Chapter 3
A Pattern-Based Method to Develop Secure Software

Holger Schmidt
Technical University Dortmund, Germany

Denis Hatebur
University Duisburg-Essen and ITESYS Institut für Technische Systeme GmbH, Germany

Maritta Heisel
University Duisburg-Essen, Germany

ABSTRACT
The authors present a security engineering process based on security problem frames and concretized security problem frames. Both kinds of frames constitute patterns for analyzing security problems and associated solution approaches. They are arranged in a pattern system that makes dependencies between them explicit. The authors describe step-by-step how the pattern system can be used to analyze a given security problem and how solution approaches can be found. Afterwards, the security problems and the solution approaches are formally modeled in detail. The formal models serve to prove that the solution approaches are correct solutions to the security problems. Furthermore, the formal models of the solution approaches constitute a formal specification of the software to be developed. Then, the specification is implemented by generic security components and generic security architectures, which constitute architectural patterns. Finally, the generic security components and the generic security architecture that composes them are refined and the result is a secure software product built from existing and/or tailor-made security components.

INTRODUCTION
It is acknowledged that a thorough requirements engineering phase is essential to develop a software product that matches the specified requirements. This is especially true for security requirements.

We introduce a security engineering process that focuses on the early phases of software development. The process covers engineering of security requirements, security specifications, and security architectures. The basic idea is to make use of special patterns for security requirements analysis and development of security architectures.
Security requirements analysis makes use of patterns defined for structuring, characterizing, and analyzing problems that occur frequently in security engineering. Similar patterns for functional requirements have been proposed by Jackson (2001). They are called problem frames. Accordingly, our patterns are named security problem frames. Furthermore, for each of these frames, we have defined a set of concretized security problem frames that take into account generic security mechanisms to prepare the ground for solving a given security problem. Both kinds of patterns are arranged in a pattern system that makes dependencies between them explicit. We describe how the pattern system can be used to analyze a given security problem, how solution approaches can be found, and how dependent security requirements can be identified.

Security specifications are constructed using the formal specification language CSP (Communicating Sequential Processes) by Hoare (1986). We present a procedural approach to construct formal CSP models for instances of security problem frames and concretized security problem frames. These models serve to formally express security requirements. Afterwards they are used to formally prove a refinement between the CSP model of a security problem frame instance and a corresponding CSP model of a concretized security problem frame instance. This refinement must preserve the security requirements to ensure that the constructed specification realizes the security requirements.

Once we have shown that the selected generic security mechanisms solve the security problems, we develop a corresponding security architecture based on platform-independent generic security components and generic security architectures. Each concretized security problem frame is equipped with a set of generic security architectures that represent the internal structure of the software to be built by means of a set of generic security components. After a generic security architecture and generic security components are selected, the latter must be refined to platform-specific security components. For example, existing component frameworks can be used to construct a platform-specific security architecture that realizes the initial security requirements.

The rest of the chapter is organized as follows: First, we introduce problem frames and present a literature review. Second, we give an overview of our security engineering process. Then we present the different development phases of the process in detail. Each phase of our process is demonstrated using the example of a secure text editor application. Finally, we outline future research directions and give a summary and a discussion of our work.

**BACKGROUND**

In the following, we first present problem frames and second, we discuss our work in the context of other approaches to security engineering.

**Problem Frames**

Patterns are a means to reuse software development knowledge on different levels of abstraction. They classify sets of software development problems or solutions that share the same structure. Patterns are defined for different activities at different stages of the software life-cycle. Problem frames by Jackson (2001) are a means to analyze and classify software development problems. Architectural styles are patterns that characterize software architectures (for details see Bass & Clements & Kazman, 1998) and (Shaw & Garlan (1996)). Design patterns by Gamma, Helm, Johnson, and Vlissides (1995) are used for finer-grained software design, while idioms by Coplien (1992) are low-level patterns related to specific programming languages.

Using patterns, we can hope to construct software in a systematic way, making use of a body of accumulated knowledge, instead of starting from scratch each time. The problem frames defined by...