Towards Automatic Security Scenario Generation

Abdulaziz Alkussayer William H. Allen

Department of Computer Science
Florida Institute of Technology
Melbourne, FL, USA
alkussaa@fit.edu wallen@fit.edu

Abstract

Software security has become a crucial component of any software system in today's market. However, the development of secure software is still a maturing process. Software architecture (SA) assessment methods have gained increasing attention in recent years. Most of these evaluation techniques are scenario-based, and thus depend heavily on the quality of the scenarios selected for their evaluation. In this paper, we present a systematic approach for the generation of coherent security scenarios for software architecture that can be used during a scenario-based security assessment. Our approach effectively incorporates security requirements, patterns and threats into a concrete security scenario. The work in this paper is part of on-going research to define a scenario-based security assessment methodology for SA.

Keywords Security Scenario, Architectural Scenario, Security Architecture.

1. 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. It is well-known that software architecture is best used to model and outline the system structures and their inter-components [1, 2]. 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. It has also been acknowledged that a good scenario-based architectural assessment depends mainly on good scenarios [2].

In this context, we believe that the vital security properties of a software architecture can be assessed similarly. In fact, assessing the architecture of a software system to discover the level of support that it provides for security is extremely important; not only to help ensure that the stakeholder requirements (e.g: security objectives) have been met, but also -and more importantly- to reveal any hidden security design flaws that may be overlooked.

Nevertheless, dealing with security, because of its nature, is different from dealing with any 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*}. The security scenario presented in this paper is based on these three factors.

2. Security Scenario

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. The profile can then be used to evaluate the software architecture. Security scenarios describe the permitted, and hence not permitted, architectural security exposure of the system. The scenario-based assessment directly depends on the scenario profile defined for the quality attribute to be assessed. The effectiveness of the technique largely depends on the accuracy of the representative scenarios comprising the profile [1].

Swiderski et al. [3] suggests that threat profiles developed during threat modeling ought to be used during architectural design and coding. Threats are well defined and classified according to the STRIDE model [3]. However, it is not clear how an architect would make use of the threat profile 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. In other words, it describes 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], which provides a better way to define and measure security requirements), depending on the security requirements alone

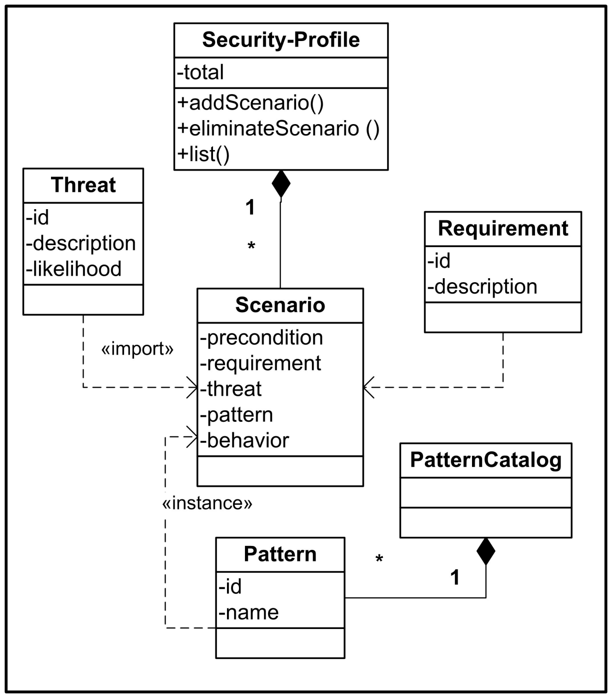

Figure 1. Generating Security Scenario

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 quality-based 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. Thus, we believe a more useful security scenario is one that increases the architect's awareness and explicitly addresses threats to the security of the system. In the following section we will propose a more systematic approach to the generation of security scenarios.

3. Our Approach

As we described earlier, security scenarios are indeed appropriate to describe the required security support of software architecture. 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 construction dynamics of instantiating a security scenario template.

The template consists of five elements: the three elements of the security scenario tuple described earlier plus two more elements: precondition and behavior. It is also important to note that only the three-element tuples {*requirement, threat, patterns*} are needed to create the security profile and carry out the assessment process, the precondition and behavior elements have been added to satisfy the completeness of the scenario template.

- The ***requirement***: is 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.
- The ***threat***: is 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.
- The ***precondition***: is the description of possible system constraint(s) that cause the security scenario to occur.
- The ***behavior***: is 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).
- The ***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.

Clearly, in some cases the same patterns will be referenced in multiple scenarios (e.g. the `Authenticator` pattern [6]). In other cases, multiple patterns may be identified in a single scenario because these patterns complement each other. For example consider a scenario where one requirement is *secure logging*. In this scenario the selected patterns would normally include both the `Audit Interceptor` [5] and the `Secure Logger` [5, 6] patterns.

4. Conclusion and Future Work

We presented a systematic method to generate security scenarios based on security patterns and threats. Also, we have shown how our approach can be easily used to carry out the process of generating scenarios automatically. Currently we are developing a proof of concept example to demonstrate the advantages gained by using our scenario-based architectural assessment of software security. The findings of our experiments along with our assessment methodology will be published in a future paper.

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, R. Scandariato, and W. Joosen, "An inventory of security patterns," tech. rep., Katholieke University Leuven, Department of Computer Science, 2006.

[6] M. Hafiz, P. Adamczyk, and R. Johnson, "Organizing security patterns," *IEEE Software*, vol. 24, no. 4, pp. 52–60, 2007.