

In this write-up, I will talk about misuse cases and steps to identify security requirements. Ivar Jacobson while working on large telecommunication systems introduced use cases. According to him use cases describe system's desired behavior in the form of a story ('Scenario') from the point view of a user or interfacing system('Actor') and supported by subsidiary scenarios in the form of alternatives and exceptions [Jacobson 1992]. On the other hand misuse cases are the inverse of use cases. The concept was coined in 1990s by Guttorm Sindre of the Norwegian University of Science and Technology, and Andreas L. Opdahl of the University of Bergen, Norway. The basic concept is describing the steps of performing a malicious act against a system, just as you would describe an act that the system is supposed to perform in a use case. So, use cases models the behavior expected from the system and misuse cases models the behavior not expected from the system.

### Misuse cases – a Brief Description

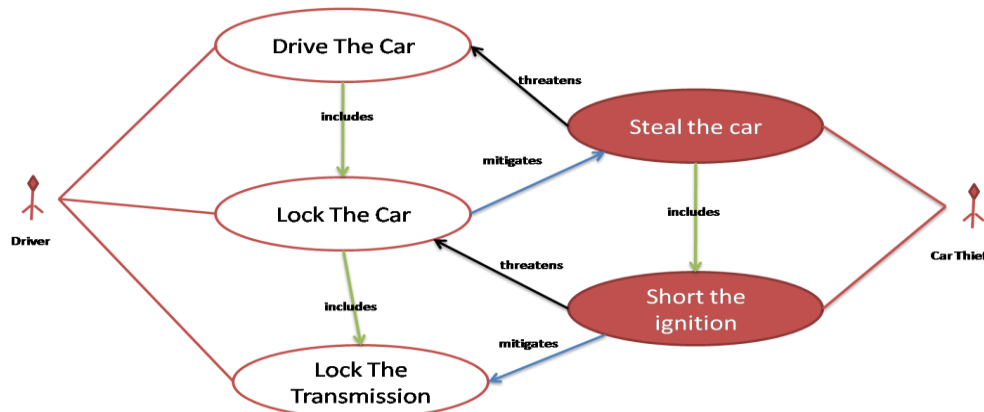
Before going in the details of misuse cases one should understand what a Negative Scenario is?. A Negative Scenario is a scenario that is not desired by the organization in question but is desired by a hostile agent (not necessarily human). According to Sindre & Opdahl 'Negative scenario' is a situation which system users desire not to occur and it is applied in a Use Case context to create the idea of a Misuse Case [Sindre & Opdahl 2000]. It helps one to identify threats which when posed often lead to new set of requirements expressed in the form of a use case.

A typical misuse case diagram is always drawn with a corresponding use case diagram. The misuse case models make use of relation types found in the use case model; include, extend, generalize and association. In addition, it introduces two new relations to be used in the diagram:

**Mitigates:** A use case mitigates the chance of successful completion of the misuse case.

**Threatens:** A misuse case can threaten the chance of successful completion of a use case.

Apart from the relation types a new concept of 'Misuser' is also introduced. He is the one with hostile intent and initiates the misuse case intentionally or unknowingly. A typical use case-misuse case model will look like something shown below:



It is argued that this modeling has several strengths:

1. **Non-functional requirements:** Using misuse cases help capture non-functional requirements upfront and not at the end when projects have gone far ahead. It treats non-functional requirements in the same way as functional requirements.
2. **Design Decisions:** It avoids premature design decisions as security is confronted upfront.
3. **Communication:** Good way of enhancing the communication between the developers and the stakeholders, and is thus a valuable tool to use for verifying that the developers and the stakeholders agree on critical system solutions. This is in particular true regarding Trade-off analysis.
4. **Ease of requirements capture:** Creating misuse cases will often trigger a chain reaction which makes it easier to identify both functional and non-functional requirements. The discovery of a misuse case will often lead to the creation of a new use case as a counter measure (which in turn might be subject for yet a new misuse case).
5. **Resemblance with UML:** It relates very well to both Use Case and UML, making employing the model more seamless than what would be the situation with many other tools.

#### Functional and Non Functional Requirements Interplay

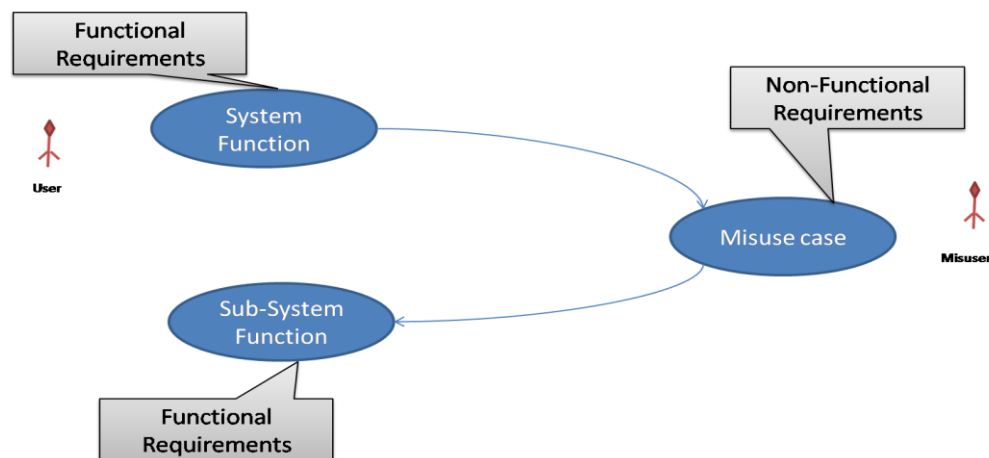
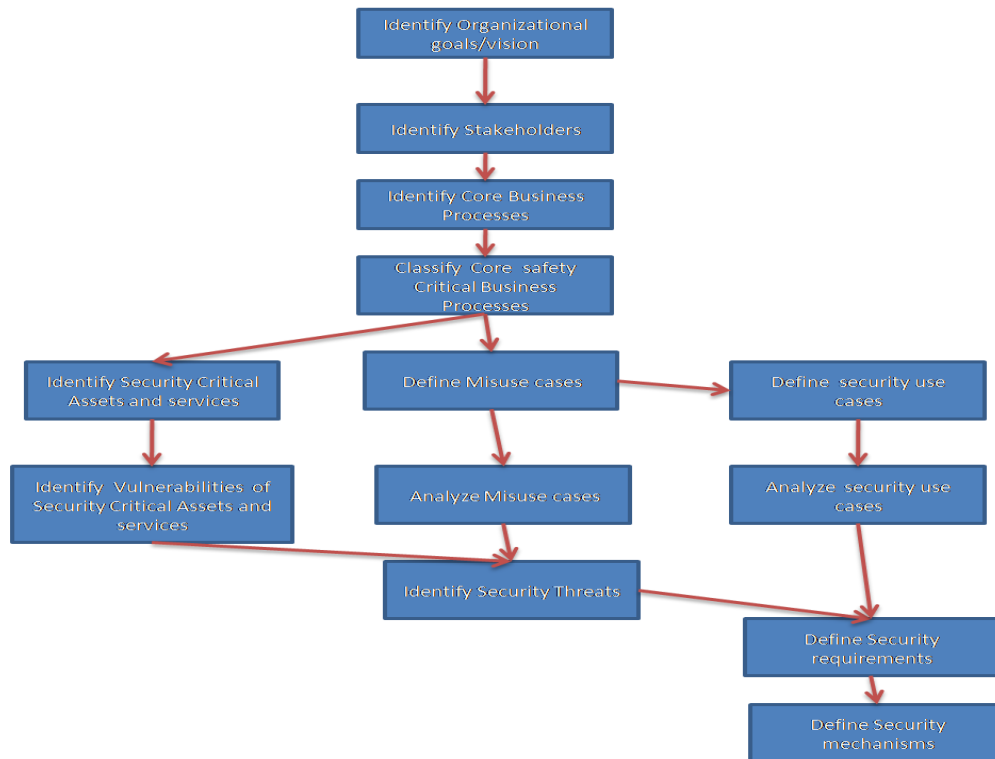


Figure above shows how Functional and non-functional requirements inter-play. A typical misuse case is the one which points out threats which in turn derive non-functional requirements like security requirements. To mitigate such threats a designer will create a sub-functional block which in turn are derived from functional requirements to mitigate the threat. In short, both use case and misuse cases are used to elicit functional and non-functional requirements.

## Steps for getting security requirements through Misuse cases



A Business Analyst should define following steps to gather security requirements:

**1. Identify Organizational Structure/Goals/Vision :** Any requirements gathering exercise starts with the awareness about the client organization. This step is about knowing about the organization, their people, culture, hardware, software, primary goals, mission, vision, values etc.

**2. Identify Stakeholders:** Stakeholders are the most crucial factors in any software development as timely and effective consultation of relevant stakeholders is of paramount importance in requirements elicitation. They are the prime source of requirements. This step deals with their identification and capturing their expectations.

**3. Identify Core Business Processes:** In this step the analyst identifies the core business processes of an organization that plays crucial role in proposed system's behaviour. They can be both manual and automated.

**4. Classify core safety critical business process:** Out of the processes identified in step 3, analyst should classify safety critical processes out of it. Safety critical processes are the

processes that can pose threat to the proposed system if altered/ deviated from proposed behaviour by any means possible.

**5. Identify Security Critical assets and services:** Now this steps is derived from the findings of step 4. These are the critical resources of the organization that are consumed by safety critical processes.

**5.1 Identify Vulnerabilites:** This step is for identifying the weak points in safety critical assets and services.

**6. Define/Analyze Misuse cases :** They are defined and analyzed in consultation with security teams keeping the various negative scenarios as mentioned above.

**7. Define/Analyze Security Use Cases:** This step is done in consultation with security teams for defining security related use cases of the system. These use cases are identified in a way to mitigate the threats posed by the mis use cases that are identified in step 6.

**8. Define Security Requirements/mechanisms :** These are defined based on the findings of step

To sum up, having a negative aspect in requirements elicitation helps in making requirements more complete, covering the security ground well. This in turn help in more robust and secure system development.

Thanks !

Manish Kumar  
1704.manish@gmail.com