Towards Automatic Security Scenario Generation

Abdulaziz Alkussayer William H. Allen
Department of Computer Sciences
Florida Institute of Technology
Melbourne, FL USA

Introduction

Intuitively, designing software with security in mind will produce a more secure architectural design and eventually more secure software. Yet, it is still unclear how to conduct and evaluate this process. A good architectural design is one that is able to perform certain tasks (i.e. functionalities) and exhibit certain properties (e.g. security) [1]. Quality attributes of a software system such as modifiability, portability, maintainability and usability, have matured more than security. Many of these attributes can be assessed during the design phase using scenario-based software architecture analysis methods.

In a scenario-based assessment, a set of scenarios is developed that conveys the actual meaning of the requirement. These scenarios are organized into a profile which can be used to evaluate the software architecture. Security scenarios describe the permitted, and hence not permitted, architectural security exposure of the system. The effectiveness of the technique largely depends on the accuracy of the representative scenarios comprising the profile [2].

Dealing with security, because of its nature, is different from dealing with other quality attributes. Our approach generates a set of security-oriented scenarios that address three critical factors: the stakeholder's requirements (*the security objective*), the potential threats (*the problem*) and patterns (*the solution*). By doing so, we can address representative scenarios in the security profile as a three-element tuple {requirement, threat, patterns}.

Background

Swiderski et al. [3] suggests the use of well defined threat profiles during architectural design and coding. However, it is not clear how an architect would make use of threat profiles as an assessment instrument. Moreover, there is little existing research regarding the way that a given software architecture deals with a threat that is not mitigated and, as a result, becomes a vulnerability that can be compromised by an attacker.

On the other hand, neither security requirements nor abuse cases that can be used as a way of explaining security requirements are sufficient to construct a security scenario; this is because the term `abuse case' in its basic sense describes an abnormal (malicious) usage of system functionalities, i.e., a threat to the system in the context of a functional requirement. Even in cases where security requirements have been defined using a sound methodology (such as the one described in [4]), depending on the security requirements alone to assess the security of software architecture is problematic.

In this context, Rozanski et al. [1] define a quality-based scenario that can be used to capture concerns about different quality attributes. This scenario is general and hence can be used to identify most of the quality attributes. However, the fact that it doesn't address the potential threats to the system makes it less useful in describing security concerns.

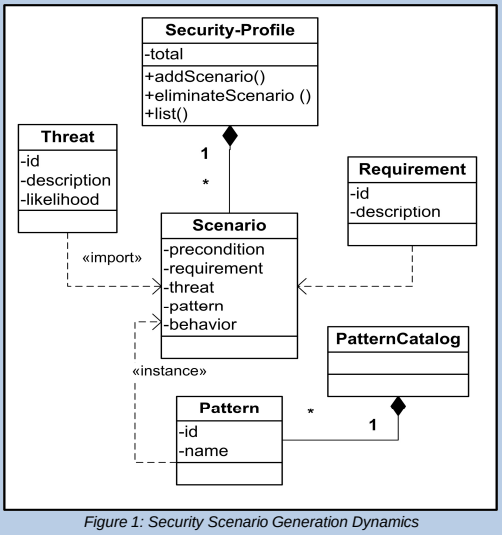

Figure 1: Security Scenario Generation Dynamics

Scenario ID: SS-002

{ Precondition }
" *the customer representatives login page is publicly accessible and the deployment environment is working normally..."*

{ Requirement }
Req.003: "*the system shall enforce authentication of customers and users at every entry points in a secure manner…"*

{ Threat }
Thr.004: " *a malicious attacker brute forces customer care login page to impersonate a legitimate representative…"*

{ Patterns }
Authentication Enforcer, Single Access Point, Check point

{ Behavior }
" *the system should immediately log fatal breach message to the security monitoring console and …"*

Figure 2: Sample Template (eShop system)

The Template

In order to generate a coherent security scenario an architect needs to closely consider the concurrent security engineering activities; namely, threat modeling and security requirements. Figure 1 depicts the process of instantiating a security scenario template. Figure 2 shows a sample security scenario example where the three elements of the security scenario tuple (requirement, threat, patterns) are depicted in *red*.
The following is an explanation of the five constituent elements of the security scenario template.

Precondition
The description of possible system constraint(s) that cause the security scenario to occur.

Requirement
The specific security requirement [4] that describes the required security property of the system. It is important to be able to trace a security scenario back to its constituent requirement because in the end the design decision depends on the stakeholders' understanding of different trade-offs. This traceability aids in reaching that level of understanding.

Threat
A description of the threat to the system in which it explicitly (directly) violates the requirement or implicitly (indirectly) leads to a violation. The threat must be imported from the threat profile [3] and not artificially created.

Patterns
The set of patterns that should be incorporated in order to mitigate the corresponding threat and safeguard the required behavior. Selecting the right pattern is not an easy task, particularly because different catalogs may contain similar patterns that are given different names. However, one can make use of well organized repositories such as [5, 6] and possibly the mitigation strategy documented during the threat modeling.

Behavior
The behavior required of the system when a particular scenario is encountered. Note that it is important to focus exclusively on security and avoid describing the system behavior from other quality-attribute perspectives (e.g. performance).

Summary

Security scenarios plays a critical role in assessing the security of software architecture. Hence, careful incorporation of security requirements, threats, and patterns to generate scenarios shall:

- Effectively increase the architect's awareness of security.
- Explicitly address security threats and risk exposure of the system.
- Generate concrete representative scenarios for the security profile.

References

[1] N. Rozanski and E. Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley, 2005.
[2] J. Bosch, Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach. Addison-Wesley, 2000.
[3] F. Swiderski and W. Snyder, Threat Modeling. Microsoft Press, 2004.
[4] M. Ionita, D. Hammer, and H. Obbink, "A framework for security requirements engineering," in SESS'06, 2006.
[5] K. Yskout, T. Heyman, et al., "An inventory of security patterns," tech. rep., Katholieke University Leuven, 2006.
[6] M. Haz, P. Adamczyk, and R. Johnson, \Organizing security patterns," IEEE Software, vol. 24, no. 4, pp. 52{60, 2007.