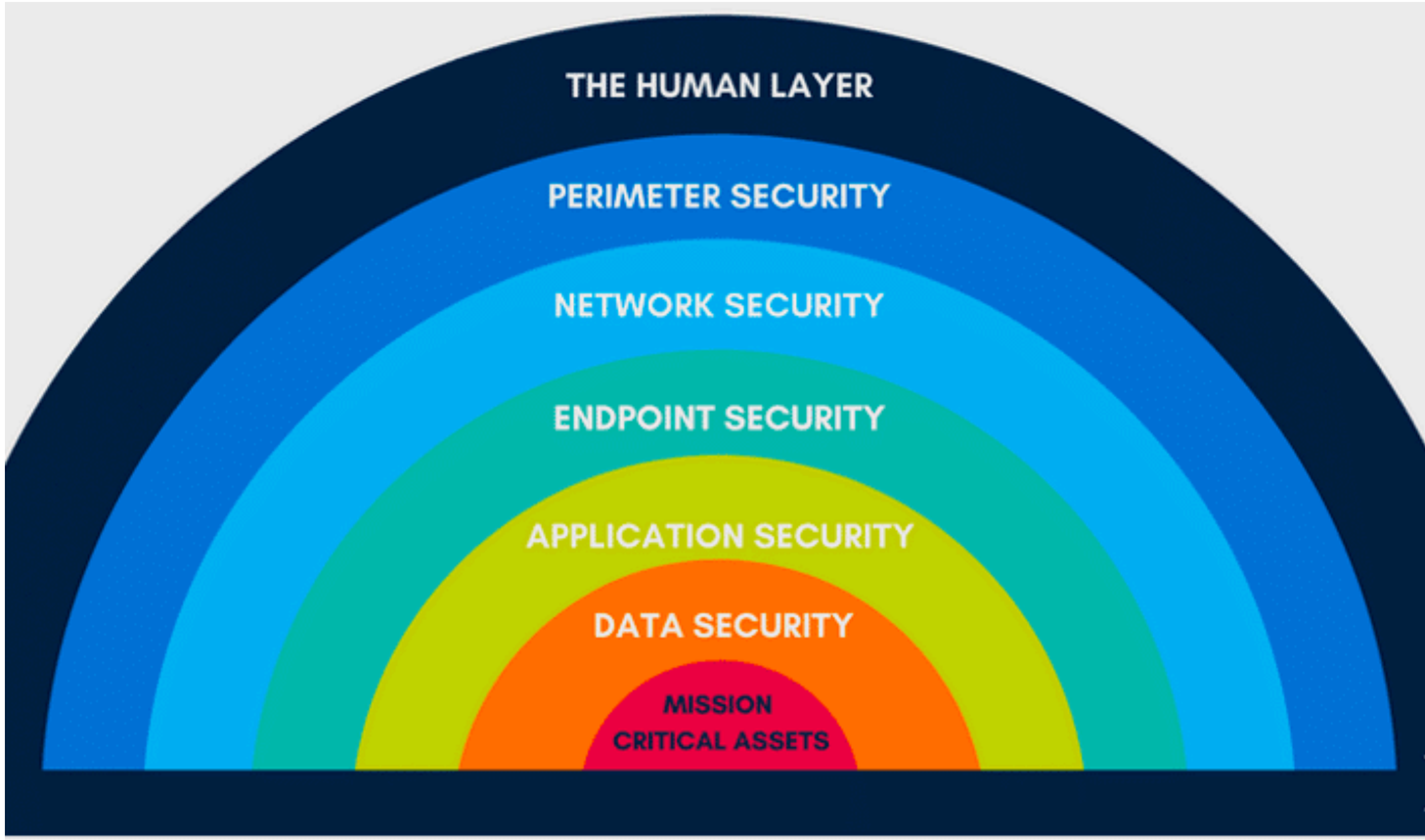


# Security Policy Examples

- Physical security policy
- Access control policy
- Password policy
- Email policy
- Remote access policy

# Managing Complexity

- In complex systems, security is handled in a layered way
  - …
  - Physical Security
  - **Network Security**
  - **Systems Security**
  - **Software Security**
  - **Data Security / Cryptography**
  - …



# Trust

- It is essential that "security" is not a “leap of faith”

- Security Engineering 
  - Building **trust** via rigorous processes/arguments

# Security Engineering I

- Defining the problem:
  - Security requirement analysis
  - Defining the threat model:
    - Security assets and security goals
    - Relevant classes of threats / adversaries

# Security Engineering II

- Defining the **trust model**:
  - In which components / actors do we trust?
  - To do what?
- Defining the solution:
  - Designing the adequate **security policies** (what)
  - Identifying the necessary and sufficient **security mechanisms** (how)

# Security Engineering III

- Validating the solution:
  - Validating how the trust models are adequate for the concrete system
    - (the adversary is not forced to follow our threat model!)
  - Demonstrating (formally or informally) that:
    - The **security mechanisms** are sufficient, together with the **security assumptions**, to implement the **security policies**.

Security auditing seeks to validate this process

# Penetration Testing

- A method to empirically validate the security solution
- Ethical hacking: simulating an attack to search for vulnerabilities that could be exploited by an adversary
- Black box: simulating real attacks
- White box: with privileged knowledge of the system
- The main value lies in the report and the final recommendations

# Acknowledgements


- This lecture's slides have been inspired by the following lectures:
  - CSE127: Introduction + Security Foundations