A business process-driven approach to security engineering

Antonio Maña, José A. Montenegro
Computer Science Department, Universidad de Malaga, Spain
{amg,monte}@lcc.uma.es

Carsten Rudolph
Fraunhofer Institute for Secure Telecooperation – SIT, Germany
Carsten.Rudolph@sit.fraunhofer.de

José Luis Vivas
Hewlett Packard Laboratories, UK
jsv@hplb.hpl.hp.com

Abstract

A challenging task in security engineering concerns the specification and integration of security with other requirements at the top level of requirements engineering. Empirical studies show that it is common that end users are able to express their security needs at the business process level. Since many security requirements originate at this level, it is natural to try to capture and express them within the context of business models where end users feel most comfortable and where they conceptually belong. In this paper we develop these views, present an ongoing work intended to create a UML-based and business process-driven framework for the development of security-critical systems and propose an approach to a rigorous treatment of security requirements supported by formal methods.

1 Introduction

The view that the main challenges facing the development of secure systems are non-technological in nature is now widespread. Security involves people and organizations, relationships among people, and relationships between people and systems. Accordingly, security of computer systems is almost always compromised by exploiting vulnerabilities in the environment in which they are inserted and the way they are employed, not by breaking dedicated mechanisms such as cryptographic protocols. However, there is currently little systematic support for software engineers to design secure systems.

However, recently the emphasis is shifting away from technology to understanding the nature of the organization [7]. Technology and business are being combined in order to provide solutions to the needs of secure systems, and there is an increasing awareness that security concerns should influence every phase of software development, from requirements engineering to deployment.

Business modelling is a technique to model business processes providing support for the development of adequate automated solutions, e.g. for e-business systems. It allows developers to focus on the environment in which the software system will work, the roles and responsibilities of the actors involved, and the objects that are handled by the business. Thus, business modelling focuses also on interacting behavior among humans and other agents within an organization, regardless of whether computers are involved [5].

Since most security threats originate at this level, business modelling becomes an important tool for studying and capturing security requirements. Moreover, experience shows that it is at the business process level that end users are able to express their security needs [19].

Our intention is to develop a UML-based framework for representing security semantics in an integrated development environment including the business process and the system model. This representation must enable a rigorous treatment of security requirements, which must be specified and analyzed formally. The focus is on the entire system including the environment or context in which the software system will eventually be inserted. This approach has the advantage that it facilitates the integration of security with other requirements. Security should be viewed as a compo-
ment of a larger system being secured, and the focus must fall upon the latter’s purpose and threats it will face [19]. Since security is only one of many requirements that developers must take into consideration, this approach provides support for the integration of security concerns into ordinary software development.

2 Security Requirements in Software Design

So far, there is little systematic support for the development of secure software, and the specification of security policies is typically not integrated with general software development [2, 18]. Security is usually added as an afterthought to system requirements and design [6]. Work in the area of system security has focused mostly on the technological part of the subject, such as cryptography, security protocols, firewalls, etc [18], resulting in what some authors call the “technology trap” [7]. This discrepancy may derive from the fact that these technologies are much easier to understand and to secure [16]. However, recently the emphasis is shifting away from technology to understanding the nature of the organization [7]. There is nowadays also an increasing awareness that security concerns should influence every phase of software development, from requirements engineering to deployment.

Very little work has been done so far on expressing security requirements within the business process model, and most focus only on particular aspects of security. An exception is the language ALMOST [15]. ALMOST was intended as a notation to enable a firm to formulate market transactions security at a high abstraction level. The basic approach is to consider the security requirements as immanent to business transactions and dependent on the circumstances of the applications, not barely on the integrity of messages resulting from the conditions of communications along insecure channels. A three-layered architecture is proposed, with graphical concepts to specify security requirements at the upper layer, a repository of already modelled solutions at the middle layer, and a repository with hard- and software building blocks at the lowest layer. Four perspectives are defined yielding a comprehensive view of a business process: informational, functional, behavioral, and organizational, to which a fifth new perspective is added, the business process perspective. Security requirements specified in the business process model must also be represented and analyzed in the other models.

The approach of UMLsec [13], which can be described as formal, concentrates on the design phase of system development. The idea is to use the language extension mechanisms of UML to encode security semantics. Cryptographic protocols are specified with the help of sequence diagrams within the general framework of a security-enhanced UML. Sequence diagrams are given a formal interpretation in the specification framework Focus [3].

Although UML seems to be the most suitable notation for closing the gap between the business and the system model, many believe that UML has limitations as a business model specification language. Thus, other emerging industry standards such as BPML, BPEL4WS, WSFL, and XLang are also based on the notion of business process.

A promising line of work for providing automated support in the development of general systems, including security aspects, is represented by the use of patterns [10], particularly security patterns [17]. Security patterns are intended to capture security expertise in a constructive way and to make this expertise available to developers who are not security experts. Although most patterns were supposed to be design patterns, as noted in [4], many software design patterns can be used at higher levels of abstraction.

3 The FML (Formal Methods and Modelling Language) Framework

3.1 Computer-aided Security Engineering

An early consideration of security properties has been proved to be beneficial for the deployment of secure systems. Supporting the specification of the different security properties in the UML model is an important advance. However, the difficulty and specificity of the security properties require automatic tools to support the design and deployment of secure systems. Our approach is based on:

1. Extending the UML model to capture security requirements. We propose extensions of UML that can represent security requirements at different levels of abstraction from business to implementation models.

2. Integrating the specification of the security properties in the UML model in such a way that they can be processed by automatic means. For this purpose we use tagged values: one of the most basic, but also most powerful, extension mechanisms of UML. Opposed to other solutions, we do not consider stereotypes to be suitable for our purposes. The basic reason is that, although conceptually suitable, stereotypes do not provide the practical tools required in our environment. The basic reason is that, although conceptually suitable, stereotypes do not provide the expressiveness required in our environment. A security property is usually a complex expression with modifiers and parameters that stereotypes are not able to capture. However, we use stereotypes during the first stages of the development, especially in the business model.

3. Formal semantics for security requirements. Definitions of security properties in terms of formal language
theory are used to define formal semantics for the security requirements specified in the UML model. This supports a rigorous treatment of security requirements throughout the system design process.

4. Using those specifications to assist the developer in the design of secure solutions. Our framework allows developers to incorporate proved security solutions. Reusability and reliability are two important advantages of this approach.

5. We use the concept of pattern to represent proved security solutions. The security solutions are represented as UML patterns and included in a "library" of solutions proved by means of formal methods.

The modelling tools in our proposal are based on graphical representations. However, in order to process the specific and to detect and correct the possible inconsistencies related to the security requirements expressed in different parts of the specification, we take advantage of the flexibility and interoperability of the XMI representation of the UML models. The XMI representation is the link between the different abstraction levels in our framework. XMI is also used for the storage and automatic processing of the security patterns. In order to be able to automatically include those patterns in the systems being deployed, we use semantic descriptions of the solutions expressed in XML.

![Figure 1. FML Framework](image)

An overview of the full development process is shown in Figure 1. In the first stages the users specify their security requirements in the UML model. Once the security requirements are incorporated in the model, our framework processes the model identifying those elements with associated security properties. For each of those elements, or groups of them, our system locates those patterns that are suitable for the specific case in the security solutions library. In order to identify the most suitable solution in the library, we take into account different parameters such as the desired property, actors, pre and post conditions, restrictions and environment. This search is conducted in the semantic descriptions of the solutions. When a suitable solution is found, developers can incorporate it in their model automatically. This is possible because both, the model and the protocols and solutions of the library are described in XMI.

3.2 Applying Formal Methods

This section presents new ideas on the use of formal methods in the context of business process models. As explained above, this paper proposes to specify security requirements as early as possible in the system development process, namely on the level of business process models. However, in order to derive a secure implementation, these security requirements have to be exactly specified and then propagated throughout the whole development process.

**Formal security models for business processes**

In order to support a rigorous treatment of security requirements in the system development process, security requirements must be formally specified. Therefore, formal semantics with respect to the security requirements are needed even on the abstract level of business processes. For security sensitive systems, it is useful to extend the business process model by a formal security model that specifies a system behaviour satisfying all security requirements found in the business process model. Only the requirements of the business process, that have to be enforced by security measures have to be included in the formal security model.

The following well-known approach to describe a system behaviour can be used to build formal security models for business process specifications with security requirements. The behaviour $S$ of a system can be formally described by the set of its possible sequences of actions. Therefore $S \subseteq \Sigma^*$ holds where $\Sigma$ is the set of all actions of the system and $\Sigma^*$ the set of all finite sequences of elements of $\Sigma$, including the empty sequence denoted by $\epsilon$. This terminology originates from the theory of formal languages [8], where $\Sigma$ is called the alphabet, the elements of $\Sigma$ are called letters, the elements of $\Sigma^*$ are referred to as words and the subsets of $\Sigma^*$ as formal languages. Words can be composed: if $u$ and $v$ are words, then $uv$ is also a word. This operation is called the *concatenation*; especially $\epsilon u = u \epsilon = u$. A word $u$ is called a prefix of a word $v$ if there is a word $x$ such that $v = ux$. The set of all prefixes of a word $u$ is denoted by $\text{pre}(u)$; $\epsilon \in \text{pre}(u)$ holds for every word $u$. We denote the set of letters in a word $u$ by $\text{alph}(u)$. Formal languages which describe system behaviour have the characteristic that $\text{pre}(u) \subseteq S$ holds for every word $u \in S$. Such languages are called *prefix closed*. System behaviour
is thus described by prefix closed formal languages.

Several approaches exist to formalize security properties of a system formalized in terms of formal languages. These properties include different variants of authenticity [12]. Furthermore, confidentiality properties can be defined similar to possibilistic non-interference properties as described in [14]. Apart from these elementary security properties, business processes can have more complex security requirements. For example, in some electronic commerce applications fairness is essential. Different partners have different goals and in some phases of the process, in so-called binding phases, either all partners achieve their respective goals or no partner achieves any goal. If one partner has obligations to fulfill the other partner needs proofs about these obligations. For example, if one partner has the obligation to pay money, the other partner needs a proof that he owns him the particular amount of money. Formalisms to describe complex security requirements of business processes have been proposed in [11].

We can summarize that existing formalisms are sufficient to build formal security models for business process for a variety of security requirements. Further work is needed to exactly define the formal semantics of the business process model such that these definition can be applied.

**Formal propagation of security requirements**

The previous section describes the first step towards a rigorous treatment of security requirements, namely the formal specification in terms of a formal security model for the business process. In the next steps of the system development process these security requirements have to be propagated from one layer of abstraction to the next finer layer and then the satisfaction of the resulting security properties has to be shown for every model. Formal security models as described in the previous section support the propagation of security requirements.

Different formal models of the same system are partially ordered with respect to their levels of abstraction. Formally, abstractions are described by so called alphabetic language homomorphisms. These are mappings $h^* : \Sigma^* \rightarrow \Sigma'^*$ with $h^*(xy) = h^*(x)h^*(y)$, $h^*(\varepsilon) = \varepsilon$ and $h^*(\Sigma) \subseteq \Sigma' \cup \{\varepsilon\}$. So they are uniquely defined by corresponding mappings $h : \Sigma \rightarrow \Sigma' \cup \{\varepsilon\}$. In the following we denote both the mapping $h$ and the homomorphism $h^*$ by $h$.

Now, language homomorphisms can be used to propagate the security requirements between levels of abstraction. Appropriate formalizations for authentication and non-repudiation properties and sufficient conditions for language homomorphisms to transport these properties have been proposed in [12]. Language homomorphisms to propagate confidentiality are subject to further work.

We now illustrate the approach with a small example: A client accesses data from a database and the security requirement is the authenticity of the data. We denote this property with $p$. The formalization of $p$ intuitively says: An action by the database sending the data must have happened with exactly this data. In a business process model there may be no actions between sending and receiving data. The security requirement is relatively easy to understand on this level of abstraction. A finer layer of abstraction may introduce intermediate steps, for example communication using the Internet and intermediate application servers. As the system gets more complex it may be difficult to find the appropriate representation of the abstract security requirement as specified in the business process model. However, it should be relatively easy to assign actions of the finer system to actions at the business model. If the so defined language homomorphism $h$ satisfies the conditions to transport authenticity, the appropriate representation can be derived as the inverse $h^{-1}(p)$ of the property on the formal security model of the business process. Furthermore, if one can show that the homomorphic image of the behaviour of the finer system is equal to the formal security model of the business process, one can conclude that $h^{-1}(p)$ is satisfied by the refined system as well.

4 Scenario: CASENET project

The CASENET[1] trial partners provide details of applications for the validation and evaluation of CASENET methods and tools, representing a constant feedback cycle between research partners. The trials cover three different activity domains: (i) Business/Citizen to Government: Online access to accounting and tax information, a prototype application by the City of Cologne, (ii) Business-to-Business: Integrating Online Contracting and E-notary applications in various deployment scenarios by NetUnion, and (iii) Citizen Access to Administrative Services: e-education administration platform under development by SADIENL, named PASEN, which we use in this paper to illustrate our proposal.

PASEN is a web portal allowing the educational e-management between the legal tutors and the administrative staff or the teachers. The services to be offered include integration of e-administrative services allowing the management of licenses, permits and training activities for teachers, as well as such basic services as pre-admission, students registration, student training and grant management.

In this section we focus on the PASEN services that need to be secured, and develop processes, using the Erikson-Penker annotation [9], to achieve the respective service. The services specified by the partner were: (i) registration, (ii) grant, (iii) pre-admission, (iv) homework, (v) student absences and (vi) examinations. We group the tasks homework, student absences and examinations in a task named student training, since the original tasks manages similar
The most challenging task here is how to map the requirements at business model level of abstraction into the distinct phases of software development. Our aim is to develop and apply security patterns already at the business model, both to transform a purely functional description of a process into one which integrates the security requirements, and to decompose a representation at a certain level of abstraction into a more refined one. In general, the hardest task is not to encode security semantics in distinct views and models of a system, but rather to develop methods of transformation from one representation to another which guarantee that security properties are preserved. Furthermore, in our approach this top-down propagation of the security requirements is supported by the use of formal methods.

5 Conclusion

We have presented ongoing work intended to establish a business process-driven software development framework based on the UML notation and integrating security requirements at the early stages of software development. Basically, our suggestion is to extend UML in order to express security requirements, to make extensive use of security patterns defined in an XML-based notation to automate parts of the development process, and to apply specification and analysis methods currently under development to verify relevant parts of the proposed solutions. We believe that the emergence of a new generation of tools and XML-based notations intended for business process management renders this approach a very promising one.

The most challenging task here is how to map the requirements at business model level of abstraction into the distinct phases of software development. Our aim is to develop and apply security patterns already at the business model, both to transform a purely functional description of a process into one which integrates the security requirements, and to decompose a representation at a certain level of abstraction into a more refined one. In general, the hardest task is not to encode security semantics in distinct views and models of a system, but rather to develop methods of transformation from one representation to another which guarantee that security properties are preserved. Furthermore, in our approach this top-down propagation of the security requirements is supported by the use of formal methods.

References