Weaving Security Aspects into UML 2.0 Design Models

Djedjiga Mouheb, Chamseddine Talhi,
Vitor Lima, Mourad Debbabi, Lingyu Wang
Computer Security Laboratory
Concordia University, Montreal, Canada
{d_mouheb, talhi, v_nune, debbabi, wang}@ciise.concordia.ca

Makan Pourzandi
Software Research
Ericsson Canada Inc.
Montreal, Canada
makan.pourzandi@ericsson.com

ABSTRACT

Security plays a predominant role in software engineering. Nowadays, security solutions are generally added to existing software either as an afterthought, or manually injected into software applications. However, given the complexity and pervasiveness of today's software systems, the current practices might not be completely satisfactory. In most cases, security features remain scattered and tangled throughout the entire software, resulting in complex applications that are hard to understand and maintain. In this paper, we propose an aspect-oriented modeling approach to systematically integrate security solutions into software during the early phases of the software development life cycle. First, we present the security design weaving approach, as well as the UML profile needed for specifying security aspects. Then, we illustrate the approach through an example for injecting the design-level security aspects into base models.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications; D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms

Design, Security

1. INTRODUCTION

Security plays a predominant role in software engineering. But, very often security solutions are added to existing software either as an afterthought phase of the software development life cycle, or manually injected into software code or UML models. However, with the increasing complexity of today's software systems, adding security as an afterthought leads to huge cost in retrofitting security into the software and further can introduce additional vulnerabilities. Besides, because of the pervasive nature of security, adding security manually into a UML design is tedious, may lead to additional security vulnerabilities and security components may become tangled and scattered throughout the whole design. Consequently, the resulting UML design model will most likely become difficult to understand and maintain.

Existing approaches on integrating security concerns into UML design mostly focus on specifying security requirements and sometimes analyzing UML models against the specified requirements (a more detailed review of related work will be given in Section 2). How to systematically enforce the specified requirements remains an open problem. On the other hand, our work is inspired by the AOM methodology. Using AOM, security solutions can be precisely defined and injected into the base models at the matched places. However, to date, there is no standard language to support AOM, nor a standard mechanism for weaving aspects into the base models.

In this paper, we provide an end-to-end approach for systematically weaving security aspects into UML design models. By end-to-end, we mean an approach that starts from specifying the needed security requirements and ends with injecting the corresponding solutions at the appropriate locations in the design models. Our solution encompasses the following: first, we devise UML profiles required for both, capturing security requirements and specifying the corresponding security solutions. Second, we study how to identify join points for the main UML behavioral diagrams as there is no standard definition of UML join points. Third, we propose a weaving procedure for injecting design-level security aspects into base models.

The main contributions of our work are two fold. First, the proposed AOM approach can separate security concerns from the software functionalities. This separation of concerns allows security experts to specify security solutions as aspects including the details on how and where to apply them in the application, and provide these solutions to designers with limited security knowledge. Second, the novel concept of design-level weaving of security aspects allows designers to systematically integrate high-level security solutions into the base models without having to understand the inner working of the security solutions.

The remainder of this paper is organized as follows. Section 2 gives an overview of the related work. Afterwards, in Section 3, we summarize our approach for weaving security aspects into UML design models. Section 4 presents the AOSM profile, the UML join points, and the weaving procedure. A case study is given in Section 5 to illustrate the security design weaving approach. Finally, we conclude the paper and present our future work in Section 6.

AOM'09, March 2, 2009, Charlottesville, Virginia, USA.

2. RELATED WORK

We present in the sequel two categories of related work. First, we present approaches to design weaving of UML models. Second, we summarize the current literature related to integrating security concerns at the design level. Regarding AO modeling using UML, an overview can be found in [10].

Various approaches have been proposed for weaving UML design models, the following is a brief overview of the existing contributions. [11] presents Motorola WEAVR, a tool for weaving aspects into executable UML state machines. Motorola WEAVR supports two types of join points that are action and transition join points. However, this weaver is based on the Telelogic TAU G2 implementation, therefore, it is tool-dependent and not portable. [2] proposes a model weaver for AO executable UML models. The considered join point model intercepts only the interaction between objects. The weaving engine is based on XSLT transformations that manipulate UML models in XMI format. However, this approach targets only executable UML models. [3] presents XWeave, a weaver that supports the weaving of models and meta-models. XWeave is based on the Eclipse Modeling Framework. In contrast to our approach that uses a UML profile, pointcuts used by XWeave are expressed using oAW, an expression language based on OCL.

Regarding the use of AOM for security, few approaches have been published recently. [8] proposes an aspect-oriented approach where the UML meta-model is augmented with new diagrams to represent access control requirements. In contrast to our approach that uses a UML profile to represent security aspects, this approach extends the UML language by defining new artifacts to capture security features. [9] proposes an aspect-oriented approach for modeling access control requirements. This approach models access control aspects as patterns using UML diagram templates [7] that are instantiated by binding elements in the aspect models to elements in the application domain. Then, the aspect models are woven into the base model using a composition algorithm.

Other approaches summarized and evaluated in [6] have been proposed in the last years to integrate security features during the early phases of software development life cycle using UML. The majority of these approaches propose extensions of the UML language using standard UML extension mechanisms to specify security requirements. The evaluation of UML models against the specified requirements is based on automatic verification tools such as model checkers and theorem provers [4]. These approaches are useful attempts for specifying and verifying security requirements on UML design, however, enforcing those requirements on UML design is not their main concern as our approach.

3. APPROACH OVERVIEW

This section illustrates a summary of our approach for weaving security aspects into UML 2.0 design models. The approach architecture is depicted in Figure 1. The main steps of our proposed approach are the following:

1. Specification of Security Requirements: The designer should be able to specify the security requirements that he/she wants to enforce on his/her design. To this end, a UML profile is defined such that security requirements can be attached to UML design elements as stereotypes parameterized by tagged values. The covered security requirements are those commonly specified and verified on software and for which a security solution can be provided as an aspect. Examples of these security requirements are secrecy, authentication, access control, etc. Due to space limitation, the specification of security requirements is not addressed in this paper. The reader can refer to [5] for more details on security requirements specification.

2. Specification of Security Solutions: The security expert provides a security solution as a security aspect for each security requirement covered by the security requirements specification profile. Our solution is based on AOM, i.e., these security solutions are specified using a UML profile for aspect-oriented security modeling (See details in Section 4.1). Once designed by security experts, security solutions are packaged into a security aspects library and later woven into the design model to enforce security requirements.

3. Definition of UML Join Points: A security solution mainly consists of security behaviors (advices in AOP jargon) that should be injected before/after/around some specific points (join points in AOP jargon) of the UML design. Since there is no standard definition of join points for UML design, we provided a precise definition of join points for the main UML diagrams that are involved in design weaving (See Section 4.2).

4. Design Weaving: This represents the actual addition of security solutions into UML design. Based on the security requirements specified by the designer, the corresponding security solutions are selected. Then, the involved security advices are injected into the base model based on the security solutions specification and the join points specified by the security expert and specialized by the designer (See Section 4.3).

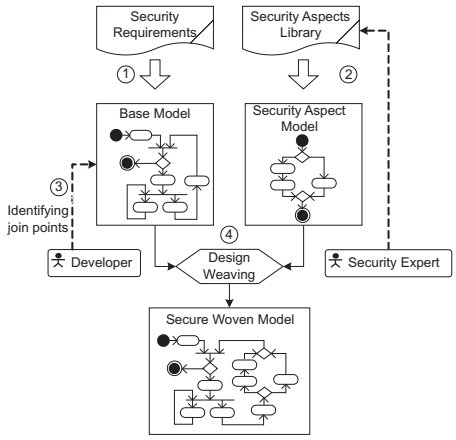

Figure 1: Overview of the Proposed Approach.

4. SECURITY DESIGN WEAVING

This section presents the main components needed by the different steps of our approach. In this paper, we focus on the design-level weaving aspect. In the following, we present the AOSM profile needed for the specification of security solutions for UML design. Next, we provide our definition of UML join points. Then, we explain the weaving procedure.

4.1 UML Profile for Aspect-Oriented Security Modeling (AOSM)

This section presents the meta-model specification of our AOSM profile that allows the specification of security solutions for UML design. This profile is based on aspect-orientation to support the separation of security concerns from the software functionalities.

4.1.1 Aspects Specification

Figure 2 presents the meta-model proposed for the specification of aspects. An aspect is modeled as a stereotyped class. Advices are modeled as special kind of operations stereotyped by the name ≪*advice*≫. The advice behavior is specified in behavioral diagrams. The advice type is given by a tag *type* whose values are provided in the enumeration *AdviceType*. The location where an advice should be injected is specified by the meta-element *Pointcut*. Section 4.1.2 presents the elements needed for specifying pointcuts. Aspects may also introduce new features and relationships to the existing base model. This is specified by the meta-elements *NewMember*, *Generalization*, and *Realization*.

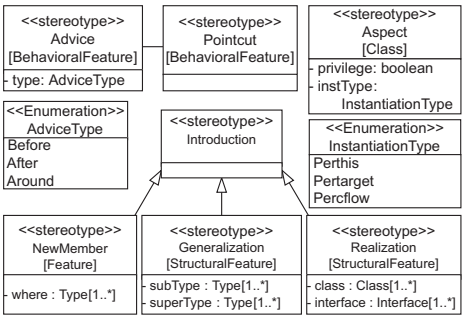

Figure 2: The Meta-Model for Specifying Aspects.

4.1.2 Pointcuts Specification

The meta-model proposed for the specification of pointcuts is presented in Figure 3. The set of specified pointcuts are those that are commonly used by the most popular AOP languages. Since the join points that are matched by the pointcuts are mainly UML actions, activity diagrams are selected to model pointcuts. In addition, activity diagrams offer elements that can represent scope and context information (e.g., activity partitions, action pins) as well as the parameters exposed by the pointcuts at the identified join points that can be represented as activity parameter nodes.

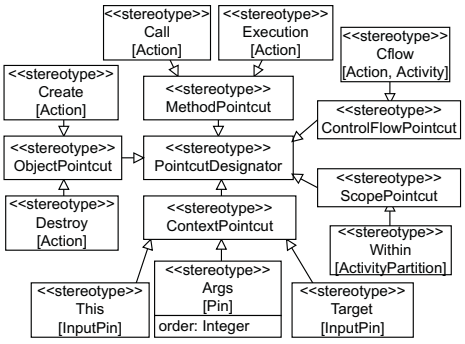

Figure 3: The Meta-Model for Specifying Pointcuts.

4.2 Join Points Identification for UML Design

The set of pointcuts classes that can be specified using our profile is selected such that it represents the common pointcuts designators usually provided in the most popular AOP languages. However, the actual join points that should be matched by the pointcuts belong to the behavioral diagrams specifying the behavior of the software under design. While join points in programming languages are well studied, to the best of our knowledge, there is no complete and well elaborated study of join points for UML. Table 1 summarizes our definition of join points for the main UML behavioral diagrams. For space limitation, we only presented a representative set of join points in Table 1. We will discuss UML join points in details further in a separate paper.

Table 1: Join Points Definition for the main UML Behavioral Diagrams.

Join Point	Activity Diagrams	Sequence Diagrams	State Machine Diagrams
Operation Call	*Call-Operation-Action*	*SendOperationEvent*	Call operations inside states and transition effects
Operation Execution	Execution of the behavior invoked by call actions	*Execution-Specification*	Execution of the behavior invoked by call operations inside states and transition effects
Object Creation	*CreateObject-Action*	*Creation-Event*	None
Object Destruction	*DestroyObject-Action*	*Destruction-Event*	None
Field Reference	*ReadStructural-FeatureAction*	None	None
Field Assignment	*WriteStructural-FeatureAction*	None	None
Exception Handler	*Exception-Handler*	None	None

4.3 Design Weaving

In this section, we show how security advices will be woven into a base model design. The main steps of the design weaving process are the following:

4.3.1 Pointcuts and Aspects Instantiation

During this step, the designer instantiates the generic aspects by choosing the elements of his/her model that are targeted by the security solutions. The pointcuts specified by security experts are chosen to match specific points of the design where security advices should be injected. Since the security solutions are provided as a library of aspects, pointcuts are specified as generic patterns that should match all possible join points that can be targeted by the security solutions. However, a specific design can involve elements that should be targeted by the security solutions while named by the designer following a naming convention that is not expected by the security expert. For this reason, the designer is given the opportunity to choose manually such elements.

4.3.2 Join Points Identification

During this step, the actual join points where the security advices should be applied are identified and linked to the corresponding advices. The elements selected by the designer in the first step are elements of the class model. However, the actual join points where the weaving is done belong

mainly to the behavioral diagrams specifying the behavior of the software/system under design. From the elements selected by the designer during the first step of the weaving, and following our join points definition, actual join points are identified in the base model. The identified join points are then linked to their corresponding advices by applying stereotypes named with the advices to these join points.

4.3.3 Advices Injection and Actual Weaving

During this step, the advices are woven into the base model at the identified locations according to the specification of the security solution. Actually, depending on the target diagram, the advice behavior can be packaged into composed elements. For instance, in activity diagrams, the advice behavior can be encapsulated in a *CallBehaviorAction* [7] that directly invokes the advice behavior. Injecting the advices behavior as composed elements hides the complexity of the advices to the designer. This solution helps keeping the design as simple as possible by reducing the number of elements directly added to the base model.

5. CASE STUDY

In this section, we illustrate our security design weaving approach through an example shown in Figure 4. The example depicts the process of a URL request from a user to a Web server through an unsecure communication channel. After the user initiates a request for accessing a Web page, the Web server may require authentication from the user depending on the requested URL. In this case, the user must send his credentials to the Web server. If the user's credentials are valid then the Web server will provide the user with the requested page. Otherwise, the user's request will be rejected. However, the base model shown in Figure 4 specifies this functionality without any security mechanism. In the following, we show how the designer can add SSL (Secure Sockets Layer) [1] secure communication to his/her application by weaving an SSL aspect into the base model.

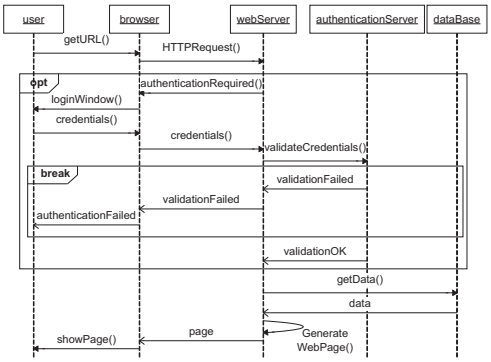

Figure 4: URL Request Sequence Diagram.

We start by illustrating the SSL aspect specification using the AOSM profile (See Figure 5). The SSL aspect provides two advices (*SSLHandshakeAdvice* and *SSLExchangeAdvice*) that implement the handshake and data exchange phases of the SSL protocol. The locations where the advices should be injected are captured by the pointcuts *SSLHandshakePointcut* and *SSLExchangePointcut*. In addition, the SSL aspect introduces two attributes (*SSLUser* and *SSLServer*) that represent the client and the server sides of the SSL implementation. The following sections present the main steps of the weaving. Currently, we are extending the IBM Rational Software Architect tool to implement the weaving steps.

5.1 Pointcuts and Aspects Instantiation

The designer instantiates the pointcuts of the SSL aspect provided by the security expert by choosing the elements of his/her model where the SSL advices should be injected. In our example, the designer selects the advice *SSLHandshakeAdvice* for the operation *Browser.authenticationRequired()* and the advice *SSLExchangeAdvice* for the operation *Webserver.credentials()* (See Figure 6).

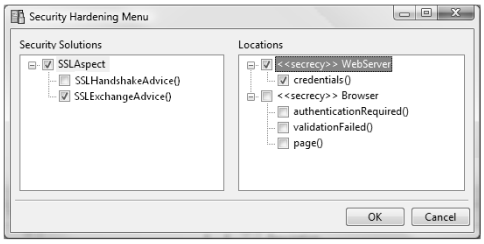

Figure 6: User Defined Join Points.

5.2 Join Points Identification

The actual join points where the SSL advices should be injected are identified and linked to the corresponding advices (See Figure 7). In our example, the join points correspond to the invocation of the operation calls *browser.authenticationRequired()* and *webServer.credentials()*. As shown in Figure 7, stereotypes ≪*SSLHandshake*≫ and ≪*SSLExchange*≫ are applied respectively to the messages *authenticationRequired()* and *credentials()* to link them to the corresponding advices.

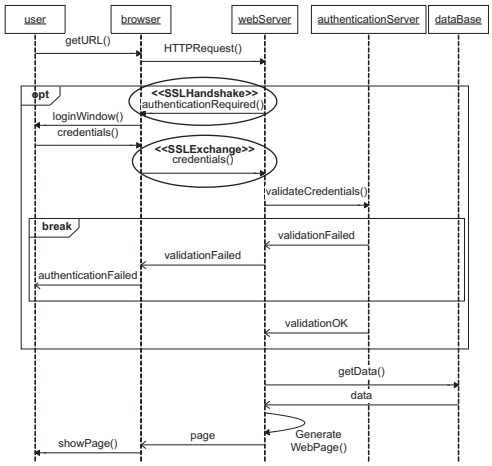

Figure 7: Identifying Join Points in the Base Model.

5.3 Advices Injection and Actual Weaving

During this step, the advices of the SSL aspect are woven into the base model of Figure 4 at the identified locations (See Figure 8). The advices of the SSL aspect are packaged into composed fragments called *InteractionUse* [7]. As the reader can notice, the complexity of the SSL advices is not shown to the designer in the woven model.

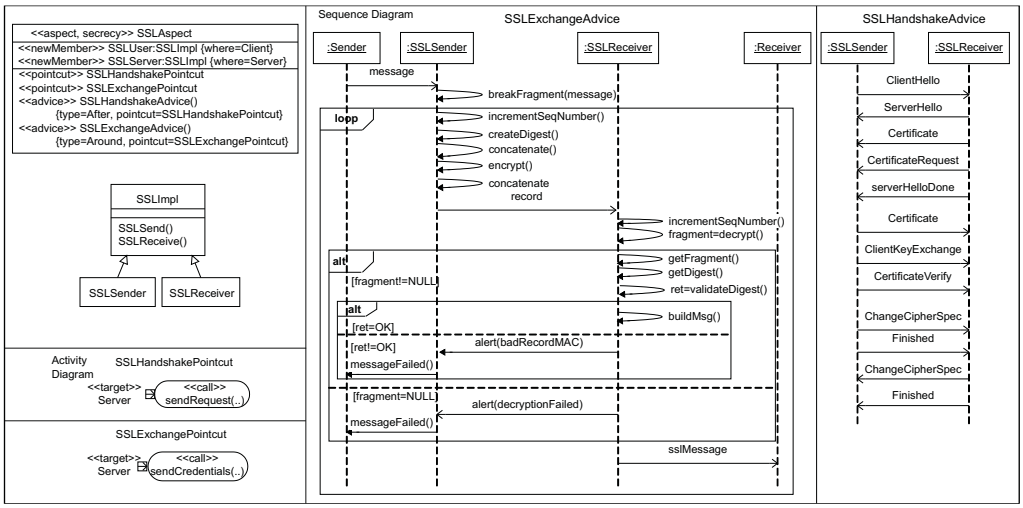

Figure 5: The Specification of the SSL Aspect using the AOSM Profile.

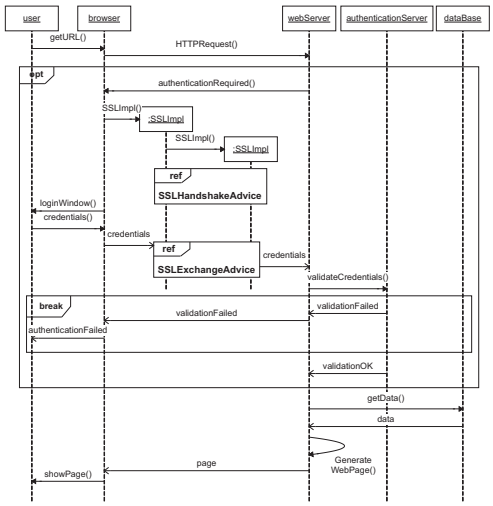

Figure 8: Woven Model.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel approach for integrating security concerns into UML design models. We showed how security solutions are specified using the AOSM profile and incorporated into design models using a design-level weaving mechanism. This approach is well suited for job separation: security experts provide high-level security solutions including the details of how to apply them in UML models and the designers apply them in their design by adapting them to the design context. With our approach, even the designers with limited security knowledge can use the security solutions to enforce the needed security requirements in a systematic way. In addition, our approach allows transparent and automatic design weaving, the designer does not need to perform any manual operation in the weaving process. As another result of our contribution, security solutions can be integrated into software from the early phases of the development life cycle. This in turn helps accelerating the development of secure applications and reducing errors. In the future, we will investigate the generation of secure code from the woven models. We will also extend the AOSM profile with more aspect-oriented features to allow the specification of more sophisticated security aspects.

7. REFERENCES

[1] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL Protocol Version 3.0. Internet Draft, 1996.

[2] L. Fuentes and P. Sánchez. Designing and Weaving Aspect-Oriented Executable UML Models. *Journal of Object Technology*, 6(7):109–136, 2007.

[3] I. Groher and M. Voelter. XWeave: Models and Aspects in Concert. In *Proc. of the AOM'07*, pages 35–40, New York, NY, USA, 2007. ACM.

[4] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2004.

[5] J. Jürjens. *Secure Systems Development with UML*. Springer Verlag, 2004.

[6] D. Mouheb, M. Debbabi, L. Wang, and M. Pourzandi. UML-Based Approaches for the Development of Secure Software and Systems: A Comparative Study. In *Proc. of PTITS'08*, Montreal, CA, 2008.

[7] Object Management Group (OMG). Unified Modeling Language: Superstructure, Version 2.1.2, 2007.

[8] J. Pavlich-Mariscal, L. Michel, and S. Demurjian. Enhancing UML to Model Custom Security Aspects. In *Proc. of AOM@AOSD'07*, 2007.

[9] I. Ray, R. France, N. Li, and G. Georg. An Aspect-Based Approach to Modeling Access Control Concerns. *Information and Software Technology*, 46(9):575–587, 2004.

[10] A. Schauerhuber, W. Schwinger, E. Kapsammer, W. Retschitzegger, M. Wimmer, and G. Kappel. A Survey on AO Modeling Approaches. Technical Report, Vienna University of Technology, 2007.

[11] J. Zhang, T. Cottenier, A. Berg, and J. Gray. Aspect Composition in the Motorola Aspect-Oriented Modeling Weaver. *Journal of Object Technology. Special Issue on AOM*, 6(7):89–108, 2007.