# Chapter 3
# Interplay of Security Requirements Engineering and Reverse Engineering in the Maintenance of Undocumented Software

**Andrea Herrmann**
*University Heidelberg, Germany*

**Ayse Morali**
*Ascure N.V., Belgium*

## ABSTRACT

*During software maintenance of security-relevant software, before changes are made, the impact analysis must be able to answer two questions: (1) How do changes in the (security) requirements affect the system´s architecture and code? (2) How do changes in the code affect the system´s requirements satisfaction, e.g. security?*

*A precondition to answer these questions reliably is that both requirements and code are documented and these models are linked traceably to each other. However, in practice, this often is not the case. In a situation where parts of this documentation are missing or outdated, in a first step, one needs methods for reconstructing and linking the models to each other. Such methods are known in the domains of requirements engineering and reverse engineering.*

*Requirements engineering and reverse engineering are two separate arts so far. However, as both solve parts of the same problem, we expect that combining methods from both domains can even better support the software documentation reconstruction and thus the impact analysis of security-relevant systems.*

*The contribution this chapter makes is to provide a literature overview on security requirements engineering methods as well as reverse engineering methods and to provide for categories to discuss how one can choose and link such methods to answer the two impact analysis questions.*

DOI: 10.4018/978-1-61350-438-3.ch003

## INTRODUCTION

Software maintenance includes the modification of a software system after its delivery in order to adapt it to changed requirements or to a changed environment or for the fault-correction (IEEE, 1998). This means that the requirements as well as the code evolve. It is important that before changes are made, their impact is analyzed, especially for security-relevant software.

The impact analysis must be able to answer two questions: (1) How do changes in the (security) requirements affect the system´s architecture and code?, and (2) How do changes in the code affect the system´s requirements satisfaction, e.g. security?

A precondition to answer these questions reliably is that both requirements and code are documented and these models are linked traceably to each other (Pfleeger and Bohner, 1990). These documentations must be complete and up-to-date. However, in practice, this often is not the case, as a survey has shown (Schier and Herrmann, 2010). In a situation where parts of this documentation are missing or outdated, in a first step, one needs methods for reconstructing and linking the models to each other. Only the second step can be to use these models for impact analysis and all other activities of model-driven software development.

So far, the analysis of security requirements engineering and reverse engineering play different roles during software maintenance:

The requirements engineering of security requirements identifies and models – in the form of requirements - what security means for different stakeholders, whether an existing software system is secure (i.e., whether it satisfies the security requirements) and how the security of this system can be improved. As security requirements are and must be specified from different viewpoints (like manager, user, architect and developer perspective), we claim that security requirements engineering can and should link these perspectives to each other (see Figure 1). Reverse engineering "is the process of analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction" (Chikofsky and Cross II, 1990).

Requirements engineering and reverse engineering are two separate arts so far. (For instance, authors of requirements engineering articles do not cite reverse engineering articles and vice versa.) Requirements engineering belongs to the forward engineering process and usually proceeds top-down – from business objectives to user requirements. Reverse engineering vice versa proceeds bottom-up – from code to architecture models – as illustrated in Figure 1. However, as both solve parts of the same problem, we expect that combining methods from both domains can even better support the software documentation reconstruction and thus the impact analysis of security-relevant system.

In what follows, we focus on *security requirements*, where we believe that the topics treated in this chapter are most relevant. Also, we want to focus the literature overview, because security requirements engineering uses specific models which are often requirements engineering models adapted to security. Nevertheless, we expect that the framework we develop for to combine requirements engineering and reverse engineering methods can be applied to other software properties also, not only to security.

This chapter – based on the state of the art – discusses how methods from both disciplines can be combined during software maintenance in order to support the impact analysis. This chapter has the following structure: First, we start with some definitions needed in this chapter. Then, we develop categories which help to classify methods for the impact analysis and the model preparation for it. The subsequent two subchapters describe and classify the state of the art of security requirements engineering and reverse engineering methods. Finally, we summarize the state of the art and discuss how these methods can be chosen