Model-Driven Security with *A System of Aspect-Oriented Security Design Patterns* [*]

Phu H. Nguyen, Jacques Klein, and Yves Le Traon
Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg
{phuhong.nguyen, jacques.klein, yves.letraon}@uni.lu

ABSTRACT

Model-Driven Security (MDS) has emerged for more than a decade to propose sound MD methodologies for supporting secure systems development. Yet, there is still a big gap before making MDS more easily applicable, and adoptable by industry. Most current MDS approaches have not extensively dealt with multiple security concerns but rather a specific one, e.g. authorization. Besides, security patterns which are based on domain-independent, time-proven security knowledge and expertise, can be considered as reusable security bricks upon which sound and secure systems can be built. But security patterns are not applied as much as they could be because developers have problems in selecting them and applying them in the right places, especially at the design phase. In this position paper, we propose an exploratory MDS approach based on a *System of aspect-oriented Security design Patterns* (shortly called SoSPa) in which security design patterns are collected, specified as reusable aspect models to form a coherent system of them that guides developers in systematically selecting the right security design patterns for the job. Our MDS approach allows the selected security design patterns to be automatically composed with the target system model. The woven secure system model can then be used for (partial) code generation, including (configured) security infrastructures.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]: Patterns—*security design patterns*; D.2.13 [**Reusable Software**]: Reuse models—*reusable aspect models*

General Terms

Design, Security

[*]Supported by the Fonds National de la Recherche (FNR), Luxembourg, under the MITER project C10/IS/783852.

Keywords

Model-driven security, aspect-oriented modeling, model composition, security patterns, authentication, authorization

1. INTRODUCTION

Information security is getting more and more important nowadays. The more benefits brought to us from advances in computers and information technology, the more crucial and challenging information security is becoming. However, the traditional methods for supporting the development of secure systems are still inefficient to ensure security. Not many days pass without new headlines on the newspapers about IT security vulnerabilities, security attacks, digital data leaks, and so on. Security requirements are becoming more complex, always changing, often scattered and tangled with functional requirements and other non-functional quality criterion, but are not seriously taken into account at the early stages of the development process. Moreover, economic pressure always shortens the development time, but increases the frequency of required modifications. Eventually, many security defects had been exploited in practice when it was too late and costly.

In this context, *Model-Driven Security* (MDS) has emerged more than a decade ago as a specialized *Model-Driven Engineering* (MDE) approach that aims at providing more efficient support for developing secure systems. MDS provides means for improving the *productivity* of the development process and *quality* of the resulting secure systems, with *models* as the main artifact. There have been many MDS approaches introduced over the last decade. Yet, there is still a big gap before making MDS more easily applicable, and adoptable by industry. In our recent systematic review of MDS [7], the results have shown that there is a lack of approaches dealing with multiple security concerns at the same time. Most current MDS approaches have not extensively dealt with multiple security concerns but rather a specific one, e.g. authorization (especially, access control). Moreover, very few MDS approaches fully leverage the advantages of *Aspect-Oriented Modeling* (AOM) in specifying security concerns for secure system development. AOM, or more generally, *Aspect-Oriented Software Development* (AOSD) might be a good solution for tackling complexity in dependable systems by following the well-known principle of divide and conquer. More specifically, AOSD aims at addressing crosscutting concerns such as security, synchronization, concurrency, persistence, performance, etc., by providing means for their systematic identification, separation, representation and composition.

On the other hand, from security engineering's point of view, one of the best practices is using of patterns to guide security at each stage of the development process [9]. Patterns are applied in the different architectural levels of the system to realize security mechanisms. However, there is not any MDS approach yet proposing a system of aspect-oriented security design patterns where the interrelationships among the security design patterns, and with other non-functional quality aspects (e.g., performance, usability, etc.) are taken into account. Security design patterns could be considered as reusable security design bricks upon which sound and secure systems can be built. It is important to note that security patterns are based on domain-independent, time-proven security knowledge and expertise. But they are not applied as much as they could be because developers have problems in selecting them and applying them in the right places, especially at the design phase [10]. Indeed, security patterns could be applied at different levels of abstraction, e.g. architectural design rather than detailed design. Moreover, the levels of quality found in security patterns are varied, not equally well-defined like software design patterns [4]. Particularly, many security patterns are too general, without a well-defined, solution-oriented description. There is also a lack of coherent specification of interrelationships among security patterns, and with other non-functional quality criterion like performance, usability, etc.

A sound methodology for supporting secure system development must guide security throughout the development cycle, especially during the design phase. To the best of our knowledge, none of existing MDS approaches has proposed a *System of aspect-oriented Security design Patterns* which provides not only well-defined security design patterns but also the support meta-information that guide developers in systematically selecting the right security design patterns for the job. In this position paper, we propose an exploratory MDS approach based on a *System of aspect-oriented Security design Patterns* (SoSPa) in which security design patterns are collected, specified as reusable aspect models (RAM) [6] to form a coherent system of them. As we target a full MDS approach, the SoSPa is proposed to be built on a meta-model which is part a *Model-Driven Software Development* framework. Our framework allows the selected security design patterns to be automatically composed with the target system model. The woven secure system model can then be used for (partial) code generation, including (configured) security infrastructures.

2. BACKGROUND

First, we give a definition of *security pattern*, and our definition of *a system of security design patterns*. Next, two important security concerns are presented, i.e. authentication and authorization. Then, RAM is briefly introduced.

2.1 System of Security Patterns

According to Schumacher et al. [9], a *security pattern* describes a particular recurring security problem that arises in specific contexts and presents a well-proven generic scheme for its solution. Security patterns typically do not exist in isolation because applying one solely can not make a system secure to different threats. Related patterns should be grouped and work together to address more complex security problems in the real world. *A system of security design patterns* is a collection of patterns for designing secure systems, together with guidelines for their implementation, combination, and practical use in secure systems development.

2.2 Identification & Authentication

Identification & Authentication (I & A) addresses the security problem in recognizing an actor that is interacting with a system. I & A techniques are described in [9], e.g. User ID/Password, Biometrics, Hardware token, etc.

2.3 Authorization

Authorization addresses the security problem in deciding who is authorized to access specific resources in a system. There are different mechanisms for implementing authorization, e.g. Role-Based Access Control (RBAC), Organization-Based Access Control (OrBAC), Discretionary Access Control (DAC), etc. Other well-known patterns in this category are described in [9], e.g. Multilevel Security, Reference Monitor, Role Rights Definition, etc.

2.4 Reusable Aspect Models

RAM [6] is an aspect-oriented multi-view modeling approach with tool support for aspect-oriented design of complex systems. In RAM, any concern or functionality that is *reusable* can be modeled in an aspect model.

3. OUR MODEL-DRIVEN APPROACH

Our approach is based on a *System of aspect-oriented Security design Patterns* (SoSPa) which provides not only well-defined security design patterns but also the support meta-information that can guide developers in systematically selecting the right security design patterns for the job. Fig. 1 shows our meta-model of SoSPa (SoSPa-MM) that is an extension of the RAM meta-model [6]. The core elements of SoSPa-MM are depicted in white. The rest are core elements of RAM meta-model. Security design patterns are collected, specified as reusable aspect models (RAM) to form a coherent system of them. RAM is very suitable for specifying security design patterns because of its multi-view modeling ability. To be more specific, the system of security design patterns consists of an extensible set of security concerns (e.g. authentication, authorization, encryption, etc.) which can fulfill the security objectives/properties (e.g., confidentiality, integrity, availability, privacy). Each security concern is composed of a set of aspect-oriented security design patterns that realize the security concern. To support the selection of security design patterns, we use feature model (FM) as in software product line (SPL) to capture the variability of security patterns. The FM is to be extended to be able to specify the interrelationships of security design patterns. The meta-info about interrelationships of security design patterns are well specified within the system of them. For example, some security design patterns could complement one another, or exclude each other. The relations can be transitive and/or symmetrical. We capture the core dependency types among security patterns, i.e. *depends on*, *benefits from*, *alternative to*, *impairs*, *conflicts with*, as discussed in [10]. Pattern A *depends on* pattern B means that A will not function correctly without B. The depends-relationship is not symmetrical, but transitive. Pattern A *benefits from* B means that implementing B will add to the value already provided by implementing A. Pattern A *conflicts with* pattern B means that implementing B in a system that contains A will result in inconsistencies. Moreover,

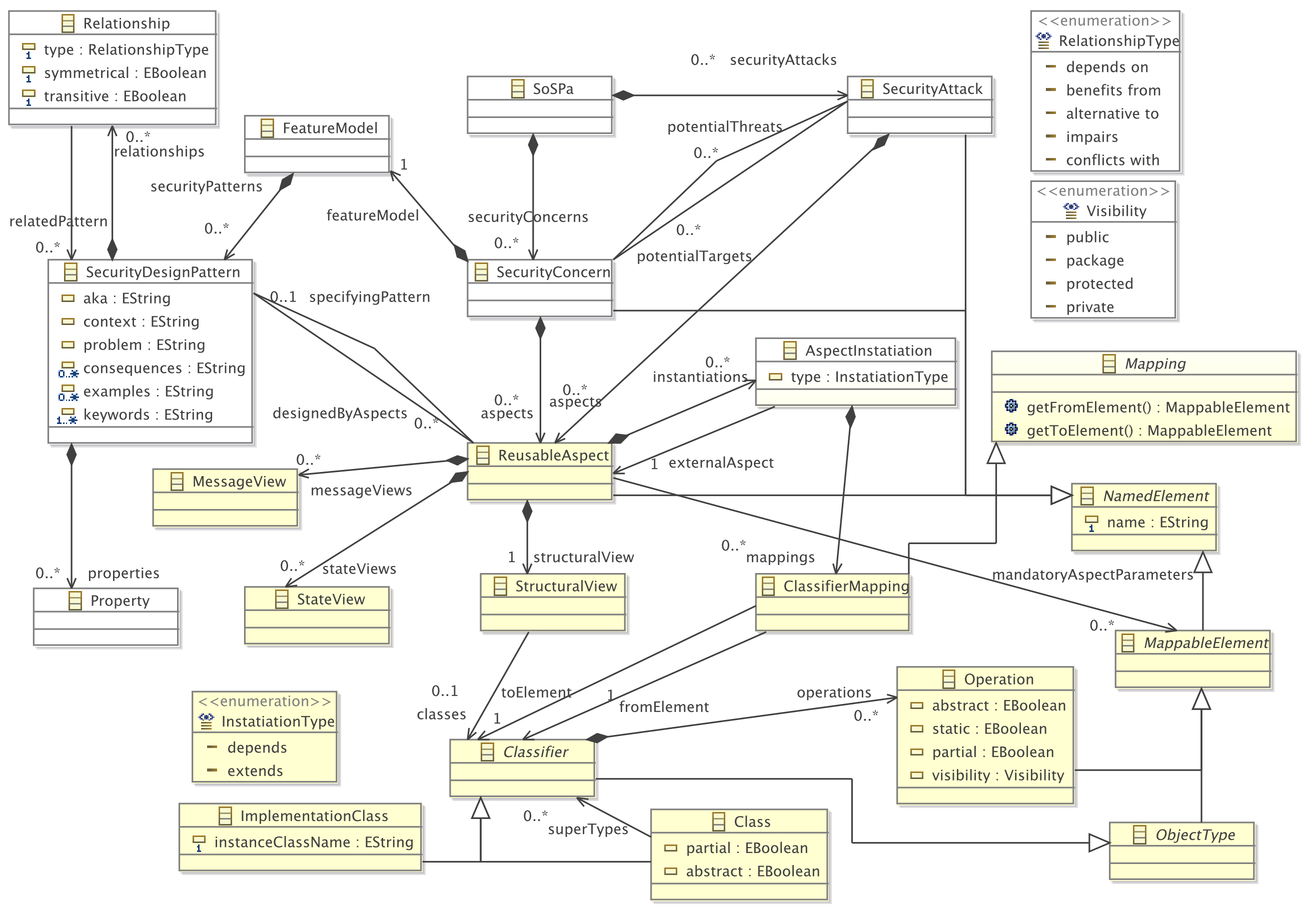

Figure 1: Meta-model of A *System of aspect-oriented Security design Patterns* (SoSPa-MM)

each security design pattern also contains meta-info describing the side effects of its adoption on other non-functional quality criterion, e.g. performance, usability, etc. For each non-functional quality criteria, an impact model of the security design pattern for the criteria can be built as discussed in [1]. All these meta-info are useful for analysis of the trade-off among alternatives which leads to a thoughtful decision on systematically selecting the right security design patterns for the job. On the other hand, attack models can also be specified using SoSPa-MM. These attack models are associated to the security concerns, and can be woven into the system model to generate misuse models for formal analysis of security, or generate test cases for testing.

The process of selecting and composing security design patterns into the target system design consists of the following main steps:
1. Identify the security-critical assets of the target system with the priorities to have them.
2. For each asset, determine security concerns for the corresponding asset type. For each security concern, describe the context and the security problem that need to be solved.
3. Identify the possible attacks/threats for the security-critical assets by consulting well-known sources, e.g. the OWASP top 10 most critical web application security risks[1]. Risk analyses can be done using many different risk analyses methodologies, e.g. the CORAS framework as discussed in [3]. The results of this step are the attack models which can then be specified by our meta-model.
4. For each security concern, use the feature model of the security concern (e.g. a simple FM of authorization is shown in Fig. 2) to select the most appropriate security design pattern, i.e., the pattern that best matches with the context and the security problem, most satisfies the interrelationships with the other already selected security design patterns, and maximizes the positive impact on relevant non-functional quality criterion like usability, performance, etc. This step derives the detailed design for the selected security pattern, including its *customization interface* and *usage interface*. The *customization interface* of a RAM model consists of so-called *mandatory instantiation parameters* that must be instantiated in order for the model to be used within a specific application. The *usage interface* of a RAM model is comprised of all the *public* model elements, e.g. *public* class properties like attributes and operations. More details about these two kinds of interface of a RAM model can be found in [1].
5. For each selected security pattern, use the customization interface of the generated design to adapt the generic design elements to the application-specific context. This step generates the mappings of the parameterized elements in the security design pattern with the target elements in the target system design. Any constraints between mappings of all the selected security design patterns need to be resolved.

[1]`www.owasp.org`

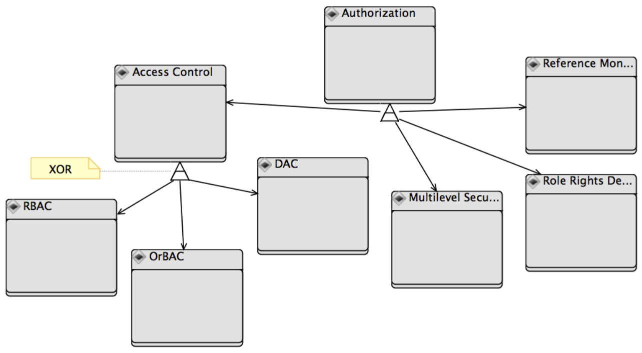

Figure 2: A Feature Model of *Authorization*

6. Automatically weave all the selected security design patterns into the target system design. The mappings from previous step are the input for the weaving process in this step.
7. Analyze the woven secure system against the attack models obtained from step 3. Depending on each security pattern, the attack models can be used for formal verification of security properties in the woven model, or can be used for test cases generation in a security testing approach.

The first two steps are assumed to be done from the requirement engineering process. To be more specific about step 4, in this paper we only consider the first two criterion for selecting the most appropriate security design pattern, i.e. the matching with the context and security problem, and the interrelationships with the other selected security design patterns. The third criteria, i.e., to maximize the positive impact on relevant non-functional quality criterion like usability, performance, etc. is out of scope of this paper and is left for the future work.

4. RELATED WORK

SecureUML [2] and UMLsec [5] are two different MDS approaches based on UML profiles that can be used for specifying security requirements and specifications in a system design. SecureUML is specifically meant for access control (authorization). UMLsec is not really proposing the use of security patterns even though there is an approach about specifying cryptography pattern using UMLsec. Neither SecureUML nor UMLsec proposes a system of security patterns or leveraging aspect-oriented modeling in its approach.

There are other aspect-oriented security design methodologies for modeling security aspects using patterns, and then weaving them into functional models [3], [8]. In [3], the authors mainly aim at security property validation in abstract specifications of system models. The approach proposed in [8] is only for modeling access control concerns. Also, these approaches do not consider a system of security patterns where their interrelationships and constraints are taken into account.

Our MDS approach proposed in this position paper is inspired by [1]. In that paper [1], the authors propose an approach based on RAM for designing software with concern as the main unit of reuse. They show how their ideas can be realized by using an example of low-level design concern, i.e. the *Association* concern. The application of their approach to high-level concerns like security is kept open but not dealt with yet. In this paper, we specifically target security with a *System of aspect-oriented Security design Patterns* which can be specified by extending RAM. We extend the concept of using variation interface in [1] for specifying the interrelationships among security patterns. We also plan to extend the concept of using impact model for specifying the constraints of security patterns with other non-functional quality criterion like performance, usability, etc.

5. CONCLUSION AND FUTURE WORK

In this position paper, we have proposed a MDS approach based on a *System of aspect-oriented Security design Patterns* (SoSPa). In our approach, security design patterns are collected, specified as reusable aspect models to form a coherent system of them that guides developers in systematically selecting the right security design patterns for the job. To be more specific, a *System of aspect-oriented Security design Patterns* can be specified using our meta-model called SoSPa. This meta-model is part a full *Model-Driven Software Development* framework. Our framework allows the selected security design patterns to be automatically composed with the target system model. The woven secure system model can then be used for (partial) code generation, including (configured) security infrastructures.

We are working on validating our ideas with different case studies, targeting three main security concerns, i.e. authentication, authorization, cryptography, plus interrelationships among them. Future work can be also dedicated for extending the approach in [1] for specifying the constraints of security patterns with other non-functional quality criterion like performance, usability, etc. Moreover, the steps of risk analyses, and formal verification and validation of security properties still remain for future investigation.

6. REFERENCES

[1] O. Alam, J. Kienzle, and G. Mussbacher. Concern-oriented software design. In *MODELS*. 2013.

[2] D. Basin and J. Doser. Model driven security: From UML models to access control infrastructures. *ACM Transactions on Software*, (1945):353–398, 2006.

[3] G. Georg, I. Ray, K. Anastasakis, B. Bordbar, M. Toahchoodee, and S. H. Houmb. An aspect-oriented methodology for designing secure applications. *Information and Software Technology (IST)*, 2009.

[4] T. Heyman, K. Yskout, R. Scandariato, and W. Joosen. An analysis of the security patterns landscape. In *Proceedings of SESS '07*, SESS '07, 2007.

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

[6] J. Kienzle, W. Al Abed, F. Fleurey, J.-M. Jézéquel, and J. Klein. Aspect-oriented design with reusable aspect models. In *TAOSD VII*. 2010.

[7] P. H. Nguyen, J. Klein, M. Kramer, and Y. Le Traon. A Systematic Review of Model Driven Security. In *Proceedings of the 20th APSEC*, 2013.

[8] I. Ray, R. France, N. Li, and G. Georg. An aspect-based approach to modeling access control concerns. *IST*, 46(9):575 – 587, 2004.

[9] M. Schumacher, E. Fernandez, D. Hybertson, and F. Buschmann. *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, 2005.

[10] K. Yskout, T. Heyman, R. Scandariato, and W. Joosen. A system of security patterns, 2006.