2009 Third IEEE International Conference on Secure Software Integration and Reliability Improvement

SSIRI 2009 Student Doctoral Program

Extending Model Driven Architecture with Software Security Assessment

Xucheng Tang Beijun Shen
School of Software
Shanghai Jiao Tong University
800 Dongchuan Road, Shanghai 200240, China
{xchtang, bjshen}@sjtu.edu.cn

Abstract

Security plays a crucial role in software systems. Existing research efforts have addressed the problem of how to model the security aspect of software at a particular phase of software lifecycle. However, security is still not integrated in all the phases of software lifecycle. In this paper we introduce how classical MDA framework can be extended to consider the security aspect. Such extension offers early assessment and early validation of security requirement, which helps to discover security flaws early in the software development process and reduce the cost of removing flaws.

Keywords: *Software Security Assessment, Model-driven Architecture*

1. Introduction

Security is one of the most important non-functional requirements in the development of modern software systems. However, it's difficult to develop a secure system. Many software systems are vulnerable and can be easily invaded. One of the main reasons is that security properties are tested after software deployment, which is too late for detecting security flaws and too costly for removing them, this result in the release of insecure software to final users.

In order to reduce security flaws early in the software development lifecycle, previous research mainly focus on two approaches. One is the application of formal methods. However, it's too difficult for developers to familiar with such methods, even after training. The other one is extending UML, a modeling language which is easy to understand and popular among developers, to model security. Several UML profiles have been defined to extend UML so as to model security properties.

Torsten Lodderstedt et al. [4] introduce secureUML to describe the information about role-based access control (RBAC) in the design phase. Codes of access control infrastructure can be generated from the design model. However, security properties other than RBAC are not considered in their profile.

Jürgen [23] provides a UML profile called UMLsec, which extends UML to allow expression of security-relevant information within the diagrams in a system specification. The profile also allows a formal evaluation for security requirements. Based on UMLsec, he study how to extract security requirements from legal regulations [1], how to analysis UML models against security requirements [2], and how to analysis java codes against security requirements [3].

Siv Hilde Houmb et al. present SecurityAssessment-UML [5], a UML profile for model-based security assessments. With that easily understandable profile, developers can identify and analysis security risk in the models.

The profiles mentioned above only focus on a particular phase of software development process, instead of integrating security aspect in all the phases of software lifecycle.

In recent years, Model Driven Architecture (MDA) [7] is an increasing concern. MDA address the complete software lifecycle and split it into 3 layer models (i.e. CIM, PIM and PSM). MDA supports model reuse, automatic model transformation and code generation, which helps to deliver applications with less efforts and higher quality.

Canonical MDA approaches just focus on functional requirements. Though there are some research efforts addressing the non-functional properties in MDA, they only target one phase in the MDA context. Few researchers discuss how to insert non-functional models into the whole MDA context, as mentioned in [8, 9]. In 2006, Vittorio Cortellessa et al. propose SPMDA, which embeds software performance analysis into the MDA context [12]. Base on SPMDA, they widen the scope of MDA by introducing a Non-functional MDA (NFMDA) framework [10]. Then the authors plug performance analysis and reliability assessment into the framework [11]. However, security and other non-functional properties are still not showed how to integrate into

978-0-7695-3758-0/09 $25.00 © 2009 IEEE
DOI 10.1109/SSIRI.2009.21

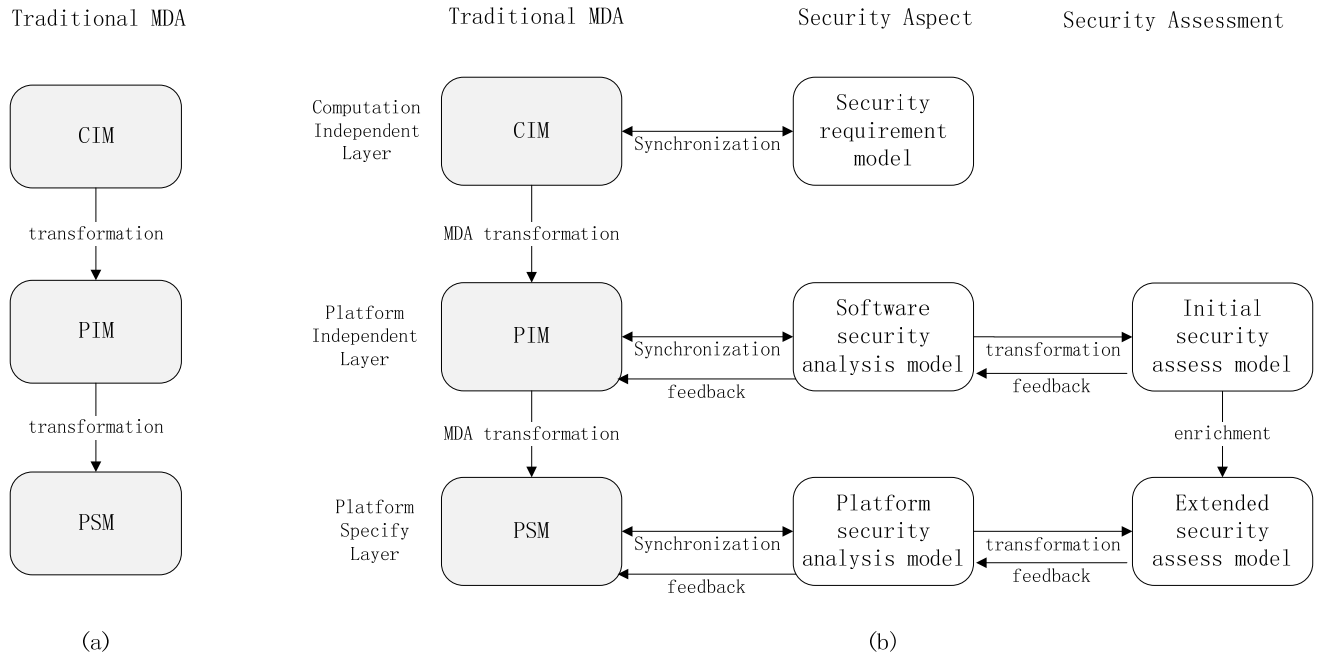

Figure 1. The traditional MDA frame work (a) and extended MDA framework (b)

NFMDA. Jordi Cabot and Nicola Zannone [8] realize this problem, they discuss that existing security proposals need to be placed within the framework for model-driven security engineering and suggest some alternative methodologies in each phase. However, since those methodologies are independent, it's still a big challenge to integrate them.

In this paper, we extend typical MDA approach, adding quantitative security assessment models, which provide feedback at every stage in the software lifecycle. This approach helps to identify security flaws as early as possible, preventing downstream propagation of security problems and amplification of flaw removal cost. We also define the relationship between the models in typical MDA and the extended models.

The remainder of this paper is organized as follows: In Section 2 we briefly introduce the traditional MDA framework, which is the foundations of our work. In Section 3 we present how traditional MDA can be extended with software security assessment. We present the related works in Section 4 and finally conclude this paper in Section 5.

2. Model Driven Architecture (MDA)

MDA is built upon three types of abstraction layers using modeling languages, as showed in Figure 1a. Computation Independent Model (CIM) describes the functional requirements using use-case diagrams, Platform Independent Model (PIM) specifies the business logic and system design which is independent to software platforms (such as J2EE and .NET) and hardware platforms. Platform Specific Model (PSM) contains the details about how the software system will be implemented and how it will be deployed. Models at a lower level of abstraction are generated from the upper level by transformation techniques and adding details.

3. Security Assessment in MDA

3.1. Framework Overview

We extend the design-oriented MDA framework with security analysis-oriented models, which can be transformed into assess models to provide a quantitative feedback.

Figure 1b shows our extended framework. The models on the left-hand side are the same with the models in traditional MDA. We employee three types of analysis-oriented models to represent the security aspect, as shown in the middle of Figure 1b. The security requirement model is used to specify security requirements. The software security analysis model aims to identify security related information in absence of platform specify information. Finally, the platform security analysis model identifies security related information of the platforms. Security related information can be used to automatically transformed into security assess models (right-hand side of Figure 1b), which will return the assessment results as a feedback. With these feedbacks, the analysis-oriented models check whether the security requirements are met. The initial security assess model provides a rough feedback because the platform specified information is unknown and just

estimated. The extended security assess model is the enrichment of the initial security assess model, adding details from the platform security analysis model, and gives a more precise feedback.

We think that it's necessarily to separate models in security aspect from the design-oriented models in traditional MDA, since directly integrating security properties into the MDA models will pollute the models and confuse the developers, who are usually non-security experts.

3.2. Risk Model

Given a threat, we use the risk model presented in [17] to assess security:

*Risk = Attack * Vulnerability * Consequence* --------- *(1)*

Where

Attack describes the Frequency of an attack,

Risk is the annual economic loss for the threat,

Vulnerability represents the probability that a specific failure mode, i, will occur, assuming that the attack has occurred;

Consequence means total measure of consequences of failure for the threat failing in mode i.

With such equation, security requirements can be defined as an upper bound of risk under a specified threat. The value of *Attack* and *Consequence* are determined in the computation independent layer, the task of the platform independent layer and platform specify layer is to get the value of *Vulnerability* and calculate *Risk* to see whether it meet the requirement.

3.3. Security Requirement Specification

In the security aspect, the first type of security model, security requirement model, is aimed to elicit risks and determine security requirements. The process can be divided into 4 steps.

1. Asset identification

The first step is to identify all critical assets in the enterprise. Assets are everything of economic value which should be protected by a secure system. Assets can be tangible such as the computers and routers, or intangible like user information and enterprise reputation. Asset identification is important that it helps to find out vulnerable assets, since asset is the target of threats.

2. Threat identification

The next step is to find out all threats. Given the use cases from CIM and the assets from the previous step, security experts find out which use case has potential security thread, and which asset will be the victim in the threat.

3. Threat analysis

In the third step, experts analyze the *likelihood* of each threat and the economic loss (*consequence*) of related asset once the threat happen.

In order to gain a better overview of security requirements already identified, we use misuse case proposed by Sindre and Opdahl[21, 22].

Misuse case is extended from typical use case to cover the cases which should not happen in the software system. In the specialized use-case diagrams, misuse cases are depicted as black ovals.

4. Security requirement

The last step specifies the annual economic loss (*risk*) of each threat using annotations.

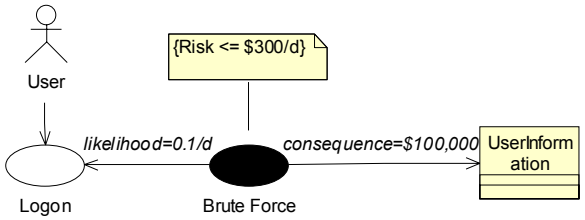

Figure 2. An instance of security requirement model

Figure 2 shows an instance of security requirement model for "logon", a simple usecase in CIM. At first, critical assets including "UserInformation" is identified. Then, the threat "Brute Force" is identified, meaning that "logon" may be misused to gain user information using brute force. In the third step, the likelihood and consequence of that threat is analyzed. At last, a security requirement is specified that the annual economic loss due to that threat should below $300 per day.

3.4. Software Security Analysis Model

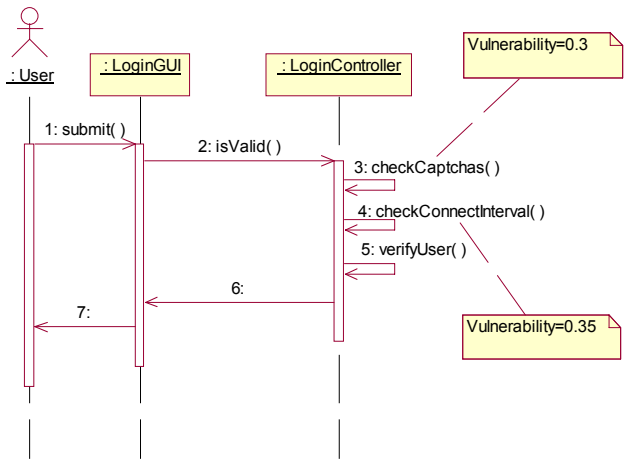

Figure 3. A software security analysis model

The software security analysis model adds security related information to the sequence diagrams in PIM

using annotations. As show in Figure 3, two treatments are used to prevent the threat of brute force. One is employing captchas. Captchas is short for "Completely Automated Public Turing test to tell Computers and Humans Apart". For example, the system may randomly produce a picture with four digitals, and the user is asked to type the digitals correctly to prove that he/she is a human. Since brute force cannot be done manually, this treatment is aimed to prevent the security threat. However, the technique of image recognition makes it possible to tell the digitals in a picture, security experts can identify the vulnerability of this treatment according to the complexity of the captchas. Another treatment records the timestamp when the user try to logon, and restricts that once a user fails to logon, he/she should wait for at least one second to try again. Sometimes the treatments can be separated from the base design model. In this situation, they can be modeled by aspect-oriented modeling so as to further separate security aspect from the design model.

With the information in the software security analysis model, an initial security assess model can be generated automatically, as shown in figure 4. Since the model is platform independent, all platform treatments are unknown, this is why the assess model is called initial. Security experts identify the upper and lower bounds of platform vulnerability according to their experiences and the history data. Finally a coarse value of vulnerability is calculated and the rough risk of that threat can be calculated according to equation (1). In Figure 4, the value of vulnerability is 0.0105~0.021.

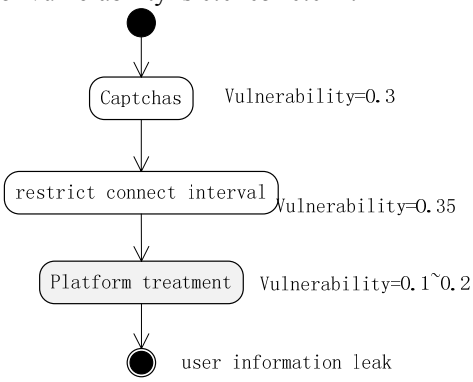

Figure 4. An initial security assess model

3.5. Platform Security Analysis Model

The platform security analysis model annotates the deployment diagrams in PSM. For the model in Figure 5, we can see that in the specified platform, a software firewall and a hardware router can prevent the threat of brute force. Security experts analyze their vulnerability and annotate them in the diagram.

The platform security analysis model can be used to enrich the initial security assess model. The node "platform treatment" in Figure 4 can be extended, as Figure 6 shows. After extension, a more accurate value of vulnerability, 0.0105, can be calculated by the extended security assess model.

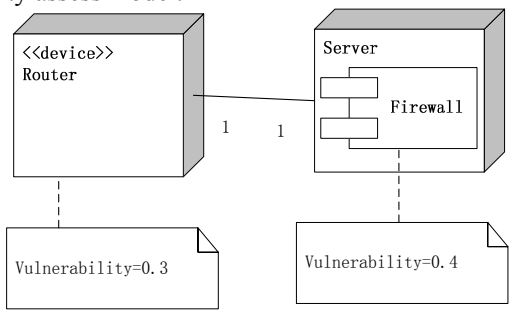

Figure 5. A platform security analysis model

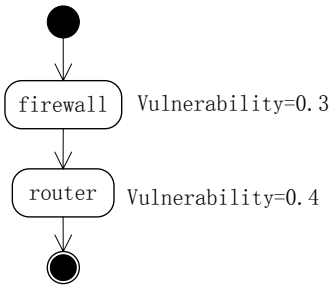

Figure 6. Part of an extended security assess model

Once the security models are integrated into MDA, we can get such benefits.

1. Security assessment can give lower and upper bounds on risks. If the lower bound on risks of a threat is larger than the specified requirement, then it's necessary to review the design or deployment and try to reduce the risks. This will save time and cost comparing with doing security testing after implementation and deployment.
2. In the design phase or implement phase, if there is more than one strategy to choose from, the result of security assessment will tell which one is more secure. This can help developers to decide which one to choose.
3. The security of software systems often rely on one or several critical components, which can be called security bottleneck. Security assess model helps to find out the security bottleneck, and the developers can concentrate their efforts on those components and try to make them more secure.
4. Security models can be reused. Once the security requirement model is finished, it can be reused when a similar software system is going to be developed. Similarly, the software security analysis model is still serviceable when the system moves to another software or hardware platform. Reusing models will save a lot because risk identification

and assessment initiated from scratch are costly and time consuming.

3.6. Synchronizations

Security requirement model is based on CIM. Each use case in CIM will be copied to the security requirement model, and then reviewed by the stakeholders and security experts, added with threats associated to relating assets and use cases, marked with likelihood and consequence. If the use case is so secure that there is no threat for it, it should also be marked as <<no risk>>. These two kinds of models are synchronized. That means, once a use case is removed from CIM, it will also be deleted in the security requirement model automatically, and if a new use case is added, or an existing use case is modified, the experts only have to annotate it. In this way, the two models are kept consistent with low cost. Similarly, the synchronization relationship also exists between PIM and software security analysis model as well as PSM and platform security analysis model.

4. Related Works

4.1. Security Assessment

Security assessment is used to validate how well the system meets specified security criteria. Researchers primarily use qualitative metrics to assess software security [6]. However, security is not an either-or property, i.e., usually a security property cannot be just described as secure or insecure. This calls for quantitative security metrics. Several quantitative security assessment approaches have been presented, e.g. [5, 6, 16]. Our work doesn't aim at introducing new security assessment metrics. Instead, we show that quantitative security assessment is useful and can be used to extend traditional MDA framework.

4.2. Non-functional Aspects in MDA

Since non-functional features has been playing an increasingly important role in software systems, it's necessary to integrate them into the MDA framework. As we mentioned in section 1, the focus of research in this area has been on performance [11-13] and reliability [11]. Although there are some literatures reporting model driven security approaches [14, 15], they target only specific security aspects at particular phases of the software development process, as said by Jordi Cabot and Nicola Zannone [8]. Our framework assess security at every stage in the MDA context, thus it can provide feedback as early as possible.

To our knowledge, NFMDA [11] is the first framework which embeds non-functional properties into the whole MDA context. Our work is inspired by NFMDA. However, NFMDA fail to separate Non-functional properties from the basic design model, which make it difficult to maintain the models.

4.3. Aspect-Oriented Modeling for Security Aspect

Premkumar T. Devanbu et al. [18] pointed out that security aspect can be modeled by aspect-oriented modeling (AOM) in order to isolate security features from the base modes and simplify software evolutions. Some researchers have proposed their approach [19, 20]. Our work focus on security assessment rather than security modeling. However, integrating AOM in our framework will bring some benefits. As mentioned in section 3.4, security treatments that are not closely coupled with the base design models can be modeled using AOM in order to separate security aspect from the basic models.

5. Conclusion and Future Work

In this paper we proposed to extend canonical MDA approach with quantitative security assessment models. Such extension provides security feedback at every stage in the software lifecycle, which helps to identify security flaws as early as possible. We also discuss about the relationship between the models in typical MDA and the extended models.

Our work is preliminary due to lack of extensive evaluation. As future work we would like to implement a tool to support our framework and perform extensive evaluation to verify the usefulness of our framework.

6. Acknowledgements

This research is supported by the National High-Tech Research Development Program of China (863 program) under Grant No.2007AA01Z139.

7. References

[1] Shareeful Islam and Jan Jürjens. Incorporating Security Requirements from Legal Regulations into UMLsec model. Modeling Security Workshop, 2008 (ModSec08)

[2] Jan Jürjens. Sound methods and effective tools for model-based security engineering with UML. 27th International Conference on Software Engineering, 2005. (ICSE05)

[3] Jan Jürjens. Security Analysis of Crypto-based Java Programs using Automated Theorem Provers. 21st IEEE/ACM International Conference on Automated Software Engineering, 2006. (ASE06)

[4] Torsten Lodderstedt, David Basin, and Jürgen Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. 5th International Conference on the Unified Modeling Language, 2002. (UML02)

[5] Siv Hilde Houmb and Kine Kvernstad Hansen. Towards a UML Profile for Security Assessment. 2nd International Workshop on Critical Systems Development with UML, 2003 (CSDUML03)

[6] Joseph Pamula and Paul Ammann. A Weakest–Adversary Security Metric for Network Configuration Security Analysis. Proceedings of the 2nd ACM workshop on Quality of protection

[7] J. Miller (editor), Model-Driven Architecture Guide, omg/2003-06-01 (2003).

[8] Jordi Cabot and Nicola Zannone. Towards an Integrated Framework for Model-driven Security Engineering. Modeling Security Workshop, 2008 (ModSec08)

[10] Vittorio Cortellessa, Antinisca Di Marco, and Paola Inverardi. Non-functional Modeling and Validation in Model-Driven Architecture. 6th Working IEEE/IFIP Conference on Software Architecture, 2007 (WICSA07)

[11] Vittorio Cortellessa, Antinisca Di Marco, and Paola Inverardi. Integrating Performance and Reliability Analysis in a Non-Functional MDA Framework. Fundamental Approaches to Software Engineering, 10th International Conference, 2007 (FASE07)

[12] Vittorio Cortellessa, Antinisca Di Marco, and Paola Inverardi. Software Performance Model-Driven Architecture. 21th Annual ACM Symposium on Applied Computing, 2006 (SAC06)

[13] Mathias Fritzsche and Jendrik Johannes. Putting Performance Engineering into Model-Driven Engineering: Model-Driven Performance Engineering. ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems

[14] D. Basin, J. Doser, and T. Lodderstedt. Model Driven Security: from UML Models to Access Control Infrastructures. TOSEM, 15(1):39–91, 2006.

[15] M. Clavel, V. da Silva, C. Braga, and M. Egea. Model-Driven Security in Practice: An Industrial Experience. In Proc. of ECMDA-FA'08, LNCS 5095, pages 326–337. Springer-Verlag, 2008.

[16] Norman F. Schneidewind. Cyber Security Prediction Models. The R & M Engineering Journal, American Society for Quality, December 2005

[17] James W. Jones. ASME risk analysis and management for critical assets protection (RAMCAP) methodology document. Presentation to PS&S Interagency Working Group, September 2004.

[18] Premkumar T. Devanbu and Stuart Stubblebine. Software Engineering for Security: a Roadmap. Proceedings of the Conference on the Future of Software Engineering, 2000

[19] Jaime Pavlich-Mariscal, Laurent Michel, and Steven Demurjian. Enhancing UML to Model Custom Security Aspects. Proceedings of the 11st workshop on Aspect-oriented modeling, 2007

[20] Djedjiga Mouheb, Chamseddine Talhi, Vitor Lima, Mourad Debbabi, Lingyu Wang and Makan Pourzandi. Weaving Security Aspects into UML 2.0 Design Models. Proceedings of the 13th workshop on Aspect-oriented modeling, 2009

[21] Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements by misuse cases. Proceedings of TOOLS Pacific 2000, pages 120-131. IEEE Computer Society Press Sydney, Australia

[22] Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements with misuse cases. Requirements Engineering. 10(1):34-44, 2005

[23] Jan Jürjens. UMLsec: Extending UML for secure systems development. 5th International Conference on the Unified Modeling Language, 2002. (UML02)