Information and Software Technology 000 (2017) 1–21

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A formal approach to derive an aspect oriented programming-based implementation of a secure access control filter

Amel Mammar[a,*], Thi Mai Nguyen[a], Régine Laleau[b]

[a] SAMOVAR, Télécom SudParis, CNRS, Université Paris-Saclay, 9 rue Charles Fourier 91011 EVRY, France
[b] Université Paris-Est, LACL, UPEC, IUT Sénart Fontainebleau, France

ARTICLE INFO

Article history:
Received 8 November 2016
Revised 3 July 2017
Accepted 7 August 2017
Available online xxx

ABSTRACT

Context: Nowadays, Information Systems (IS) are at the heart of most companies and constitute then a critical element that needs an adequate attention regarding security issues of sensitive data it manages.
Objective: This paper presents a formal approach for the development of a filter to secure access to sensitive resources of information systems.
Method: The proposed approach consists of three complementary steps. Designers start by modeling the functionalities of the system and its security requirements using dedicated UML diagrams. These diagrams are then automatically translated into a formal B specification suitable not only for reasoning about data integrity checking but also for the derivation of a trustworthy implementation. Indeed, a formal refinement process is applied on the generated B specification to obtain a relational-like B implementation which is then translated into an `AspectJ` implementation, connected to a SQL Server (release 2014) relational database system. Such a generation is performed following the aspect oriented programming paradigm which permits a separation of concerns by making a clear distinction between functional and security aspects.
Results: A systematic formal approach to derive a secure filter that regulates access to the sensitive data of an information system. The filter considers both static and dynamic access rules. A tool that supports the proposed approach is also provided.
Conclusion: The approach has been applied on several case studies that demonstrate that the development of a tool permits to free the developers from tedious and error-prone tasks since they have just to push a button to generate the `AspectJ` code of an application.

1. Introduction

An Information System (IS) is the part of an organization responsible for collecting and manipulating all its relevant and sensitive data. Nowadays, it is at the heart of most companies and constitutes then a critical element that needs an adequate attention regarding security issues. Indeed, an information system often interacts with humans or other systems by exchanging information and any security breach may cause serious and even irreversible consequences. To avoid such risks, a common way is to control access to information systems by defining security rules. Roughly speaking, a security rule specifies, for an authenticated user, which actions are allowed/forbidden according to his/her current role and context.

To ensure the security of a system, many types of access rules may be required. These rules can be classified into two main classes: static and dynamic. Static access rules refer to a given single moment of the system (*i.e.*, the values of the data are taken at the same moment) whereas dynamic access ones require to take the execution history of the system into account, that is the actions already performed in the system in general or by a given user in particular. For example in a hospital, a static access rule can be *"only a person with the role Doctor can make a diagnosis"*, whereas a dynamic access rule can be *"the person who performs a laboratory test cannot validate it"*. In addition to these kinds of rules, usual functional constraints, like for instance the maximum number of patients each doctor can treat, need to be considered. In this paper, we propose to use three UML-based languages: a class diagram to describe the structure of the data manipulated in the system together with their functional constraints, `SecureUML` [6,28] to deal with static access rules and adapted UML activity diagrams [42] for dynamic access ones.

* Corresponding author.
E-mail addresses: amel.mammar@telecom-sudparis.eu, amel.mammar@gmail.com (A. Mammar), thimai.nguyen@telecom-sudparis.eu (T.M. Nguyen), laleau@u-pec.fr (R. Laleau).

http://dx.doi.org/10.1016/j.infsof.2017.08.001
0950-5849/

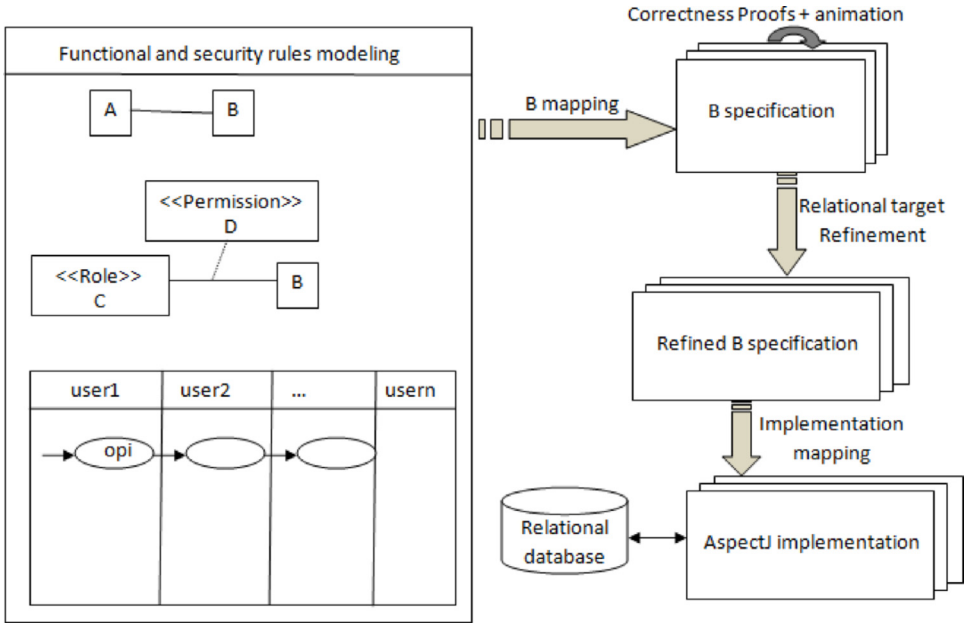

Fig. 1. An overview of the approach.

Even if the use of a graphical notation to express functional/security rules argues for an intuitive, synthetic and visual presentation of the considered system, it is often a source of ambiguities. Moreover, we need a unique language in which all the functional/security rules can be expressed in order to be coordinated to verify the consistency of all the security rules with respect to the functional aspects of the system. To this aim, we suggest to translate the obtained graphical models into a formal B specification that can be formally verified using the associated tools. We have chosen the B method because it is based on concepts that are easy to learn. Moreover, a reliable industrial free tool (`AtelierB` [11]) supports all the development stages. The obtained B specification is then taken as a basis for the development of a correct access control filter that ensures that the execution of each action of the system can happen only if all the specified functional/security rules are fulfilled. Indeed, a refinement process is applied to bridge the gap between the B abstract level and the chosen target implementation. We have already defined such a process for the development of trustworthy database applications that satisfy integrity constraints [30]. The approach presented here extends this previous work to take security constraints into account. Fig. 1 depicts the main steps of our approach that consist in:

1. modeling both functional and security rules using UML-based notations (class, `SecureUML` and activity diagrams);
2. mapping the graphical notations into a formal B specification which is proved to be correct thanks to the establishment of proof obligations;
3. refining the B specification towards a relational-like B implementation;
4. translating the relational-like B implementation into an executable `AspectJ` code.

The contributions of this paper are as follows:

- a set of generic rules to map the UML-based diagrams describing both static and dynamic security rules (`SecureUML` and activity diagrams) into B specifications. A generic specification of a B filter that permits to coordinate all the specified security rules is also generated (see Section 4).
- a set of mapping rules that translates the B specification corresponding to the translation of the `SecureUML` diagram into a set of SQL orders. The targeted database management system (DBMS) is SQL server, release 2014 (see Section 6).
- a set of mapping rules that generates an `AspectJ` implementation from the refined B specification: this includes the definition of the SQL orders to create the database, the roles, the users, and the execution rights but also the JAVA program corresponding to the operations of the system to build. We choose an aspect programming technique (`AspectJ`) in order to separate the security concerns from the functionalities of the application in such a way that security rules could be added or removed without altering the whole application (see Section 7).
- a tool that makes the presented approach workable is provided (see Section 8).

The rest of this paper is organized as follows: Section 2 gives an overview of the B method and `AspectJ` concepts detailed to the extent needed in this paper and proposes a motivating example. Section 3 shows the modeling of the functional and security requirements using UML-based notations namely, a class diagram, `SecureUML` and activity diagrams. The translation of these diagrams into a B specification is presented in Section 4. The verification and the validation of these specifications are illustrated in Section 5. Next, Section 6 illustrates a refinement process to make the obtained B specification close to a relational-like B implementation. The translation of this relational-like B implementation into an `AspectJ` code is presented in Section 7. Section 8 gives a description of the tool developed to support the proposed approach. A comparison with existing work is then provided in Section 9. The last section concludes and dummyTXdummy- presents some future work.

2. Background

In this section, we provide a broad overview of the main concepts of the B method and `AspectJ` necessary for the comprehension of the present paper. Then, a case study is provided to illustrate the presented approach.

2.1. Overview of the B method

Introduced by J.-R. Abrial [4], B is a formal method dedicated to developing safe systems. B specifications are organized

Please cite this article as: A. Mammar et al., A formal approach to derive an aspect oriented programming-based implementation of a secure access control filter, Information and Software Technology (2017), http://dx.doi.org/10.1016/j.infsof.2017.08.001