

Integration of Security Patterns in Software Models based on Semantic Descriptions^{*}

PhD Student: Diego Ray. Advisors: Antonio Maña and Mariemma I. Yagüe
Computer Science Department
University of Malaga
diegoray@lcc.uma.es

Abstract. Security and reliability issues are rarely considered at the initial stages of software development and are not part of the standard procedures in development of software and services. Therefore, our objective is to develop and implement, based on UML, a tool-supported integrated framework for business model driven security engineering supported by semantic description mechanisms and formal methods. We will define a new security engineering process, in which security mechanisms and existing security and reliability solutions are integrated in terms of security patterns. Furthermore, it is also our objective to develop supporting techniques for open development platforms to consider security issues.

1 Introduction

Twenty five years ago, the typical reaction to questions about “Information Systems Security” was another question “what is security?”. A decade later the response to the same concern was “security is an added value service”. And over the last few years the response has become “security must be built-in”. Although the latter response acknowledges the need for integrating security into the design and development phases of information systems, in practice this integration has not yet been achieved. In current system development processes, security requirements are identified independently for each system component, such as servers storing and processing sensitive data or communication links using open networks, without explicitly considering the interdependences between them. However, current pervasive technology, ambient intelligence and mobile services raise highly complex security concerns regarding identity and privacy protection. These emerging technologies need sophisticated approaches in determining the security requirements and then providing adequate security solutions. Furthermore, in addition to the security mechanisms used within a system, the overall security depends on a variety of other factors not covered by conventional software engineering processes (social context and human behaviour, IT environments, and even protection of the physical environment of systems ...)

Today rather standard security services or countermeasures (such as symmetric encryption, public key infrastructures, firewalls and intrusion detection systems) are selectively employed for fulfilling typical security requirements such as authentication or confidentiality. Such security mechanisms provide “isolated” functionality, which is not always appropriate for the specific system to be developed. They may even not be effective for addressing the potential threats introduced by emerging environments like grid computing and wireless ad-hoc networks. On top of

^{*} This work is partially supported by Spanish MCYT project TIC2003-08184-C02-01

that, contextual information such as legal requirements imposed by related laws and regulations are usually overlooked. Experience shows that even for relatively small systems, it is difficult and often error-prone to fulfil security requirements by simply combining existing security mechanisms. Nevertheless, it is true, that currently there is no development process automatically supporting the precise identification of security requirements and the correct choice of security mechanisms. On the contrary, expert knowledge in IT security is a prerequisite for applying existing security mechanisms. Unfortunately, such expertise is usually not available to the average system /software developer. Moreover, for large scale systems, the complexity of the current and emerging computing environments and their security requirements has increased to the point where experience is not enough to guarantee adequate security.

It is therefore evident that current 'ad-hoc' approaches cannot support a complete and rigorous treatment of security starting from the requirements elicitation process up to system implementation. Security requirements and available security services are not linked with the remaining system components and neither with the context of the system. This fact explains to a large degree one of the main reasons for the proliferation of software problems caused by security design flaws. Such flaws are hard to detect, and are often the cause for notoriously expensive system reorganisation and adjustments processes. Different factors are contributing to this situation: first, security is a non-functional requirement, thus it is hard to capture with standard software design techniques; second, security is partly a social and not a technical problem, thus it is hard to capture in standard design languages; third, there is no homogeneous way to represent security concerns and security mechanisms at different levels of software description, thus it is hard to trace security issues along the phases of software development. All these factors highlight that current software and system engineering techniques are not sufficient for the development of secure systems. Although the current praxis of add-on security does not necessarily result in non-secure systems, it has significant weaknesses such as: a) the security requirements of the entire system may not all be satisfied by eliminating single, isolated weaknesses in the system. Furthermore, their fulfilment can not be confirmed since the requirements have not been clearly specified in the first place; b) different implementations of the system cannot be compared with respect to their security properties, and c) the security mechanisms introduced are not directly linked to the specific security requirements of the system. Linking between security mechanisms and requirements is particularly important in case of failures of security mechanisms, when the effects of attacks on the overall security of a system must be determined.

In order to achieve high assurance of the security of complex systems in open communication environments, the state of the art of development processes for secure systems has to be considerably improved. Such improvements should include methods for precise specification of security requirements, and automated tools for classifying, selecting, adapting and re-organising existing security services, as well as for integrating them into software systems under development. Furthermore, special constructs for the expression of security requirements have to be integrated into existing modelling languages to support rigorous treatment of security issues throughout the development process.

2 State of the art

There is very little work concerning the full integration of security and systems engineering from the earliest phases of software development. Although several approaches have been proposed for some integration of security, there is currently no comprehensive methodology to assist developers of security sensitive systems. Lack of support for security engineering in those approaches for software systems development is usually seen as a consequence of: (i) security requirements being generally difficult to analyse and model, and (ii) developers lacking expertise in secure software development. All this becomes a special concern when considering complex security requirements such as those associated to applications in e-commerce, e-government and e-health scenarios.

Existing approaches are not comprehensive enough in the sense that they focus either on some special stage of development, e.g. on design or implementation, or on a specific security aspect such as access control. Moreover, they typically offer no guidance on how they can be integrated into current component or model based system development methods. Empirical studies confirm this view [18,19]. We will provide a comprehensive and integrative approach supporting the integration of security and systems engineering.

The Unified Modelling Language (UML) has become the de-facto standard notation for system development, is well supported by platforms and tools, and suited for defining designs at high abstraction levels. It offers an excellent opportunity for closing the gap between software and security engineering and for supporting the development of security-critical systems in an industrial setting. An overview of the current state of the art concerning approaches to integrate security and system engineering within UML-based frameworks is included.

Concerning security for business processes, security aspects are introduced into business process and workflow models in [8,14]. In this approach, security requirements are considered as inherent to business transactions and dependent on the circumstances of the applications. Although, originally not based on UML, later work [9] includes an extension intended to support the development of an abstract UML-based business process specification. A model driven architecture approach to security engineering, called Model Driven Security, is introduced in reference [2]. This approach, called SecureUML [1], integrates role-based access control policies into a UML-based model-driven software development process.

UMLsec [10] is proposed as an extension of UML for modelling security properties of computer systems, according to suggestions in [5]. UMLsec uses standard extension mechanisms to introduce new semantics into UML models but only addresses a few specific security requirements. Therefore, it does not support a security requirements engineering approach, like our proposed security engineering process. Another recent approach proposes to integrate security and systems engineering using elements of UML within the Tropos methodology [3,13].

Use cases have been used to capture and analyse security requirements, cf. abuse cases [17], defined as an interaction between one or more actors and a system with a harmful result, and misuse cases [12], which describe functionality that the system should block. Use cases can describe requirements for special scenarios and are suitable to improve the intuitive understanding of security requirements.

UML lacks a formal semantics [7]. However, a large community of researchers are currently exploring ways to close the gap between UML and formal specification

languages. The work is supported by the UML Precise Group (PUML) [22], created for investigating the completeness of the UML semantics and for developing new approaches to use UML more precisely. We believe that our results can be complementary to this work, by adding semantics for security and reliability requirements

Several projects have been dedicated to issues that are relevant in the present context. The SEMPER project (Secure Electronic Marketplace for Europe) [11] aimed at providing a comprehensive security framework for electronic commerce and business applications, concentrating on security architecture and services rather than secure systems development. The project COPS (Commercial Protocols and Services) [15] also concentrated on security services. COPS intended to enable the design of an infrastructure for marketplaces supporting all phases of a market transaction. CORAS [6], on the other hand, aimed at developing a tool-supported framework for model-based risk assessment of security sensitive systems. The methodology gives recommendations for the use of UML-oriented modelling in conjunction with risk assessment. Most recently, the CASENET project the main objectives of which are the development of methods for the design and analysis of security protocols started work on integrating security requirements specification into the process of application development.

Security engineering with patterns is currently a very active area of research [16]. The Open Group was preparing a book on the subject [21]. Research into investigating a template for security patterns that is tailored to meet the needs of secure system development has been recently reported in [4], where the UML notation is used to represent structural and behavioural aspects of design, and formal constraints to the patterns are used to enable verification. However, security patterns are usually not precisely described and therefore, automated tools for classification, selection and composition are not yet available. Here, our work can improve security engineering using security patterns.

Being a de facto industry standard, UML is very well provided with industrial-strength CASE tools. A list of them would be very extensive. However, no existing tool supports a security requirements engineering approach. Therefore, a central task of our work is to implement these extensions.

The basic standard for security engineering is the Systems Security Engineering Capability Maturity Model (SSE-CMM) [20]. The model highlights the relationship between security engineering and systems engineering, regarding the former as an integral part of the latter and not an end unto itself.

This overview shows that, based on the variety of different approaches, several special security properties can be considered more or less rigorously in the development process. In contrast to these solutions, we will provide an integrated security engineering process with tool support, which allows rigorous treatment of security and reliability requirements and makes existing security solutions available for security engineering.

3 Approach and Objectives

The security engineering process developed will draw together the areas of software and systems engineering, security engineering, and formal methods for the design and analysis of secure systems. In this manner, formal methods, a cornerstone for rigorous

security engineering, can be made available for use within software engineering processes. Our vision of the process is that it must provide support to security engineers for the development of formally proven security solutions, but also to software engineers in the specification of their security requirements, the validation of their models against the security requirements, and the integration of proven security solutions in their models. We believe that this approach can be generalized to other fields, such as real time systems, because it is based on defining a general collaborative framework that separates the work of experts in a specific field (security in our case) allowing software developers to take advantage of the work of the experts and to integrate proven solutions into their models.

As shown in figure 1, security engineers will analyse and define security properties. They will also use pattern design, formal modelling and validation tools to create new specialized security solutions. They will formally prove that these solutions indeed provide the desired security services, and finally they will integrate them into a library of security patterns. This library, in turn, will be used by system developers to fulfil security requirements in their models.

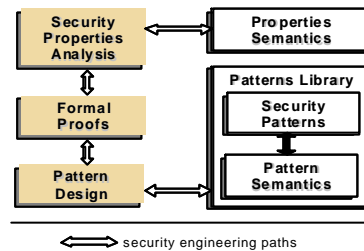


Fig. 1. Security engineers interface to the generic software development process

The ultimate objective is to provide methodologies and tools for the generation of executable systems with fully configured security infrastructures, starting from as early as the business process model. Reusability will be promoted through the development of a library or repository of security services with the help of security patterns. The proposed methodology is intended to make extensive use of techniques based on formal methods, and to be consistent with model-driven development, as defined by the Object Management Group MDA initiative.

Our software development process follows the approach of Iterative and Incremental Development (IID) and integrates security engineering techniques based on the notion of security patterns.

A variation of the basic Boehm spiral model for software development processes, one of the most popular examples of IID, is presented in figure 2. This model forms the basis of most modern software engineering processes. In this model, software is developed in several iterations, each one divided into different phases. In this particular figure, a full cycle does not represent one release of the software, but just one step in the evolution from an abstract to a more detailed model. The right side of figure 2 shows the integration of the results of the security engineering work as shown in figure 1 into the engineering / development phase of the generic software engineering process. It is at this point that our tools will be used to support developers of secure software. For instance, automated tools will support the developers in the

selection of a particular security solution, the adaptation of the solution to the context where it will be finally applied, the analysis of the consequences of changes and the coherence between models at different levels of abstraction, etc.

A pattern describes a recurring problem that arises in a specific context and specifies a generic scheme with well-defined properties for its solution. Security patterns have appeared in the literature where they have been used only as informal and often managerial measures of security, without any formal treatment and/or support for reasoning. In our approach, we will deploy security patterns in a much broader sense. We will use security patterns to represent security services with specific profiles and solutions for different environments. Each pattern will be semantically described using XML-based metamodels. These semantic descriptions will include, among other characteristics, abstraction level, type of solution, applicability, context conditions, pre and post-conditions, definition of parameters and of course, the security properties provided by the pattern. In our PhD work we will develop the necessary pattern language and methodology. Likewise, security properties will also be semantically described.

Our approach is based on the concept of a protocol as a series of semantic properties to be fulfilled. That is, a protocol can be defined through a series of semantic properties, such as ‘level of security: Confidential’, ‘Authentication is needed’, and so on... This semantic view of the security protocols enables us to move to more abstract levels and to incorporate semantics about the context of application of that protocol. Additionally, it enables the implementation of the same security property through different patterns, that is, different instantiations of the same security property class. As we have yet mentioned, our approach uses the concept of security pattern to represent existing security solutions. The semantic description of both, security properties and patterns enables i) the semantic categorization of patterns in a Pattern Descriptions Library, based on different parameters; ii) to consider the context of application of the pattern within the model, to locate the most appropriate pattern fulfilling the security requirements of the software model based on semantic information; iii) to check if interaction of the security properties implemented in different patterns in the model does not cause cancellation or deterioration of some of them; iv) to validate the correctness of new security patterns incorporated by a security engineer.

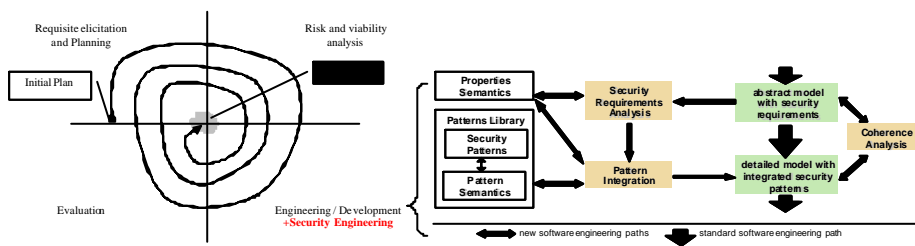


Fig. 2. Generic software development process with integrated security engineering

The results of this work will enable the development of a library of security patterns. This library is not a simple set of “best practices” or recommendations (mostly at the operational level), like those proposed in the literature, but a precise,

well-defined and automated-processing enabled repository of security mechanisms. The most relevant feature is the incorporation of rich and precise semantic descriptions of the patterns. The extensive use of semantic descriptions will enable the use of automated reasoning mechanisms capable of solving problems such as pattern composition and adaptation. Patterns will be categorized into distinct abstraction categories, corresponding to different phases of the security engineering process, to different types of security measures (technical, organizational, etc.) and to different levels of formalism (e.g. formal patterns for security services that implement formally specified requirements and informal patterns that address security issues in the organizational environment). Thus, our security patterns will address problems occurring at the business system analysis phase (e.g. risk analysis), during requirements specification, design and implementation, as well as problems of system deployment (e.g. problems relating to the security of the system environment). However, the main emphasis will be given to security patterns that guide software developers in their effort to fulfil security requirements through the design and implementation of security solutions that provide reliable security services.

The treatment of security aspects cannot be separated from the treatment of other aspects (functional, performance, etc.) within the full software development cycle. Additionally, it should be possible to automatically process specifications in order to validate them and to find appropriate solutions in formally proven “security knowledge bases”. Therefore, part of our work will concentrate on the integration of security requirements specification and security patterns into the development process. Figure 3 depicts the structure of the automated tools supporting the integrated process.

In particular, we will develop methods that support the analysis of security requirements specified in the system models in order to find inconsistencies in the model (e.g. publishing confidential information on a public repository) with the aid of semantic descriptions of the security properties.

Next, based on the security requirements specification, methods will be developed for automatic identification and retrieval of the appropriate security patterns that fulfil the security requirements specified in the specific user model environment. We envisage that, for complex security requirements, it will sometimes be necessary to compose different patterns before they can be integrated in the user model. The development of mechanisms to support composition, adaptation and integration of security patterns in user models and the development of methods to analyse the results of pattern composition constitute a central part of our research work.

Finally, selected patterns will be adapted, instantiated and integrated in the system model. Here, interdependencies between the newly integrated patterns, previously introduced security patterns and the functional behaviour of the system have to be considered.

Summarising, the main objectives of the PhD are:

- To create a framework for the semantic description and management of security properties and patterns.
- To define mechanisms to automate the analysis of security-enhanced models in order to find appropriate security solutions (patterns) to fulfil the security requirements and to allow the integration of these patterns in the model. This integration will possibly require the combination and adaptation of the pattern according to the context of the model.

- To define mechanisms to validate the integrity of the model with respect to the security requirements in order to guarantee that the subsequent changes on the model preserve the security properties. These mechanisms will be possible thanks to the semantic description of both security properties and patterns, and automated inference procedures.

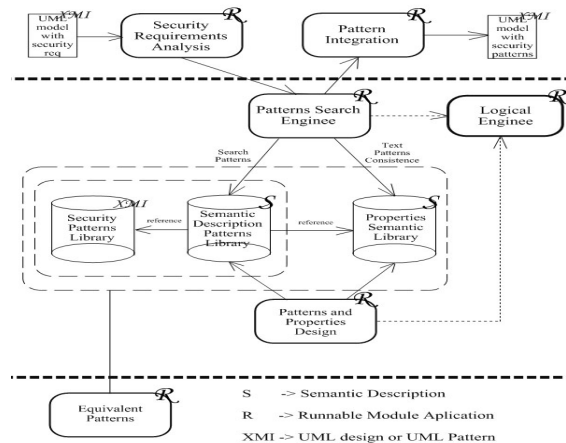


Fig. 3. Operational Architecture of the Automated System

4 Conclusions

The current practice of security engineering is hampered by the fact that it is not considered as an integral part of system engineering. As a result, implemented security solutions are often inadequate, incomplete, and flawed. Although system development is nowadays a well-established engineering discipline, it also suffers from the lack of integration between system and security engineering, since security is becoming one of the leading fields within the information technology area. The proposed work will contribute to the solution of these problems based on the integration of security engineering and software engineering practices, the full development of the concept of security pattern and the use of semantic description and formal methods technologies.

5 References

1. D. Basin and J. Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. 5th International Conference on the Unified Modeling Language, 2002. Lecture Notes in Computer Science 2460.
2. D. Basin, J. Dose, and T. Lodderstedt. Model Driven Security for Process-Oriented Systems. 8th ACM Symposium on Access Control Models and Technologies, 2003.
3. J. Castro, M. Kolp, and J. Mylopoulos. A requirements-driven development methodology. In Proc. of the 13th Int. Conf. on Advanced Information Systems Engineering, CAiSE'01
4. B. H.C. Cheng, S. Konrad, L. A. Campbell, and R. Wassermann. Using Security Patterns to Model and Analyze Security Requirements. Technical Report MSU-CSE-03-18, Department of Computer Science, Michigan State University.

5. P. T. Devanbu and S. G. Stubblebine. Software engineering for security. In Proceedings of the 22th International Conference on Software Engineering (ICS-00), ACM Press.
6. T. Dimitrakos, D. Raptis, B. Ritchie, and K. Stølen. Model based security risk analysis for web applications. In Proc. Euroweb 2002, British Computer Society, 2002.
7. R. France, A. Evans, K. Lano, and B. Rumpe. The UML as a formal modeling notation. OOPSLA'97 Workshop on Object-Oriented Behavioral Semantics.
8. G. Herrmann and G. Pernul. Viewing business process security from different perspectives. Proceedings of the 11th International Bled Electronic Commerce Conference "Electronic Commerce in the Information Society", Slovenia, 1998.
9. P. Herrmann and G. Herrmann. Security-Oriented Refinement of Business Processes. Proceedings of the 5th International Conference on Electronic Commerce Research (ICECR-5), ATISMA, IFIP, Montreal, October 2002.
10. J. Jürjens. Towards development of secure systems using UMLsec. Lecture Notes in Computer Science 2029, 2001.
11. G. Lacoste, B. Pfitzmann, M. Steiner, and M. Waidner (Eds.): SEMPER – Secure Electronic Marketplace for Europe. Lecture Notes in Computer Science 1854, 2000.
12. J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In 15th Annual Computer Security Applications Conference (ACSAC'99), 1999.
13. H. Mouratidis, P. Giorgini, and G. Manson. Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems. Proceedings of the 15th Conference On Advanced Information Systems Engineering (CAiSE'03).
14. W. Röhm, G. Herrmann, and G. Pernul. A language for modelling secure business transactions. In IEEE Annual Computer Security Application Conference (ACSAC'99)
15. A.W. Röhm and G. Pernul, COPS: A Model and Infrastructure for Secure and Fair Electronic Markets. International Journal on Decision Support Systems, Elsevier, 2000.
16. M. Schumacher and U Roedig. Security engineering with patterns. In Pattern Languages of Programs 2001, Monticello, IL, 2001.
17. G. Sindre and A. L. Opdahl. Eliciting security requirements by misuse cases. In Proc. TOOLS Pacific 2000, pages 174-183, November 2000.
18. T. Tryfonas, E. Kiountouzis, and A. Poulymenakou. Embedding security practices in contemporary information systems development approaches. Information Management & Computer Security, Vol 9 Issue 4, 2001 pp 183-197.
19. R. Vaughn, R. Henning, and K. Fox. An empirical study of industrial security engineering practices. Journal of Systems and Software, Nov. 2001.
20. Systems Security Engineering Capability Maturity Model (SSE-CMM), Model Description Document, Version 3.0. Available at <http://www.sse-cmm.org/model/ssecmmv2final.pdf>.
21. Draft version available at <http://www.opengroup.org/security/gsp.htm>.
22. UML Precise Group, <http://www.puml.org>.