Security Requirements Engineering

Riccardo Scandariato
Institute of Software Security, TUHH, Germany

ric***do . scanda***to @ tuhh.de

Master Course “Secure Software Engineering”
Summer Semester 2022

Learning objectives

- What are security goals and security requirements ?

Reading material about goals and requirements
Charles Haley, Robin Laney, Jonathan Moffett, Bashar Nuseibeh, Security Requirements Engineering: A Framework for Representation and Analysis," *Transactions on Software Engineering,* 2008

- How to elicit security requirements with MUCs ?

Reading material about MUCs
Guttorm Sindre, Andreas Opdahl, Eliciting security requirements with misuse case, Requirements Engineering 10(1), 2005

- How to prioritize security requirements ?

Learning objectives

- What are security goals and security requirements ?

Reading material about goals and requirements
Charles Haley, Robin Laney, Jonathan Moffett, Bashar Nuseibeh, Security Requirements Engineering: A Framework for Representation and Analysis," *Transactions on Software Engineering,* 2008

- How to elicit security requirements with MUCs?

Reading material about MUCs
Guttorm Sindre, Andreas Opdahl, Eliciting security requirements with misuse case, Requirements Engineering 10(1), 2005

- How to prioritize security requirements ?

Software Requirements

Requirements:

- Descriptions of **what a system should do** in terms of the services it must provide and constraints on its operation [Somerville 2011].
- **Conditions** or **capabilities** the system must meet to satisfy a contract, standard, specification, or other formally imposed documents [IEEE].
- Reflect the needs of different **stakeholders** (clients, customers, and end-users) for a system that must serve a certain **purpose.**

Requirements Engineering:

- The process of **capturing**, **analyzing**, **documenting** and **checking** system requirements.
- It is **critical** to the success of any major development project.

Functional and Non-Functional Requirements

Software system requirements can be classified into **functional** and **non-functional**:

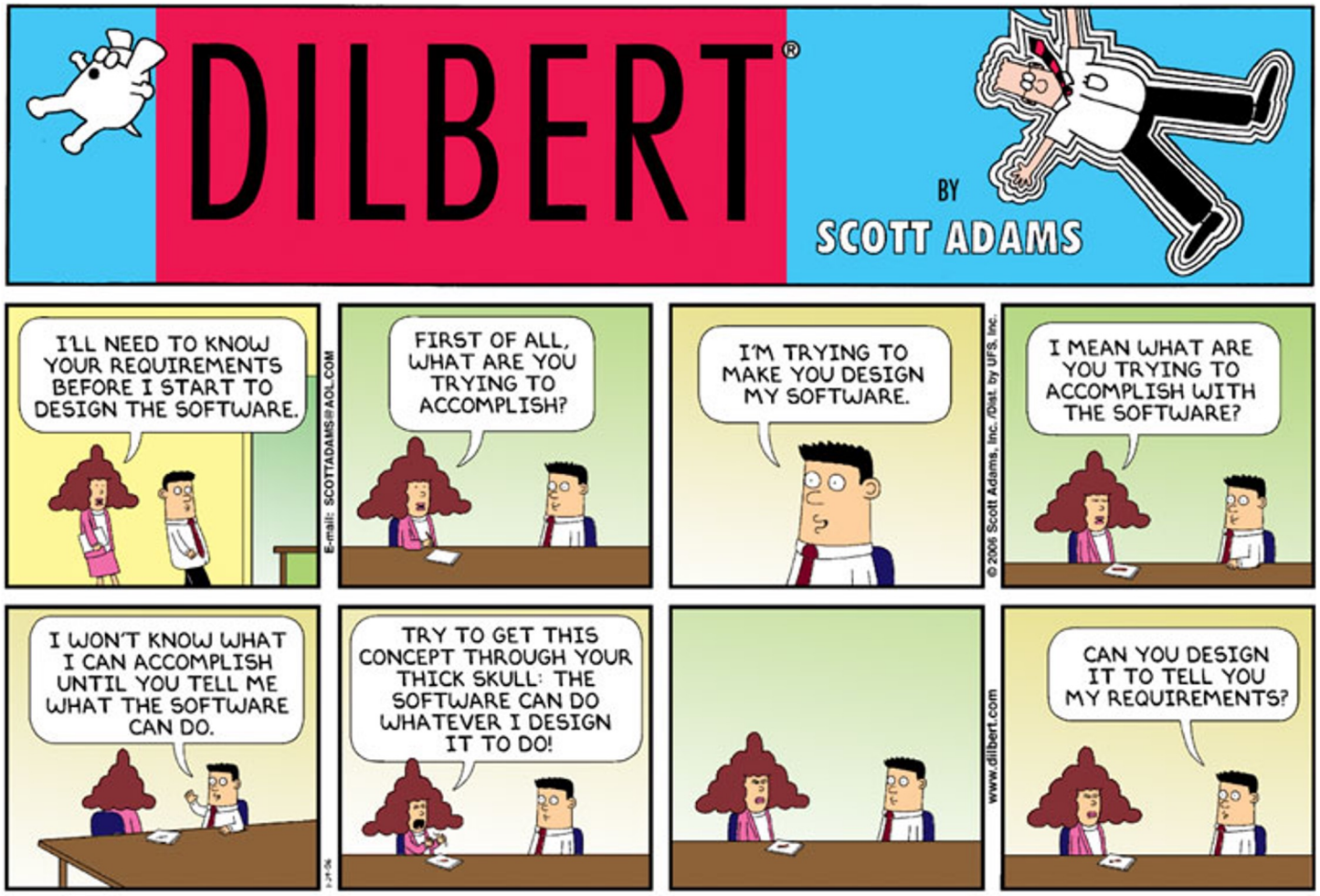
- Functional: **Statements** of what services the system should provide, how should it react to certain inputs, and how should it behave in specific situations.
  - "The system shall be able to search the students for all lectures".
  - "The system shall generate a list of students attending to an exam".
- Non-Functional: Define **constraints** on the services or functions offered by a system
  - Often referred as **quality attributes**.
  - Examples: USABILITY, RELIABILITY, SAFETY, and **SECURITY**.

Non-Functional Requirements

- Non-functional requirements **often** apply to **the system as a "whole"** rather than individual features or services
  - The system shall limit the access to specific authorized users (*security*).
- A single non-functional requirement may **generate** many **related functional requirements** and **restrict existing ones**.
- Unlike functional requirements, non-functional ones are difficult to relate to **individual system components (cross-cuttingness)**.
- Non-functional requirements may affect the **overall architecture** of a system rather than its individual components.

Non-functional requirements are often **more critical** than individual functional requirements!

Sources of Security Flaws (i)



Ambiguous and Incomplete Requirements

Typical problems of requirement engineering:

1. Not including all relevant stakeholders at the elicitation phase.
2. Restrict the analysis to functional requirements only.
3. Lack of systematic and structured methodology.

Negative consequences:

✗ Ambiguous requirements can lead to **multiple interpretations** that do not meet the stakeholder's expectations.

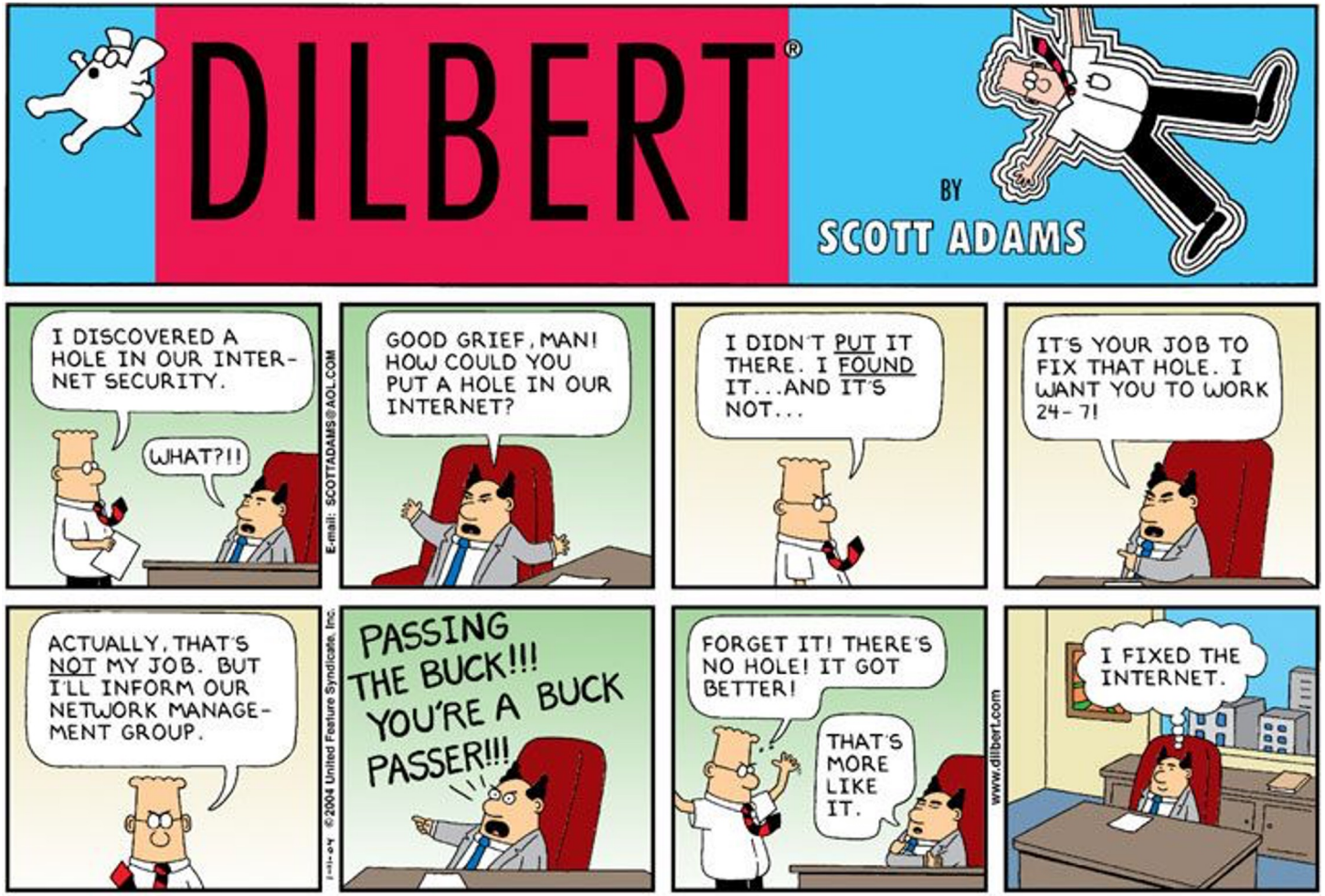
✗ Incomplete requirements can introduce **delays** and **increase costs**.

Imprecision in the specification of requirements is the cause of many software engineering problems **including security flaws**.

In case of security

- Incomplete asset analysis (information, functionality)
  - E.g., failing to identify the sensitivity of login data
- Incomplete understanding and assessment of the threat/attack landscape
  - E.g., not being aware of phishing attacks
- Incomplete, wrong, weak selection of security countermeasures
  - E.g., not specifying a two-factor authentication

Sources of Security Flaws (ii)



© UFS, Inc.

Technical Debt and Security Debt

Compromising quality aspects of a software project can be seen as *"borrowing money thinking you never have to pay it back".*

→ When not paid back promptly, **interests on the debt** can compromise the whole revenue of the project.

The term **Technical Debt** is used to describe the structural, long-term problems of software products caused by quality compromises.

Security Debt: A technical debt that entails a **security risk**.

✗ Security work is generally under-prioritized (strict production deadlines).

✗ Features and functionality, dubbed as "customer value", are pushed for as early release as possible.

✗ Security benefits are difficult to demonstrate and costs hard to justify.

Security-by-design

- ✓ Security should not be an “after though”, but an **integral part** of a software development project.
- ✓ In order to systematically develop secure solutions, security must be emphasized throughout **the whole software lifecycle**.
  - Security considerations should be integrated into the **early stages** of the development life cycle (i.e., the *requirements phase*).

STEP 1. ASSETS AND SECURITY GOALS

Assets

Technical assets are information (e.g., credit card data) or functionality (e.g., logging component) **of value** that must be properly protected

- Assets can also live in the physical world (hw, sensors, devices). This is becoming more and more important in IoT and CPS
  - IoT – Internet of Things
  - CPS – Cyber-physical systems
- Assets can also be non-technical (e.g, reputation, employees time, revenue from service)

Asset analysis

- Identifying the assets in a system
  - E.g., looking at business goals (**white hat**)
  - However, attacker has interest too (**black hat**)
  - **Challenge**: overlooking why things might be of interest to an attacker (e.g., in the case of privacy)

- Assessing the value to us ($$$) in case they are compromised
(useful in risk assessment)
  - Usually non-problematic for technical assets

- Assessing the reason why they are valuable
(leading to ***goals***)

Security concern: CIA+

ISO/IEC 7498-2: Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture

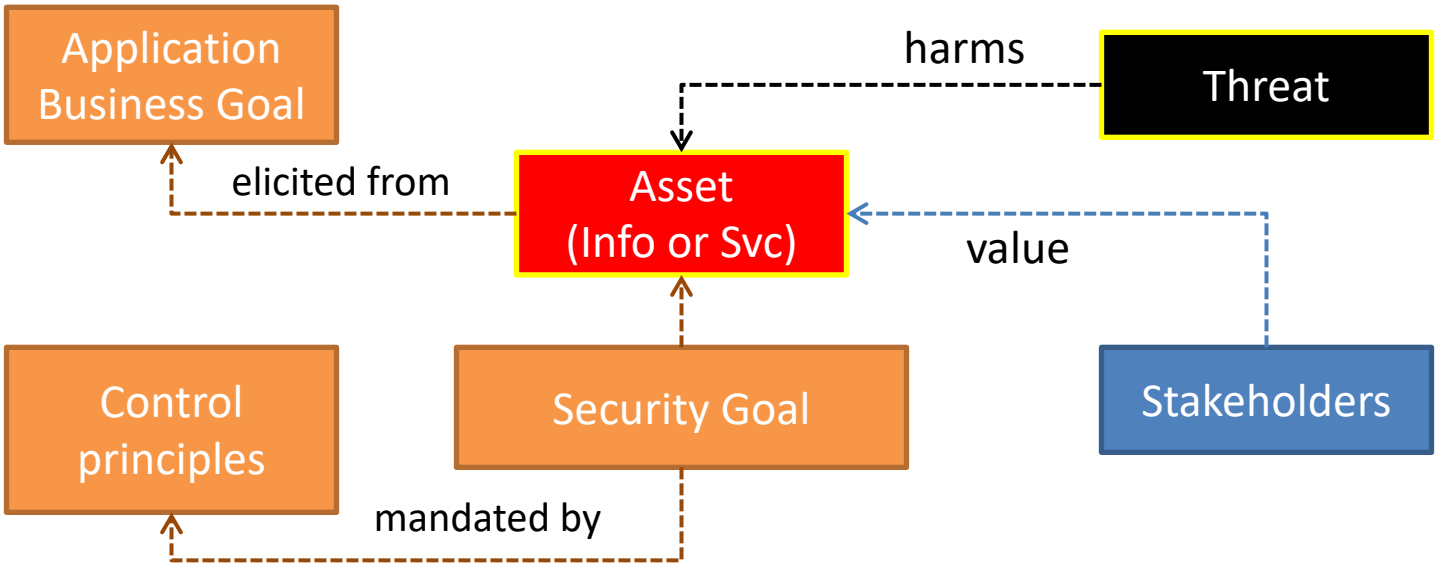
- Confidentiality (C): Protection against unauthorized disclosure of stored, processed, or transferred information.
- Integrity (I): Ensure the authenticity (e.g. origin/source) and accuracy of information. It entails restrictions for unauthorized data modification.
- Availability (A): Ensure the access (for authorized parties) to the data, resources and services necessary for the proper functioning of the system
- Access Control: only legitimate access is permitted (goal or mechanism?)
- Accountability (& non-repudiation): prove that an entity was involved in some event
  - Accountability: Ensure the recording of security relevant events and the user identities associated with these events
  - Non-repudiation: Provide unforgeable evidence that a specific action occurred (e.g., sending and receiving a message)

+ Authenticity (not in the standard)

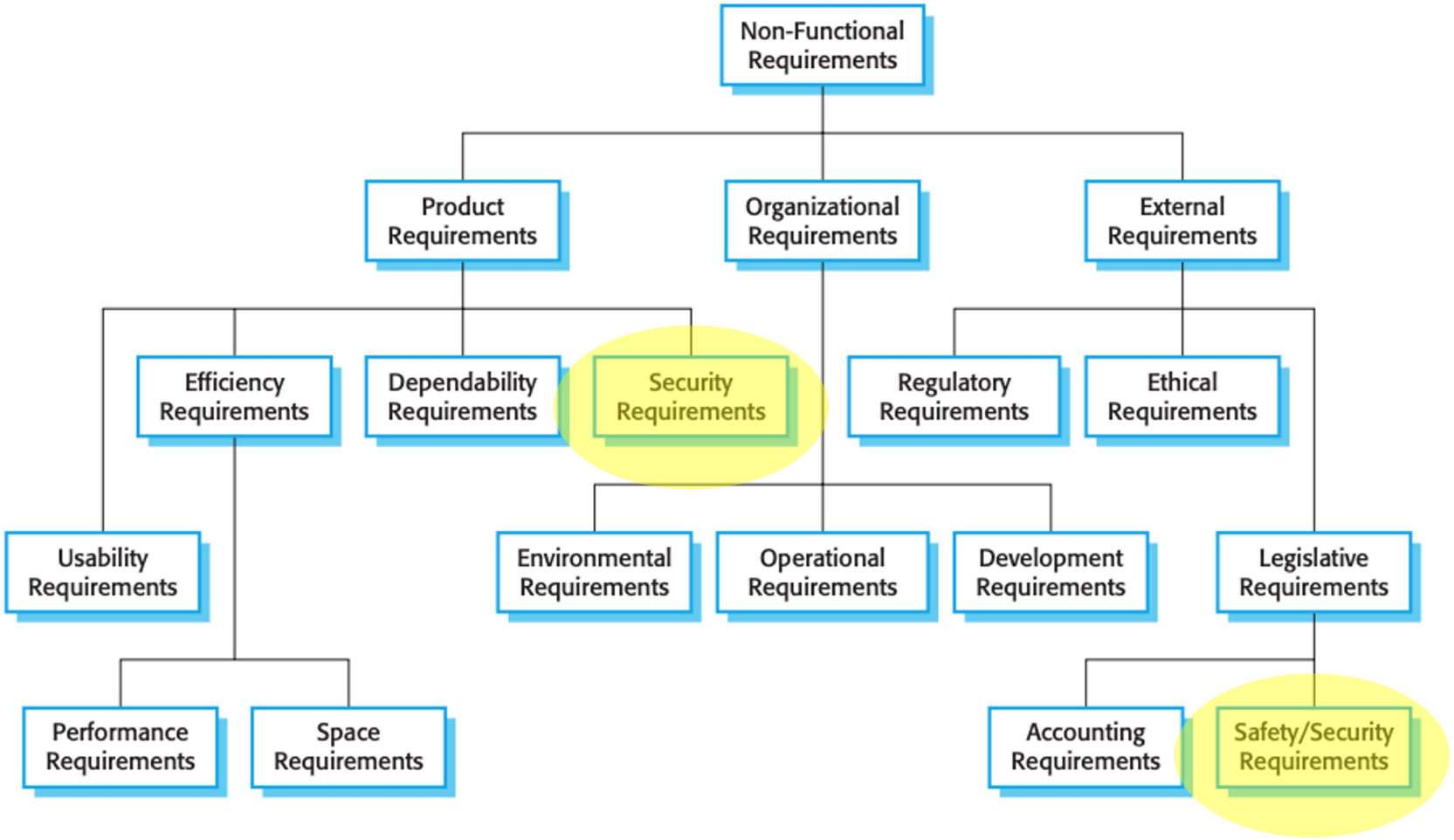
Security goals

- Concerns are abstract (taxonomical value)

- Goals represent the perceived specific needs of one or more stakeholders
- Security goals (primarily) entail the protection of an *asset* against a *harm*
  - **ACCIDENTAL** HARM → **SAFETY**
  - **INTENTIONAL** HARM → **SECURITY**



Non-Functional Requirements Revisited



Elicitation of security goals (i)

Sources of security requirements: the product (internal sources)

Security goals/requirements are dependent **on business goals**

- E.g., define those privileges that are needed for the application, then exclude those privileges that are not needed

Elicitation by conducting a **harm analysis**

- **CIA+ concerns**
- In general, harms can be recognized by negating security concerns →
  "What harm could come to [asset here] from an action violating [concern here]?"

Security goals as **"avoid"** goals

"Avoid" goals can be expressed as a triple {action, asset, harm}
where *action(s) on the asset(s) listed in threat descriptions should be* ***avoided***

Elicitation of security goals (ii)

Sources of security requirements: the product environment (extrernal sources)

Security goals may have been set elsewhere, especially when assets are covered by organization-wide policies

- Generated by **applying the management principles** to the assets and business goals of the system
- Apply **globally** throughout an organization

Provide **constraints** that would otherwise have to be derived repeatedly for each security risk analysis!

The result is a collection of ***"achieve"*** **security** ***goals*** such as "achieve separation of duties when paying invoices" or "audit all uses of account information"

Elicitation of security goals (ii) - Examples

- Company policies
  - E.g., “no administrative rights to employees”
  - “audit all uses of account information”
  - “achieve Separation of Duties when paying invoices”
- Regulations and laws
  - E.g., GDPR, compliance to HIPAA, etc.
- Business rules
  - E.g., registered user versus paying user

STEP 2. CONCRETE REQUIREMENTS

Goals, requirements, architecture

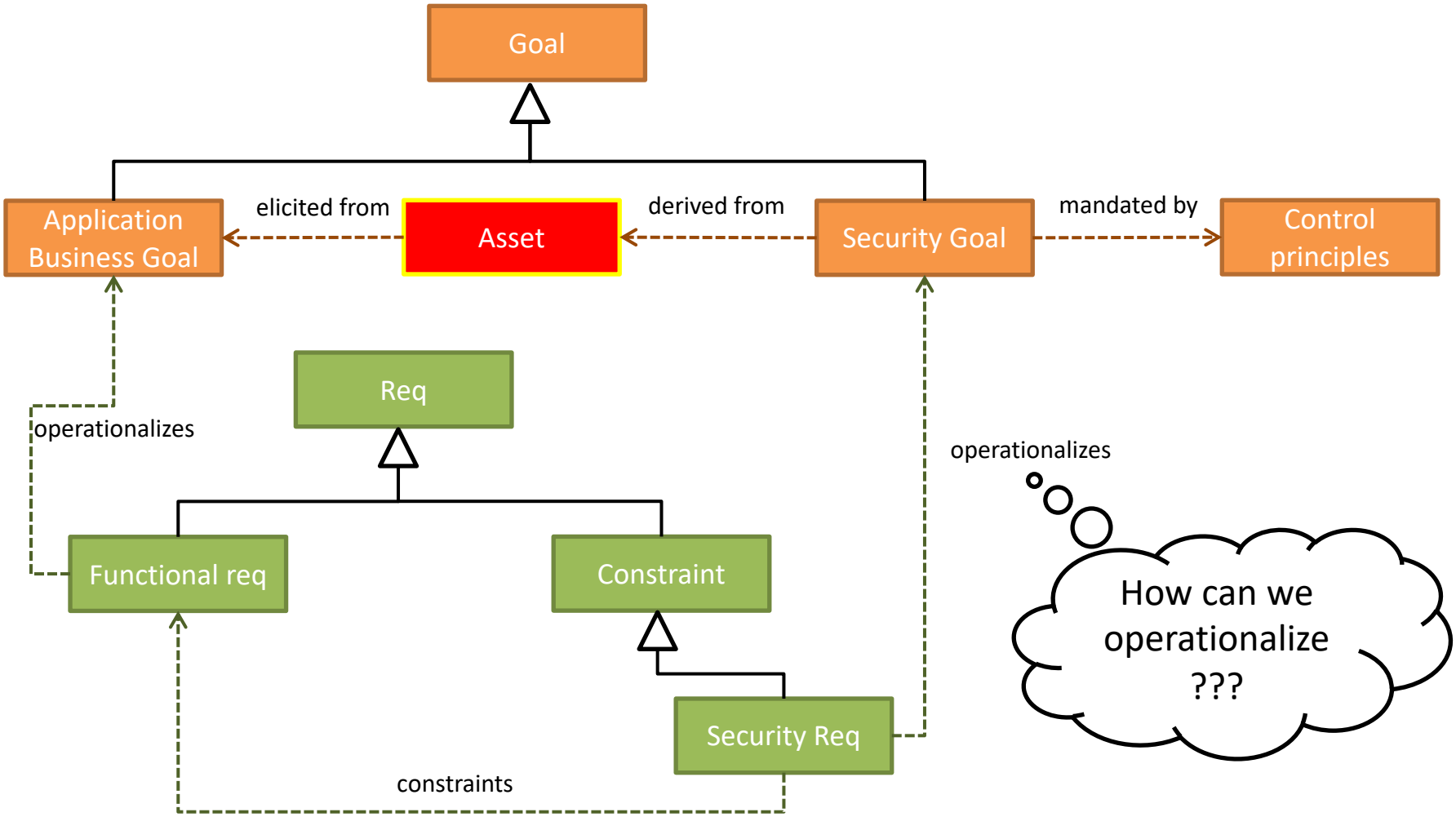
- **Goal (WHY):** Something that any stakeholder wishes to achieve
  Conflicts are possible and **must be solved!**

- **Requirement (WHAT):** A detailed (i.e., more concrete) commitment for the system-to-be (e.g., behaviours and constraints)
  They must be **realistic** → **achievable** and **verifiable**

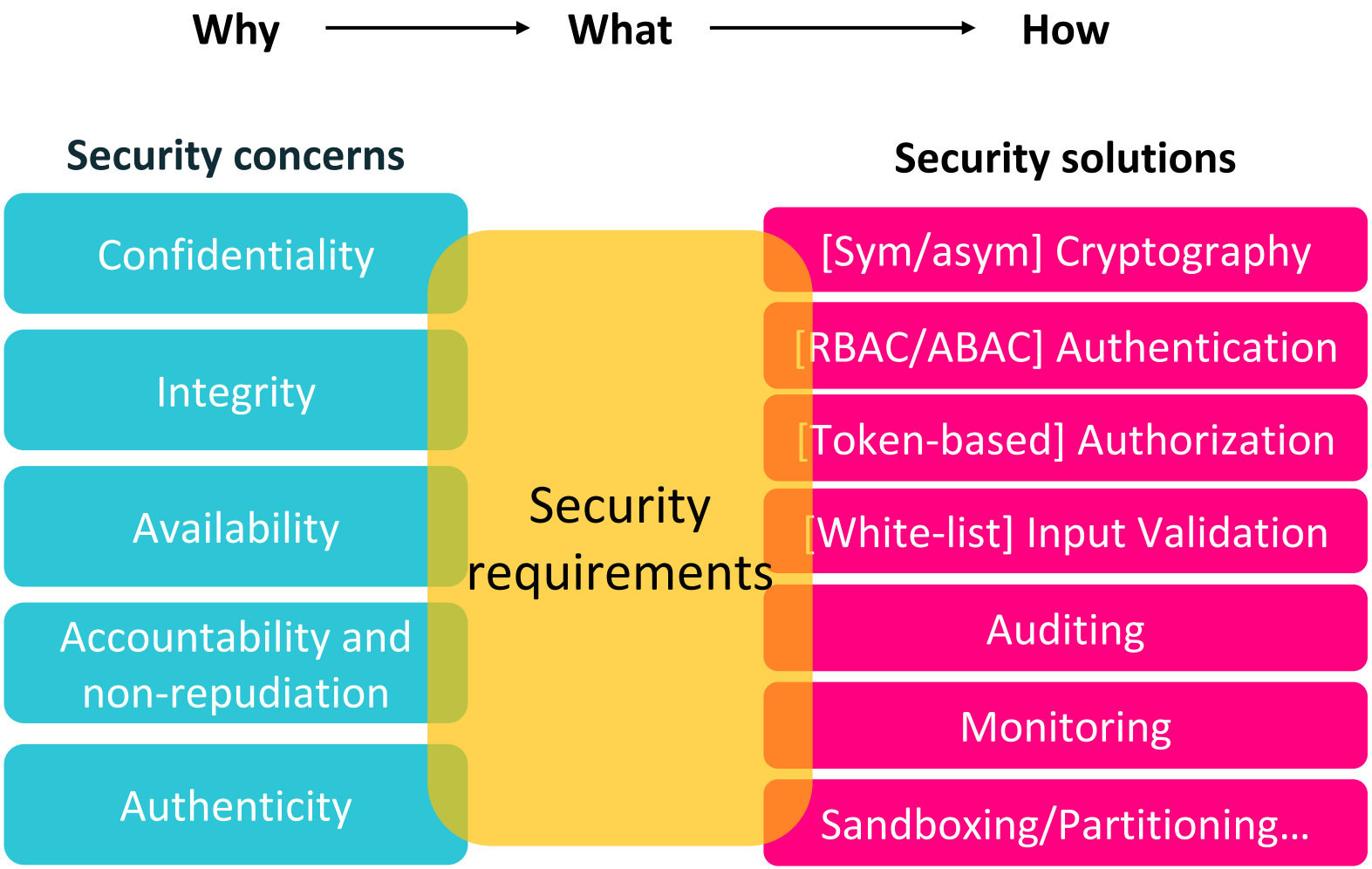
- **Architecture (HOW):** A description of the **means** needed for achieving the requirements, in terms of a **configuration of interacting components**

Strict separation not possible
(Twin Peaks → T. Heyman, et al.,
The security twin peaks, ESSoS, 2011)

Operatonalization of goals into requirements



Operationalization of security



Reality check

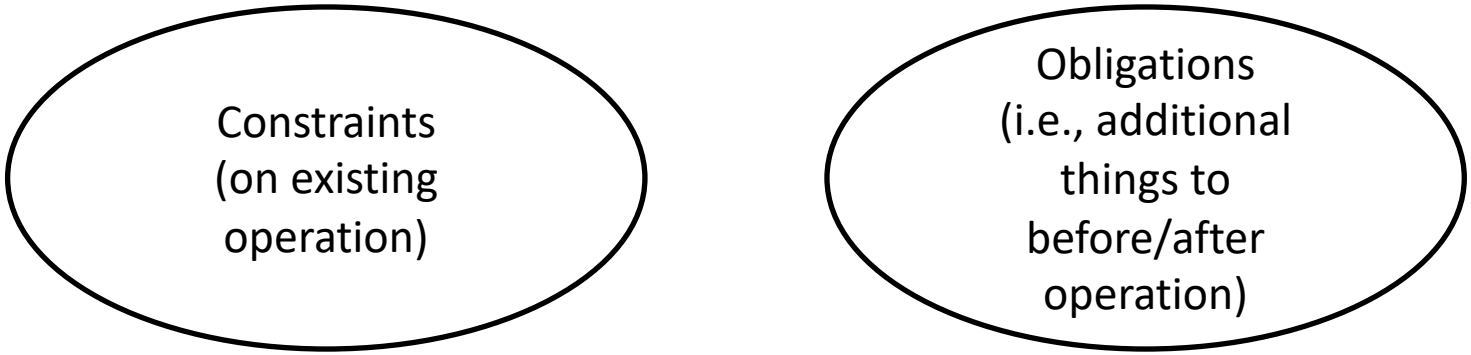
- Requirements often contain solution-oriented mechanisms (security building blocks, or even security solutions)

- "Messages exchanged between A and B should be …"
  - Confidential
  - Encrypted with symmetric encryption
  - Ecrypted with AES 256

Useful? Maybe too close to the goal

Too detailed ?

Type of security requirements



Type of security requirements

- **Simple constraint**
  - Predicates on the parameters of the operation, its originator and source
  - “The system shall not provide Personnel Information except to members of Human Resources Department”
- **Temporal constraints**
  - “The system shall not provide Personnel Information outside normal office hours”
- **Complex constraints on traces**
  - “The system shall not provide information about an organization to any person who has previously accessed information about a competitor organization (the Chinese Wall Security Policy)”

Type of security requirements

- **Constraint on response time** (availability)
  - "The system shall provide Personnel Information within 1 minute for 99% of requests"

- This differs only in *magnitude* from a *performance goal*, which might use the same format to require a sub-second response time
  - "Response always within milliseconds" vs "Degradation accepted and response within a minute at worse"

Study at home

Type of security requirements

- **Obligations** (e.g., for auditability)
  - "Invocation of a function should be logged securely before execution starts"

Only for your info

Security requirements revisited

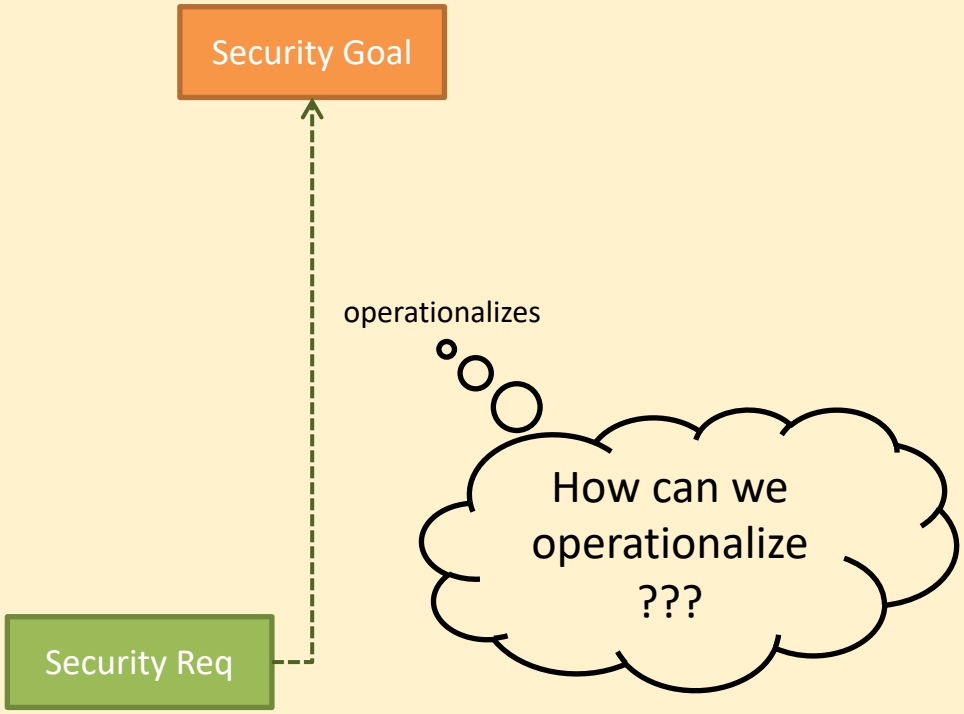
- Constraint-like* security requirements are preventative measures
  - I.e., avoid attack altogether

- What about mitigation** techniques?
  - I.e., monitor for attack and take reactive actions
  - It's a relaxed (less stringent) version of the requirement due to feasibility reasons (technical, money)

* But remember, we have obligations too

** Mitigation: not avoiding the risk, but rather dealing with the aftermath

Share your opinion



HOW TO DISCOVER SECURITY REQUIREMENTS ?

More on
threat analysis in
upcoming lecture

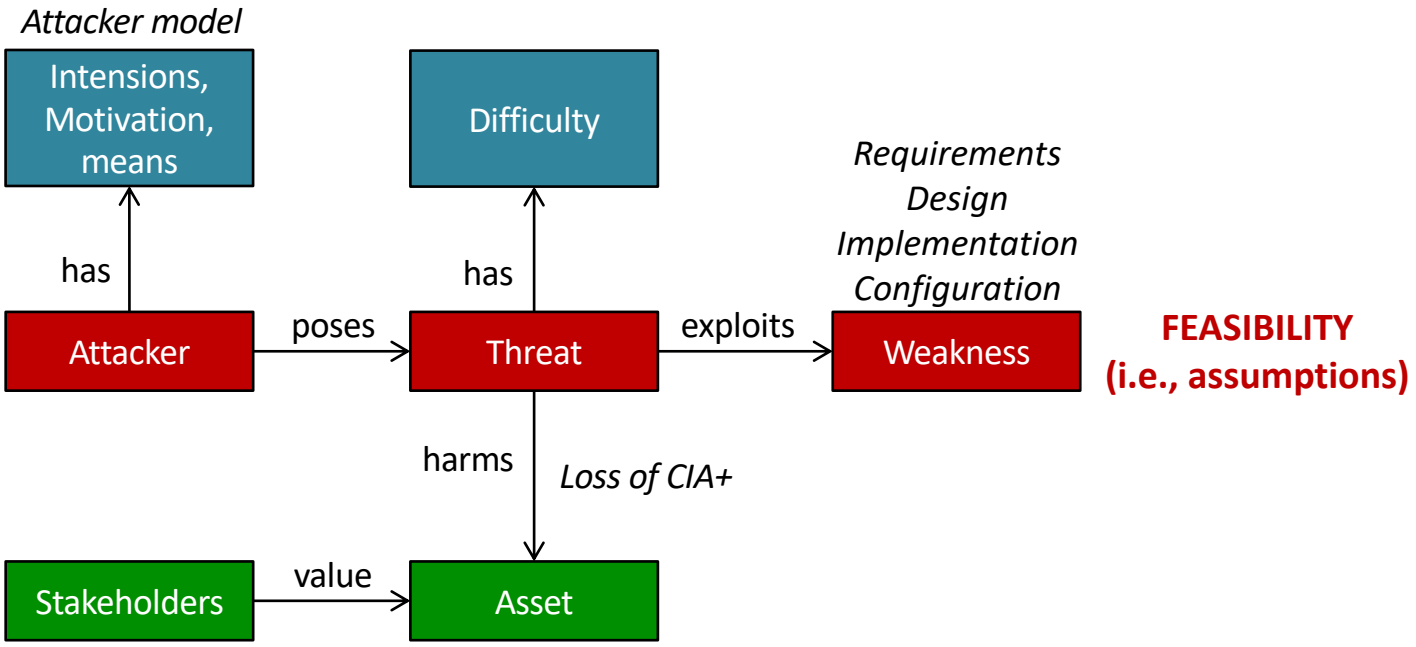
Approach

- Perform threat analysis to discover threats T_i
  where T_i is a malicious action that causes the harm mentioned in the security goal

- $SR_k = \neg(T_i + T_j \ldots)$
  i.e., security requirements are the negation (avoid) of the identified threats
- Not a one-to-one match
  - One SR can cover multiple threats
  - Same threat can be covered by multiple SRs

Harms, threats

- Harm refers to the impact
  - **Attacker-neutral** (mostly)
- Threat refers to the causes
  - **Attacker-based** (e.g. insider or outsider)



Misuse Cases

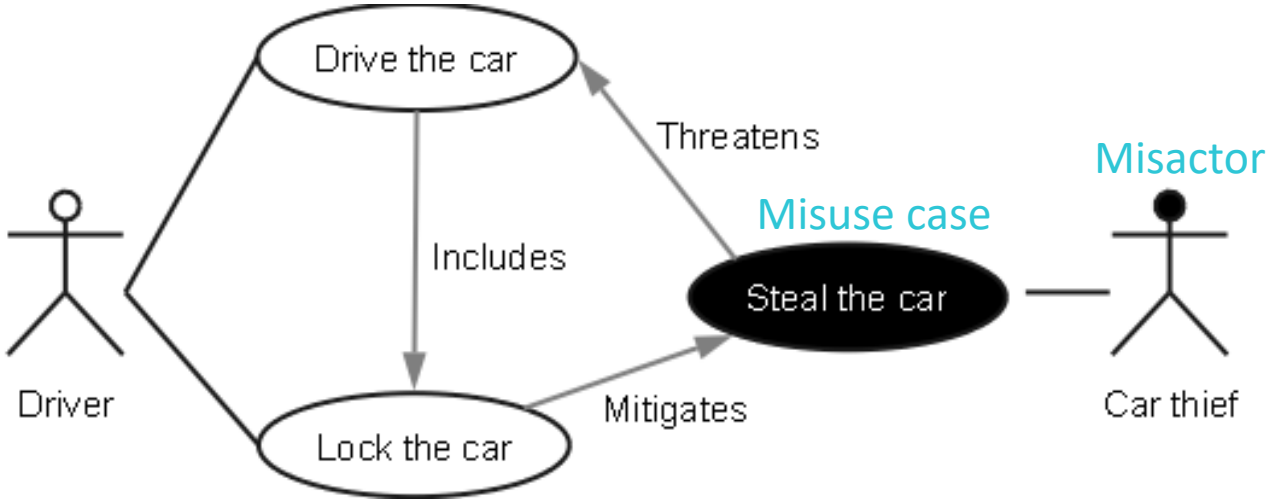
- Way of performing threat analysis (at requirements level) by **anticipating abnormal behaviour** and deriving security requirements
- Misuse Cases: They represent actions that systems should prevent

  → Extension/adaptation of **use-cases** and the corresponding notation.
- Use Cases: Identify the **individual interactions** between the system and its users or other systems. Documented through *Use Case Diagrams*:

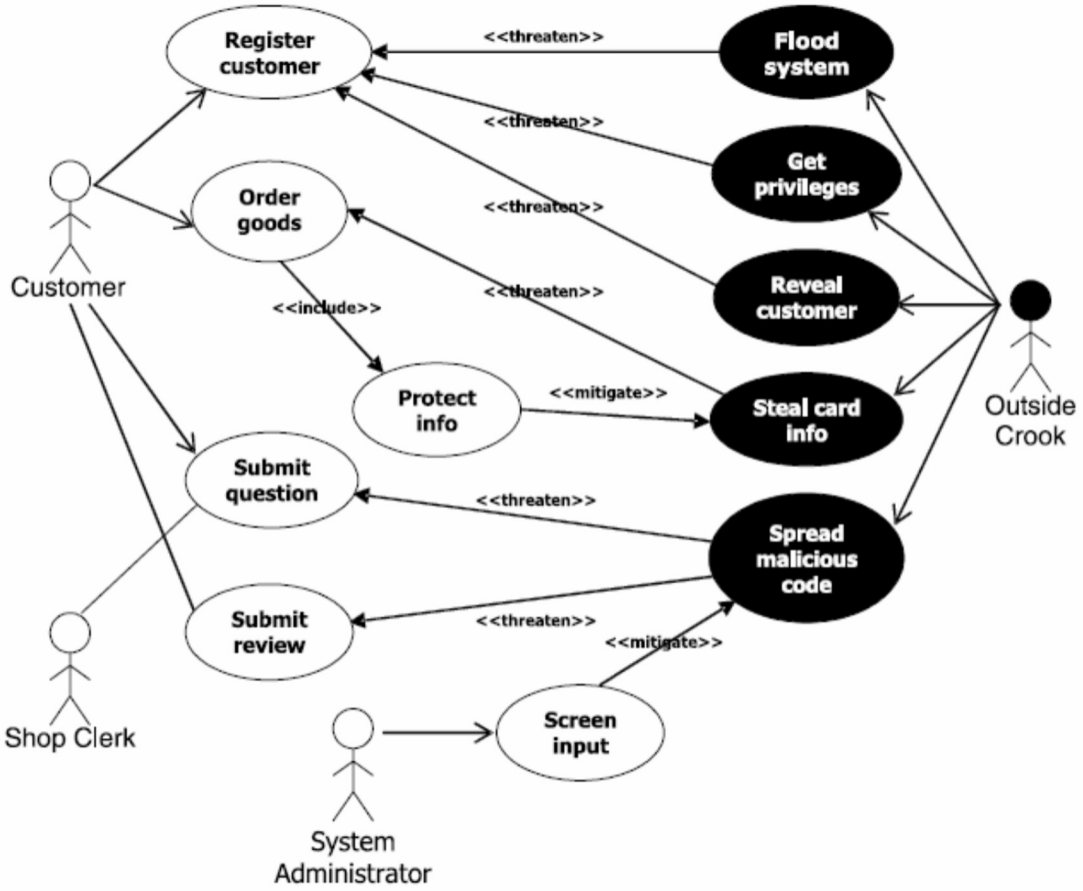
  → In its simplest form, a use case is shown as an ellipse with the actors involved in the use case represented as stick figures.

Misuse case (MUC)

- A misactor is the inverse of an actor support
  - An actor that the system should not
- A misuse case is the inverse of a use case
  - A misuse case threatens a system functionality, it's a functionality that the system should not allow
  - New functionality is introduced to mitigate the threat



Misuse Cases



"Misuse Case" is an **intentional violation** of the system by a "Mis-Actor". Misuse Cases analyze user/actor threats to the system.

The association between a misuse case and a use case can either be a ***threatens*** or a ***mitigates*** relationship.

Example question

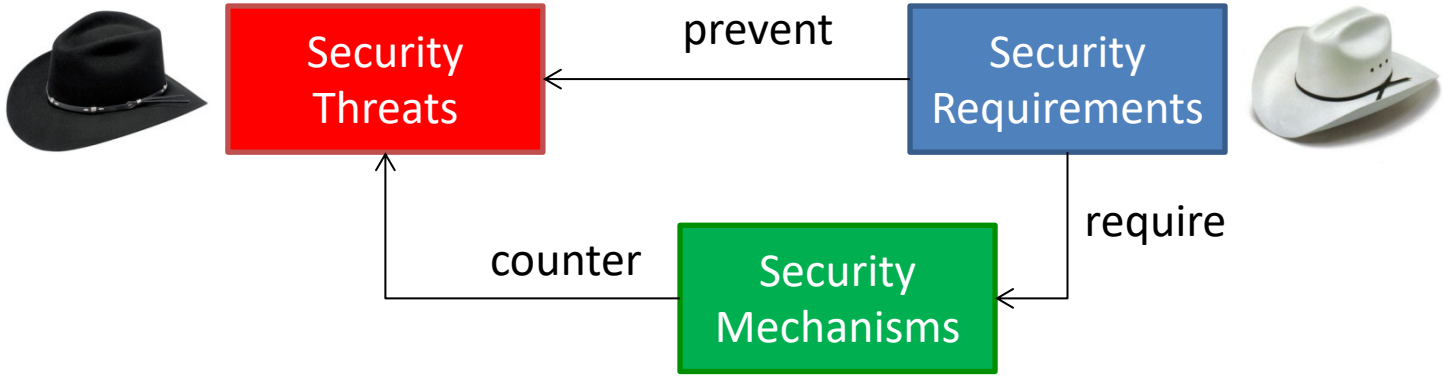
- What is the outcome of a threat analysis with MUCs?
- Difference between threat analysis with MUCs and with STRIDE?
- Can STRIDE threats be uses to derive SRs?

You need to study the lecture on STRIDE first ;)

HOW TO PRIORITIZE SECURITY REQUIREMENTS ?

How to set priorities?

- Security requirements linked to threats



- If we can attach an rank to the threats, we can prioritize the security requirements

Ranking via risk assessment

- Ranking can be obtained via risk assessment

- Upcoming lecture ☺

Learning objectives: checkpoint

- What are security goals and security requirements ?
  - ***Goal: protection of asset from harm***
  - ***Requirement: constraint or obligation to avoid threats***
- How to elicit security requirements ?
  - ***Via threat analysis (e.g. via MUC)***
- How to prioritize security requirements ?
  - ***Risk = Impact on assets x Likelihood of threats***