Information and Software Technology 51 (2009) 846–864

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

An aspect-oriented methodology for designing secure applications

Geri Georg [a], Indrakshi Ray [a,*], Kyriakos Anastasakis [b], Behzad Bordbar [b], Manachai Toahchoodee [a], Siv Hilde Houmb [c]

[a] Computer Science Department, Colorado State University, 1873 Campus Delivery, Fort Collins, CO 80528, USA
[b] School of Computer Science, University of Birmingham, Edgbaston, Birmingham, UK
[c] Department of Computer Science, University of Twente, Enschede, Netherlands

ARTICLE INFO

Article history:
Available online 13 May 2008

Keywords:
Alloy
Aspect-oriented modeling
Authentication
Secure systems design
Security analysis
UML

ABSTRACT

We propose a methodology, based on aspect-oriented modeling (AOM), for incorporating security mechanisms in an application. The functionality of the application is described using the primary model and the attacks are specified using aspects. The attack aspect is composed with the primary model to obtain the misuse model. The misuse model describes how much the application can be compromised. If the results are unacceptable, then some security mechanism must be incorporated into the application. The security mechanism, modeled as security aspect, is composed with the primary model to obtain the security-treated model. The security-treated model is analyzed to give assurance that it is resilient to the attack.

1. Introduction

Developing secure systems is a non-trivial task. Security standards such as the ISO Common Criteria [28] and risk management standards such as the Australian/New Zealand Risk Management standards [4,5] exist to aid secure systems development. However, these standards generally address system security in the broad sense, and often require extensive resources and expertise to adapt their use to the design of a specific system. These standards also do not address low-level details, such as, how to verify that a system is protected from specific kinds of attacks or how to ensure that a system has a given set of security properties. More importantly, they do not provide a methodology for designing secure systems.

Security mechanisms are typically analyzed in isolation as protocols, and depending on how they are integrated in an application, they may or may not provide adequate protection. In addition, there are often multiple mechanisms that could be used to counter an attack, so choosing a mechanism that best fits design goals may be confusing. It is also the case that solutions to different security concerns may actually conflict, rendering some ineffective against the attack they were supposed to counter. System designers need a way to verify the efficacy of security mechanisms once they have been integrated into an application design, prior to implementation. They also need the ability to include solutions in combination and analyze them against various attacks. In this paper, we propose such a methodology for designing secure applications.

We use aspect-oriented modeling (AOM) techniques [20] in our approach to designing secure systems. Complex software is not developed as a monolithic unit but is decomposed into modules on the basis of functionality. We refer to the models describing functionality as the *primary model*. Security concerns are not limited to one module of the primary model but impacts several of them. For example, an attack typically affects multiple modules. Similarly, a security mechanism that thwarts an attack will have to be incorporated in several modules of the application. The attack and the security mechanisms are localized in a separate model, which we call the *aspect*. Modeling security mechanisms and attack models as aspects has several benefits – it allows designers to understand the attacks and the mechanisms independently, which makes it easier to manage and change these models. Designers can use techniques for composing aspects with the primary model, followed by analysis of the resulting system, to understand the effect of the attack or the effect of the security mechanism on the application. Another advantage is that analyzing using different attack models or different security aspects is easier since all a designer must do is to re-compose the primary model with a new attack model or new security aspect prior to performing a new analysis.

An aspect in our work is similar to the concept of aspects used in other AOM or AOP (Aspect-Oriented Programming) approaches [2,13,14,31,34,57] in that they represent a non-functional concern, e.g., security, and they are cross cutting and must be integrated at different places in the primary model. The differences lie in how

* Corresponding author. Tel.: +1 970 491 7986; fax: +1 970 491 2466.
E-mail addresses: georg@cs.colostate.edu (G. Georg), iray@cs.colostate.edu (I. Ray), K.Anastasakis@cs.bham.ac.uk (K. Anastasakis), B.Bordbar@cs.bham.ac.uk (B. Bordbar), toahchoo@cs.colostate.edu (M. Toahchoodee), S.H.Houmb@ewi.utwente.nl (S.H. Houmb).

0950-5849/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2008.05.004

the aspects are specified, whether they are reusable, and the manner in which the aspects are integrated with the application.

We define two types of aspects: *generic aspects* and *context-specific aspects.* Generic aspects are application-independent and reusable. For instance, an attack pattern can be represented as a generic aspect. Similarly, a security protocol or a security mechanism can be modeled as a generic aspect. An application developer can create his own generic aspect or use an existing one from the library of generic aspects. Generic aspects can be independently analyzed to ensure that the properties of the attack or the mechanism have been adequately captured. Generic aspects must be instantiated in the context of a given application. The instantiation is referred to as a context-specific aspect. We use parameterized Unified Modeling Language (UML) to represent generic aspects. Context-specific aspects are represented as UML models. The instantiation occurs by binding parameters in the generic aspect to elements in the primary model. Specifying aspects using UML allow our approach to be used at different levels of abstraction.

To understand the impact of a security attack on the primary model, it is necessary to compose the context-specific attack aspect with the primary model. The composition produces the *misuse model.* Analysis of the misuse model will help determine whether the protected resources are compromised by the attack. If the results are unacceptable, a security mechanism must be integrated with the primary model. We refer to this model as the *security-treated model.* To understand the efficacy of the security mechanism, the security-treated model is composed with the context-specific attack aspect. The result is the *security-treated misuse model.* The security-treated misuse model is analyzed to ensure that the given attack is mitigated in the security treated model.

Manual analysis is error-prone and tedious. Towards this end, we investigated how this analysis can be partially automated. The tools for verifying UML models, such as, OCLE [47] and USE [25], are useful when we want to check if a specific model instance conforms to the constraints of the model. Although theorem provers are effective for analyzing properties, but they require a lot of expertise and are unlikely to be used by application developers. We chose to use the Alloy Analyzer because it is easy to use and has been used for verifying many real-world applications.

We illustrate the basic operation of our approach using an example e-commerce platform called ACTIVE [17]. ACTIVE provides services for electronic purchasing of goods over the Internet. The IST EU-project CORAS performed three risk assessments of ACTIVE in the period 2000–2003. The project looked into security risks of the user authentication mechanism, secure payment mechanism, and the agent negotiation mechanisms of ACTIVE. Our example consists of the user authentication mechanism of ACTIVE's login service. In order to keep the example tractable, we only show how to apply our methodology to one of its risks and one of the possible treatments for that risk.

The paper makes several contributions. First, it provides a methodology for designing secure applications. Second, it shows how to analyze the impact of a security attack on an application and how effective the security solutions are against a given attack. Third, it allows one to compare the efficacies of the different security solutions with respect to one or more given attacks. Fourth, it shows how to formally analyze a model and get assurance about the security properties. Fifth, it demonstrates feasibility that the approach can be used for real-world applications.

The rest of the paper is organized as follows. Section 2 describes ACTIVE. Section 3 shows an example attack to the login service. We also show how to compose the attack model with the primary model to create a misuse model. Section 4 presents a security mechanism we use to prevent the attack and illustrates how we integrate it with the primary model to create a security-treated model. This section also shows how we generate the misuse model for the security-treated model. Section 5 shows how we can analyze this model to ensure the satisfaction of the security properties. Section 6 discusses related work. Section 7 concludes the paper with some pointers to future directions. The Appendix gives the detailed Alloy models.

2. Overview of our approach

An overview of our methodology is given in Fig. 1. Step 1 analyzes the system to identify the threats to the resources. The inputs to this step are the primary model, possible threats, and the security requirements. Threats become attacks on the system when they compromise protected resources. Since an attack impacts various parts of the primary model, we abstract the specification of the attack in an aspect. To distinguish them from the other aspects used in our work, we refer to them as *attack aspects*. Step 2 involves composing the attack aspects with the primary model to create *misuse* models. Step 3 analyzes the misuse model to understand the impact of the attack. If the results are not acceptable, potential security solutions (or mechanisms) that counter the attack are incorporated into the primary model to obtain the *security-treated model.* The security-treated model is combined with the specific attack to create a *security-treated misuse model.* This is done in Step 4. The security-treated misuse model is analyzed as in Step 3, and if the results are still unacceptable, an alternate security solution must be integrated, and the new security-treated system misuse model re-generated and re-analyzed. When the analysis results are acceptable, a different attack and its potential solutions can be considered. This is done in Step 5. It is important to continue integrating security mechanisms and analyzing the resulting security-treated system against previously considered attack models since some mechanisms may interfere with each other. When such conflicts arise, the designer can integrate alternative solutions until a usable combination is identified through achieving acceptable analysis results. We next discuss each step of the methodology in more details.

2.1. Step 1: Analyze system risk

There are many different risk analyses methodologies that can be used in the first step of the methodology, and we use the CORAS framework [15,17,49]. CORAS is model-based, and uses UML diagrams and textual usage scenarios as part of a risk assessment. This fits well with existing design processes since UML is the de-facto modeling language used in the software industry. CORAS takes advantage of techniques developed for the safety domain, and has a platform of supporting tools. Using CORAS, a portion of the system to be analyzed is identified as the context for the analysis, and assets associated with particular stakeholders are identified within that context. UML use case, static class, and dynamic behavior diagrams are used to specify the system design that we refer to as the primary model.

The CORAS framework use Hazard and Operability (HAZOP) analysis to identify threats to the assets of interest, and Failure Mode Effect Analysis (FMEA) to identify system vulnerabilities. It then uses Fault Tree Analysis, along with the threats and vulnerability analysis results to identify unwanted incidents that can lead to attacks on assets. The consequences and frequencies of these incidents determine the value of the risks with which they are associated. Designers prioritize risks with respect to the system security requirements, and assess potential treatments using these priorities and the risk values.

This detailed assessment identifies the context in which specific attacks could occur and the assets that could be affected. Part of the output of a CORAS analysis is therefore the exact locations in the system design that are vulnerable to attacks and the exact