SSE-LAB 1: Security Requirements

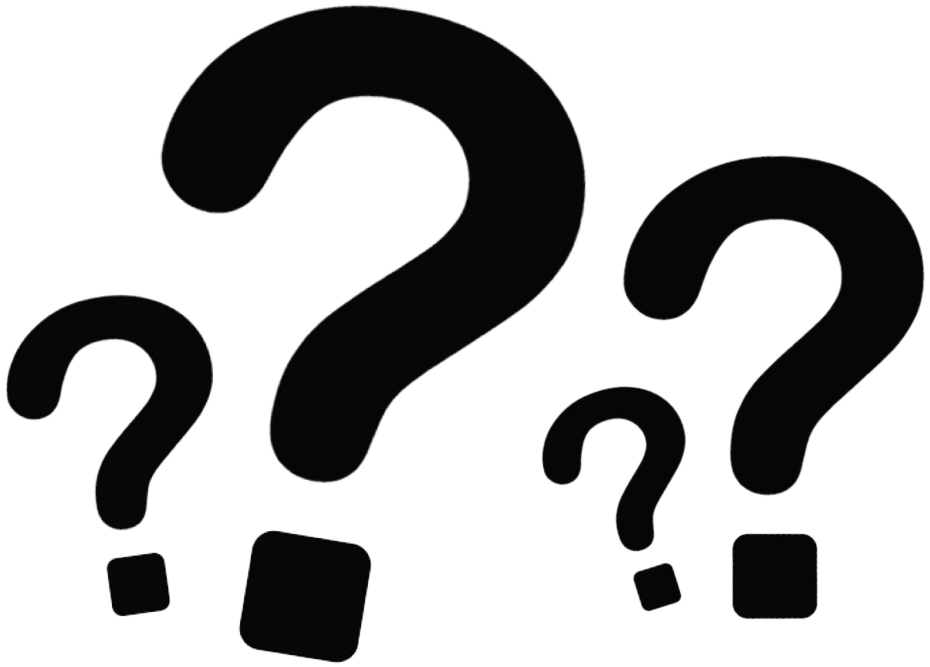
Riccardo Scandariato
Institute of Software Security, TUHH, Germany
ric***do . scanda***to @ tuhh.de

Lecturer: Nicolás Díaz Ferreyra

Agenda

- Brief conceptual review
- Practical case
- Elicitation of security goals
- Elicitation of security requirements
- Elaboration of Misuse and Abuse Cases
- Instantiation of security specification patterns (KAOS)

Security Goals & Requirements



How can we write security requirements?

Security Goals & Requirements

There is no good answer to that question!

- It means evolving from “art” to engineering…
- No particular methodology has yet achieved dominance.
- We will follow a **lightweight process**.

I - Preliminaries (*functional requirements*)

Before starting to outline security requirements we need to identify what the system is and does:

- **SECURITY** = *"the system does what it's supposed to do and **nothing more**!"*
- Hence, we first need to extract the system's **functional requirements** from its specification.

User stories are short descriptions of software features written from the perspective of the end user:

- They can help us identifying the **assets** of a system.

Assets are things or entities **of great value** that must be properly secured.

I - Preliminaries (*functional requirements*)

Remember: Functional requirements must be **testable**

- ✓ It should be possible to determine whether the system satisfies the requirement or not.
- ✓ Security requirements should be testable as well.

Case study: The "Metaverse"

Lab tasks (I):

- Read the case study description.
- Create and discuss user stories.
- Use the following pattern as a guide:

As a <user> I can perform <action>, so that <purpose>

Example: example goes here

II - Security Goals (*Harm analysis*)

The goal of system security is to protect assets from harms.

- Harms occur when an action adversely affects the value of an asset.
- Harms are usually related to some of the **CIA+** security concerns:
  - **CONFIDENTIALITY**
  - **INTEGRITY**
  - **AVAILABILITY**
  - **ACCESS-CONTROL**
  - **NON-REDUPIATION**

Security goals can be extracted though **a harm analysis**.

1. Identify concerns as triples of the form {*action, asset, harm*}
   - *"What harm could come to [asset] from an action violating [concern]?"*
2. Derive security goals by negating the concerns → **"AVOID" GOALS**

II - Security Goals (*Harm analysis*)

Security goals can be found within **organization-wide policies**:

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

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

Lab Tasks (II):

- Identify the assets from the user stories
- Conduct a harm analysis
- Identify "avoid' goals.
- Identify "achieve" goals.

III - Security Requirements (*Threat analysis*)

1. Perform a threat analysis to discover threats $\mathbf{T_i}$, here T_i is **a malicious action** that causes the harm mentioned in the security goal.
2. $\mathbf{SR_i = \neg T_i}$ → Security requirements are the negation ("avoid") of the identified threats.

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

⇒ Misuse cases: A way of performing threat analysis by **anticipating abnormal behavior** and deriving security requirements.

Lab Tasks (III):

- Identify potential attackers of the system under analysis (*Metaverse*).
- Elaborate the misuse cases and extract security requirements.

IV - Security Specification Patterns

Goal *Avoid* [SensitiveInfoKnownByUnauthorizedAgent]
FormalSpec $\forall$ ag: Agent, ob: Object
$\neg$ Authorized (ag, ob.Info) $\Rightarrow$ $\neg$ KnowsV_{ag} (ob.Info)

CONFIDENTIALITY

Goal *Maintain* [PrivateInfoKnownOnlyIfAuthorizedByOwner]
FormalSpec $\forall$ ag, ag': Agent, ob: Object
KnowsV_{ag} (ob.Info) $\wedge$ OwnedBy (ob.Info, ag') $\wedge$ ag $\neq$ ag'
$\Rightarrow$ AuthorizedBy (ag, ob.Info, ag')

PRIVACY

Goal *Maintain* [ObjectInfoChangeOnlyIfCorrectAndAuthorized]
FormalSpec $\forall$ ag: Agent, ob: Object, v : Value
ob.Info = v $\wedge$ $\mathbf{o}$ (ob.Info $\neq$ v) $\wedge$ UnderControl (ob.Info, ag)
$\Rightarrow$ Authorized (ag, ob.Info) $\wedge$ $\mathbf{o}$ Integrity (ob.Info)

INTEGRITY

Goal *Achieve* [ObjectInfoUsableWhenNeededAndAuthorized]
FormalSpec $\forall$ ag: Agent, ob: Object, v : Value
Needs (ag, ob.Info) $\wedge$ Authorized (ag, ob.Info)
$\Rightarrow$ $\Diamond_{\leq d}$ Using (ag, ob.Info)

AVAILABILITY

IV - Security Specification Patterns

Notes:

- "P⇒ Q" means "□(P→ Q)" where the temporal operator "□" means "in every future state" and "→" denotes logical implication.
- "o" means "in the next state".
- "◊≤d" means "some time in the future within d time units".

Lab Tasks (IV):

- Compute **security goals (SGs)**
  - Instantiate the specification patterns.
- Compute the **anti-goals (AGs)**.

Questions ?