Towards a Reuse-oriented Security Engineering for Web-based Applications and Services

Aleksander Dikanski, Sebastian Abeck
Research Group Cooperation & Management (C&M)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{ a.dikanski, abeck }@kit.edu

***Abstract*—Security should be considered throughout a software development process to develop secure applications. This security engineering effort is restricted due to the complexity and diffusion of todays security knowledge. Approaches, such as misuse cases for threat specification and patterns for security functionality modeling, try to use and integrate security into software development, but their combined use is still difficult. In this paper a framework for developing secure software systems is presented, which aims at incorporating and unifying existing security engineering approaches by applying well-established reuse-oriented software development paradigms, such as service-orientation. The security-related activities and reusable artifacts of important development phases are discussed and the mapping of artifacts between different development phases is presented.**

Keywords-security engineering; software development; security patterns; service-orientation

I. INTRODUCTION

The increasing number of attacks on software systems makes it more important than ever to develop secure software systems. Especially web-based applications and services are faced with numerous threats due to their public access. But, the prevailing custom of including security functionality after the functional development is infeasible, is not fulfilling the actual security needs. Security engineering aims for a consecutive secure software develop-ment by introducing methods, tools, and activities into a software development process [1].

Such an integration has not yet been achieved completely as the amount of security knowledge, including theoretic models, technologies and standards, developed until now is complex, often diffused, and seldom structured enough to be used in a software development process. Opposed to this, security can usually be considered reusable across hetero-geneous functional domains, e.g., access control models such as role-based access control (RBAC, [2]) can be used in different domains. Yet, so far structured means for reuse of security functionality are not successfully employed. Exist-ing approaches contribute mainly to specific develo-pment phases. Yet, while each of these approaches is beneficial in its intentions, they are hard to integrate.

In this work, an early version of a framework is presented, which aims at structuring existing security know-ledge in a reusable fashion and providing decision support to integrate existing security engineering approaches and methodologies more concisely. The concepts of modern reuse-oriented paradigms, such as service-orientation, soft-ware product lines (SPL) as well as model-driven software development (MDSD), are facilitated in our approach.

The goal is to present developers a structured tool set, to ease the coherent integration of security aspects into each phase and across phases. Thus an increased quality of the security functionality is achieved.The framework comprises security requirements analysis templates and security pattern languages. The former can be instantiated to analyze the security needs of an application in a deterministic way, while the latter can be used to choose appropriate security solutions and iteratively refine them.

In the next section, approaches relevant for our work will be discussed. In Section 3, the contribution of our approach will be described. We further present two projects which lead to the development of our framework in Section 4. A conclusion closes the body of this paper.

II. RELATED WORK

Reuse in security engineering processes is discussed in several approaches. The SECTET-framework [3] provides a service-oriented security engineering approach for authori-zation in inter-organizational workflows, but concentrates mostly on web-service based architectures. The Secure-Change-Project aims at a change-driven security engineering approach, in which security requirements are specified, which evolve throughout the lifetime of software [4]. While the focus of this project is change of security requirements and security design, our focus lies on presenting feasible choices and decision support for them to increase quality of security functionality.

Threat and risk analysis techniques for analyzing and specifying security requirements include STRIDE [5], attack trees [6], and misuse cases [7]. We are aiming at providing deterministic threat descriptions at an appropriate abstraction level and link them to appropriate security requirement specifications to complement theses techniques, as this is were each of them fails short and is thus difficult to apply.

Security patterns are a popular and widely accepted method for modeling technology-independent security functions [8]. Security pattern languages are utilized to describe the connections between multiple patterns and their combined usage [9]. But, alternative solutions are not considered by existing languages. So far, only SPL approaches consider such variations [10]. We aim to enhance

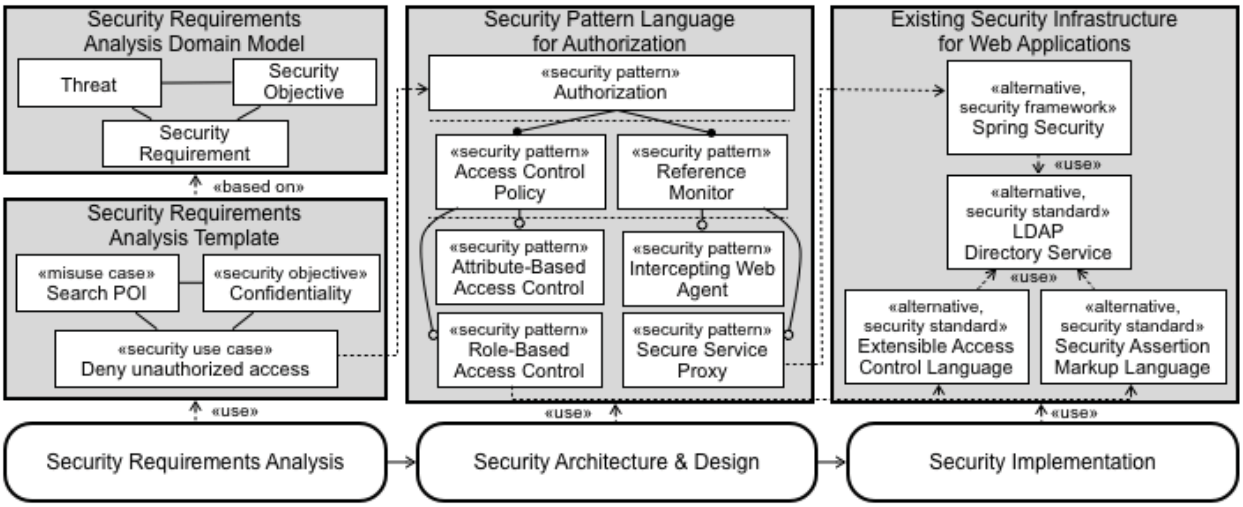

Figure 1. Overview over the security engineering framework

security pattern approaches by explicitly showing alternative pattern solutions to security problems.

Model-driven security applies to methods of model-driven software development to the security domain. Secure-UML [11], UMLSec [12], and the work of Emig et al. [13] are among the most prominent approaches in this field. They do not consider existing security infrastructures in a service-oriented way as we intend to do. Also, they also do not provide chocies between alternative security patterns.

III. Security Engineering Framework

The framework presented in the following complement and unifyies existing approaches in the security engineering field by providing reusable security-related development artifacts and a decision support for them.

Reuse is at the core of many well-established software engineering paradigms, which aim at managing complex software systems development, such as service-orientation, SPL and MDSD. A security engineering methodology based on the reuse of existing security knowledge will lead to an increased efficiency in the development of secure software and to an improved quality of the security functionality.

An important goal for our approach is to be development process agnostic, i.e., the artifacts contributed by our framework should be independent from specific software development processes and instead be applicable in different methodologies and paradigms.

We further aim for decision support and guidance in using security knowledge. The security domain comprises a large knowledge base, including, e.g., security standards and technologies as well as security models, principles and policies. A structured approach is needed for applying this knowledge in a development process. The focus lies on supporting a decision process by pointing out alternative solution to security problems.

Currently, the framework is limited to security within web application and service development, thereby neglecting lower levels of security measures such as web server, operating system, and network, even though this is considered bad security practice. Yet, we do not rule out the applicability of our approach to these levels.

The next sections will present the core elements of our framework and their intented function. The focus, thereby, lies upon the first three phases, i.e., requirements analysis, design and implementation phase. Testing and operation are important phases in the development of secure applications as well, but we exclude them here for brevity reasons.

A. Reusable Security Requirements Templates

Similar to functional requirements elicitation, security requirements need to be analyzed and specifed as well to determine application security needs. Difficulties in this phase concern the appropriate abstraction level and the format of the security requirements specification. Often they are specified by proposing security functionality, instead of constraints to the functionality [14][15].

According to our goals, our approach strives for contributing reusable *security requirements analysis templates (SecRAT)* to this phase. The template's core is based on the relationships between threats, which violate security objectives, security requirements, which are capable of mitigating the threats and implement security objectives. These entities and their relationships form a basic *security requirements analysis domain model (SecADM)*, giving structure to the templates. The templates will further be categorized into *domain-independent SecRATs*, applicable to multiple functional domains, and *domain-specific SecRATs*, describing security requirements and threats specific to a functional domain. This allows for a more focused and structured approach to requirements analysis.

Reuse of security knowledge is thereby achieved by documenting existing threat knowledge and explicitly linking it to appropriate security requirements and objectives. Therefore, if a threat is determined to be applicable in an application development process, the appropriate template can be instantiated, directly leading to related security requirements as well as objectives and vice versa.

The templates are thereby independent of any approach for analyzing and specifying security requirements such as

those mentioned in Section 2. Instead they can be used as a structured decision support tool to determine necessary security requirements as well as a common specification format for any such process and modeling tools. Further, each SecRAT is linked to an abstract security functions, thereby supporting the transfer between requirements engineering and design phase.

B. Security Pattern Language and Variability Model

The goal of the design phase is to implement security requirements using appropriate security functions. They are firstly specified at a coarse-grained, technology-independent architecture-level design and iteratively refined to an fine-grained, implementation-level design. These functions form the security architecture of one or more applications and thus need to be integrated into the overall architecture [16].

For the iterative refinement process, a concise *security pattern language (SecPAL)* for each security function design is proposed, which builds upon and complements previous approaches. It enables the use and combination of multiple security patterns, each of which relates to and implements a certain security requirement. Thus a decision support is offered, in that only compatible patterns are connected in the pattern language. Yet, opposed to previous approaches, the focus of the SecPALs lies on iterative refinement.

At each iterative refinement the design is not always obvious. In fact, a choice between several design options can be made. For example, to implement access control, several alternatives exist, including role-based (RBAC, [2]) and attribute-based access control [17], each of which might be more suitable depending on application context.

To provide an overview over viable alternative security patterns applicable to specific security problems, the SecPAL is complemented by a *security pattern variability model (SecPVM)*. In each iterative refinement of a pattern, the variability model can be applied to select an appropriate variant for a pattern, if necessary. Currently, feature models [18], a common tool to model commonality and variability in SPL development, are feasible candidates to describe security pattern variants including mandatory, optional and exclusion relationships.

C. Service-Oriented Security Design

The combination of SecRAT, SecPAL, and SecPVM is intended to support the development of new or the extension of existing security functionality for software systems. But they can also be used to support secure development projects, which need to be integrated into an existing security infrastructure, e.g., in an enterprise environment. In this context, we build upon our previous efforts [16][19] by applying the service-orientation paradigm, i.e., the reuse and restructuring of existing software systems to satisfy business needs, to the design phase of security engineering as well.

Reusing existing services narrows security design decisions. When developing an application for an IT infrastructure in which, e.g., RBAC is the standard access control policy model, a decision about the policy model to use for access control in the newly developed application is already determined.

In order to achieve the benefits of using security services in the design models, an abstraction of the implemented services to a technology-independent level is required, displaying appropriate views of the complete security architecture to developers [20].

We currently employ a manual approach, in which required abstractions are provided by security experts once for each utilized product, as we have done in previous work [19]. The SecPAL can in this case be used as guidance to identify fine-grained patterns within existing security frameworks and products. SecPVM can be used to identify and document alternative implementations offered by the security framework or product. By following the language paths in reverse direction, a relationship to more abstract, coarse-grained patterns can be established.

D. Standards- and Pattern-based Model-Driven Security

Despite the reuse of existing functionality it is inevitable that certain artifacts need to be developed as part of the security engineering approach, even though our goal is to reduce the number of such artifacts to allow for an efficient development. In this context, we continue our previous efforts on model-driven security [13], but are more focussed on integrating it into a security engineering approach using security technology standards and patterns to automatically generate necessary artifacts.

While implementing security functionality, employing security technology standards offers product independence and interoperability. Yet, applying standards without in depth knowledge is difficult, as they include a large degree of flexibility. A very good example for this is the slowly progressing adoption of XML-based security standards, developed mainly for web service-based applications [21][22]. Note that the same can be argued for security frameworks and products.

As a benefit of the security pattern identification and specification using the SecPAL described in the previous section, specific guidelines and templates on how standards are to be utilized to implement a certain security pattern. As such, we are able to provide a security platform description, which is used as a automatically generate relevant artifacts from design models specified using SecPAL.

IV. Motivating Case Study Scenario

We are currently applying, refining, and evaluating our approach by applying it in the development of two real-world projects, requiring security functionality.

A. Case Study Description

The KITCampusGuide (KCG) is a web-based and service-oriented geographic information system (GIS). It supports employees, students and guest of the Karlsruhe Institute of Technology (KIT) with their daily campus activities. Its basic functionality allows the user to search for points of interest (POI), such as buildings, rooms or offices, and display results on a campus map.

This functionality will be extended as a proof-of-concept for the european project OpenIoT by enabling students to search for available workplaces on the campus. This functionality will be implemented using smart objects. These virtual or physical objects, such as rooms, are active participants in the information systems and can be remotely queried and their state modified using sensor and actor technology.

Very early it became clear, that security aspects needed to be implemented in the KCG application as the KIT is

restricted by legislative regulations, so that the privacy and anonymity of the users, as well as integrity and confidentiality of the processed data need to be assured, due to the location sensitive nature of the application.

As the application was targeted to be integrated into the overall KIT IT-infrastructure, utilization of the existing security infrastructure was required. Therefore, the capabilities of the provided security services need to be analyzed, so that the analyzed security requirements can later be mapped upon them.

B. *Preliminary Results*

We are currently in the progress of formulating an initial set of reusable SecRATs for web-based GIS and IoT applications based on our experiences in developing the KCG application as well as best practice security requirements found in literature. These SecRATs are extensions of a domain-independent SecRAT, which is currently developed as well. From these, a first draft of the SecADM will be developed. For the documentation we are currently using textual templates, but are evaluating more formal methods.

We are further using existing works on security patterns to formulate appropriate SecPALs for different security solutions such as authentication and authorization. Not many such patterns are available in the field of IoT-applications, which is why we will document new security solutions as well. In doing so, we are evaluating and formalizing alternative design decisions into an appropriate SecPVM using feature models.

We plan to validate our approach by applying it to other projects from the same domain to test the integrity of the developed artifacts. We further plan to adopt our approach to different domains including cloud based applications. To evaluate the increased quality of the implemented security functionality empirical studies will be performed.

V. Conclusion and Outlook

In this paper, a security engineering framework was outlined, which focuses on the structured reuse of existing security knowledge by providing analysis templates and pattern languages for security to increase quality of the implemented security functionality. We argued the benefits of security engineering and pointed out short-comings of existing approaches due to missing integration and combination. We identified that a structured approach need to be based on reuse of existing security knowledge and a decision support system in order to be feasible in the field. We presented an outline for several contributions to the different phases of a development process based on and complementing existing approaches, which we think will be beneficial to more efficient security engineering. In future works, we will flesh out the details of the contributions. We further presented current projects, which are used as cases studies to demonstrate the practicability of our approach.

References

[1] R. J. Anderson, Security Engineering. 2nd ed., Indianapolis, Ind.: Wiley, 2008, p. 1040.

[2] D. Ferraiolo, R. Sandhu, S. Gavrila, and D. Kuhn, “Proposed NIST Standard for Role-Based Access Control,” ACM Transactions on Information and System Security (TISSEC), vol. 4, no. 3, pp. 224–274, 2001.

[3] M. Hafner and R. Breu, Security Engineering for Service-Oriented Architectures. Heidelberg: Springer, 2008.

[4] R. Scandariato and F. Massacci, “SecureChange: security engineering for lifelong evolvable systems,” in ISoLA'10: Proceedings of the 4th international conference on Leveraging applications of formal methods, verification, and validation, 2010, vol. Part II , Volume Part II, pp. 9–12.

[5] D. Verdon and G. McGraw, “Risk analysis in software design,” IEEE Security & Privacy, vol. 2, no. 4, pp. 79–84, 2004.

[6] B. Schneier, Secret & Lies, 1st ed., Weinheim: dpunkt.verlag, 2001, p. 408.

[7] G. Sindre and A. L. Opdahl, “Eliciting security requirements with misuse cases,” Requirements Engineering, vol. 10, no. 1, pp. 34–44, 2005.

[8] M. Schumacher, E. B. Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, Security Patterns. Chichester, England: John Wiley & Sons Ltd, 2005, p. 565.

[9] E. B. Fernandez and R. Pan, “A Pattern Language for Security Models,” Conference on Pattern Languages of Programs, 2001.

[10] T. E. Fægri and S. O. Hallenstein, “A Software Product Line Reference Architecture for Security,” in Software Product Lines, no. 8, Berlin, Heidelberg: Springer, 2006, pp. 276–326.

[11] T. Lodderstedt, D. Basin, and J. Doser, “SecureUML: A UML-Based Modeling Language for Model-Driven Security,” LNCS, vol. 2460, pp. 426–441, 2002.

[12] J. Jürjens, “Model-Based Security Engineering with UML,” in Foundations of Security Analysis and Design III, vol. 3655, no. 2, A. Aldini, R. Gorrieri, and F. Martinelli, Eds. Berlin, Heidelberg: Springer, 2005, pp. 42–77.

[13] C. Emig, S. Kreuzer, S. Abeck, J. Biermann, and H. Klarl, “Model-Driven Development of Access Control Policies for Web Services,” Proceedings of the 9th IASTED International Conference Software Engineering and Applications, vol. 632, pp. 069–165, 2008.

[14] J. D. Moffett, C. B. Haley, and B. Nuseibeh, “Core security requirements artefacts,” Department of Computing, Milton Keynes, 2004/23, 2004.

[15] J. Rushby, “Security requirements specifications: How and what,” Symposium on Requirements Engineering for Information Security (SREIS), vol. 441, 2001.

[16] C. Emig, F. Brandt, S. Kreuzer, and S. Abeck, “Identity as a Service-Towards a Service-Oriented Identity Management Architecture,” LNCS, vol. 4606, pp. 1–8, 2007.

[17] E. Yuan, J. Tong, B. Inc, and V. McLean, “Attributed Based Access Control (ABAC) for Web Services,” 2005 IEEE International Conference on Web Services, 2005.

[18] K. Lee, K. C. Kang, and J. Lee, “Concepts and Guidelines of Feature Modeling for Product Line Software Engineering,” in Software Reuse: Methods, Techniques, and Tools, vol. 2319, no. 5, C. Gacek, Ed. Berlin, Heidelberg: Springer, 2002, pp. 62–77.

[19] A. Dikanski, C. Emig, and S. Abeck, “Integration of a Security Product in Service-oriented Architecture,” in 2009 Third International Conference on Emerging Security Information, Systems and Technologies, Athens, Greece, 2009, pp. 1–7.

[20] A. Dikanski and S. Abeck, “A View-based Approach for Service-Oriented Security Architecture Specification,” in The Sixth International Conference on Internet and Web Applications and Services, St. Maarten, The Netherland Antilles, 2011.

[21] T. Imamura and M. Tatsubori, “Patterns for Securing Web Services Messaging,” OOPSLA Workshop on Web Services and Service Oriented Architecture Best Practice and Patterns, pp. 1-8, 2003.

[22] N. A. Delessy and E. B. Fernandez, “Patterns for the eXtensible Access Control Markup Language,” Proceedings of the 12th Pattern Languages of Programs Conference (PLoP2005), pp. 7–10, 2005.