PRES LUNAM
Ecole Doctorale STIM
Sciences et Technologies de l'Information et Mathématiques

Spécialité : Informatique
Laboratoire : LINA
Equipe : ÁtlanMod

A Model-driven Approach for the Extraction and Analysis of Access-Control Policies

Martínez, Salvador
Mél : Salvador.MARTINEZ@mines-nantes.fr

Abstract: Several components are required to build up a system security architecture, such as firewalls, database user access control, intrusion detection systems, and VPN (Virtual Private Network) routers. These components must be properly configured to provide an appropriate degree of security to the system. However, the configuration process is highly complex and error-prone. In most organizations, security components are either manually configured based on security administrators expertise and flair; or simply recycled from existing configurations already deployed in other systems (even if they may not be appropriated for the current one). These practices put at risk the security of the whole organization. We propose the construction of a model-driven automatic reverse engineering mechanism capable of analyzing deployed security aspects of components (e.g., concrete firewall configurations) to derive the abstract model (e.g., network security global policy) that is actually enforced over the system. Once the model is obtained, it can be reconciled with the expected security directives, to check its compliance, can be queried to test consistency or used in a process of forward engineering to generate validated security configurations.

Keywords: *Security, Model-driven engineering, Reverse-engineering*
Collaborations : The present work is a collaboration with the team SERES, LUSSI Department of Telecom Bretagne in Rennes.

1 Introduction

Nowadays Informations System (IS) are complex and composed by several heterogeneous components. There, one of the main concerns is security, that comprises several different aspects expressed as properties or requirements the information system must fulfil. One of these requirements is confidentiality[1], meaning that the information system must provide the means to ensure the data is only available to authorized subjects. This confidentiality requirement is often implemented, due to its conceptual simplicity, with respect to other techniques, by using access-control (AC) mechanisms. This way, several of the components of the information system will put in place AC mechanisms to assure their part on the security enforcement is met.

However, AC is diverse. There are several different model proposals and furthermore, the way it is implemented in final components vary largely due to the specificities of the domain. As an example, in firewalls, that filter the traffic of networks with respect to a given policy, AC is generally implemented by using simple AC lists composed by a set of filtering rules. In databases, however, the common AC mechanism is a combination between Discretionary Access-control[2] (DAC) (for delegating permissions) and Role-based Access-control[3] (RBAC) (for simplifying the administration of users). Other IS components like Web Services and web Content Management Systems (CMSs) also present particularities. Moreover, despite a few approaches to derive configurations from high-level specifications, the implementation process in each component is an error-prone task generally done by hand using low-level and often, vendor-specific, mechanisms.

As a consequence, differences between the desired and implemented policy may have been introduced. Moreover, in a dynamic context, policies can vary quite often in order to met new requirements incrementing the risk of introducing errors.

Thus, to assure the system is not under risk, and to facilitate its evolution and analysis, knowing with precision which policy is being enforced by the system turns up as a critical task. However, once implemented, knowing which policy is being enforced is, by cause of all the described complexities, a very complex and time-consuming task.

In order to tackle this problem, and because of its proven effectiveness for the specification and realization of complex systems [4], we propose the construction of a model-driven automatic reverse engineering

mechanism capable of analyzing deployed security aspects of components (e.g., concrete firewall configurations) to derive the abstract model (e.g., network security global policy) that is actually enforced over the system. Once the model is obtained, it can be reconciled with the expected security directives, to check its compliance, can be queried to test consistency or used in a process of forward engineering to generate validated security configurations.

The rest of the paper is organized as follows: Section 2 details the proposed approach. Section 3 discusses related work and finally, Section 4 concludes the paper and present some future work.

2 Approach

In Figure 1 we summarize our proposed approach to deal with the reverse-engineering of access-control policies from heterogeneous information systems. It consists in the following steps:

Extracting abstract models

The first step extracts the AC policy enforced by each of the components of the information system. This information is then represented in a Platform Independent Model (PIM), so that it can be manipulated disregarding the specificities of the concrete technologies used for the implementation. Central to this steps is the development of AC domain specific metamodels, able to represent the AC information of each component in a abstract, concise, easy to analyse and understand manner. Once this metamodels are available, the general process to obtain the platform independent representation for the AC information contained in a given component work as follow.

First of all, the information needs to be translated from the technological domain where the information is, to the modelware technical domain, e.g,. in the case of firewalls, information in the form of textual configuration files, have to be translated into models. For this first step, parsers and injectors are needed. The parser is able to read the source of the information whereas the injector uses the obtained information in order to fill a model with it. As the conceptual distance between the source information and the abstract models is usually important, making the work of injector very complex, the definition of platform-specific models, i.e., models representing the information in the same abstraction level as the source information, are needed. In that case, on a further step, this platform-specific model is transformed, by the means of model transformations tools like ATL[5], to the abstract model.

At this point, the AC control information of each component is represented in abstract deomain-specific AC models. Having the access-control information of a network represented as a model, enables the reutilization of a plethora of well-known, off-the-shelf MDE tools. Editor generators, model transformation engines, etc. become automatically available. An immediate application would be the use of the well-known OCL[6] query language to calculate interesting metrics on the model and perform some advanced queries on the rules represented in it. As an example, we can easily count the number of permissions per *subject* as well as querying if a given *subject* has permissions on a given object.

Model merge

The second step consist in combining these policies coming from the different system, so that useful information can be obtained and properties with respect to a global policy can be checked. We must be able to check if something not allowed in a component is allowed in another component creating a vulnerability in the system. We should be able to check that when a *subject* gets a permission on a given *object*, all components participate properly so that the *object* can be reached by the subject.

In the rest of the section, we will describe how the presented approach is applied on concrete IS components. Concretely, we will evaluate network system and databases. We conclude the section by outlining the requirements for the combination of the AC information of each component into a global representation of the security policy of the IS.

2.1 Network (firewall) Security

Firewalls, designed to filter the traffic of a network with respect to a given number of access-control rules, are key elements in the enforcement of network security policies. However, their configuration files are still mostly manually written, using low-level and, often, vendor-specific rule filtering languages. Moreover, the network topology, that may include several firewalls (potentially from different vendors), may impose

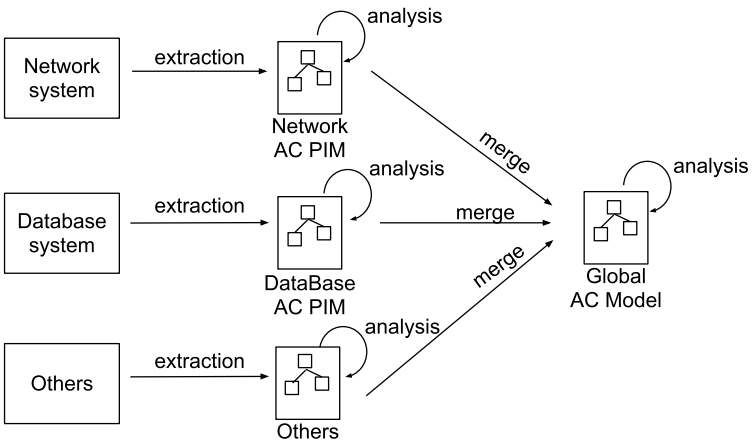

Figure 1: Extraction approach

the necessity of splitting the enforcement of the global security policy among several elements. Due to the complexity of the process, is likely that we end up with differences between the implemented policy and the desired one. We provide an approach to extract the AC information from networks so that its contribution to the security policy of an IS can be easily grasped.

Our approach, first, extracts and abstracts the information of each firewall configuration files to models conforming to a Platform-independent metamodel specially tailored to represent network-access control information in an efficient and concise way. Then, after performing structural verification of the information in the individual models, it combines these models to obtain a centralised view of the security policy of the whole network. Finally, this global network access-control model can be analysed and further processed to derive useful information. As en example, we analyse the structure of its contents to derive the network topology the firewalls operate on.

2.2 Database Security

Databases are at the core of most companies information systems, hosting critical information for the day to day operation of the company. Securing this information is a critical concern and therefore, databases participate in the enforcing of the confidentiality property on an IS.

Several database mechanisms may be needed in the implementation of the policy, e.g., triggers can be used to add fine-grained control on privileges, scattering the policy and increasing the complexity of the implementation process. We believe the understanding and analysis of the security policies at a platform-independent logical level is much easier than a direct exploration of the tables of the database dictionary where the AC information is scattered and whose structure is unfortunately not standardised.

Thus, we provide a means to represent such logical models for security concerns in database systems and we describe a reverse engineering approach that can automatically create this logical model out of an existing database.

2.3 Global Analysis

As introduced before, security properties can be analyzed in different abstraction levels. The solutions used to met a security property in an abstract level will impose some other properties in the more concrete levels. By example, confidentiality is an Abstract-level security property that is often met by using AC mechanisms in concrete systems. Concrete properties take into account solutions for the high level requirements, like AC. As such, these concrete properties take into account the solution and the properties of the system. These properties can be expressed as rules. A rule could say that if a user have a permission to read a document stored in a database table, the system must assure she has the means to do it. This is, there is a path from my machine one can follow until the database and an access into the database is provided.

Our approach aims to support this kind of property checking through the analysis of the combination of the system componentâs access control policies.

This combination process comprises putting all AC models coming from different components in the same repository and if possible, under the same model representation. Then, the elements of the model should be *matched* so that an element in a given AC model may be related to other elements in other AC models. As an example, a *subject* that has permissions in a database would be matched, if existing, with the *subject* logged in a web page or a *host* requiring a service from a server.

3 Related Work

Very few works deal with the reverse engineering of Access-control information and to the best of our knowledge, none of them is focused in extracting and combining the policies of complete Information Systems. Regarding studied concrete components, some approaches have been presented for extracting AC information out of networks[7][8], however, they are mostly focused in verification tasks and do not provide an explicit representation easy to understand and manage but impose the user to learn complex formal query languages to get the information. In Databases the problem has not been tackled whereas the approaches working with CMSs are generally very low level. We have provided two approaches to fill the gap in networks and databases whereas CMSs are left as a future work. With respect to the combination of policies coming from different components in a global representation, languages like XACML[9] provide the means to do it. However, a global analysis of the properties of the combined policy is missing. We intend to fill this gap in future work.

4 Conclusion and Future Work

We have presented an model-driven reverse-engineering approach aimed to the extraction of the security policy enforced by heterogeneous Information Systems. We have shown how by using model-driven techniques the complexity of the analysis and management of the policies of individual components is reduced enabling its efficient analysis and evolution. First steps towards the combination of the component policies have been explored. As a future work, we plan to further study the combination process along with the analysis of the resulting representation through the definition and verification of security properties. Moreover, we plan to extend our work to other individual components so that the obtained global representation is more accurate.

References

[1] Nobukazu Yoshioka, Hironori Washizaki, and Katsuhisa Maruyama. A survey on security patterns. *Progress in Informatics*, 5(5):35–47, 2008.

[2] DoD 5200.28-STD. *Trusted Computer System Evaluation Criteria*. Dod Computer Security Center, December 1985.

[3] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The nist model for role-based access control: towards a unified standard. In *Proceedings of the fifth ACM workshop on Role-based access control*, RBAC '00, pages 47–63, New York, NY, USA, 2000. ACM.

[4] Douglas C Schmidt. Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 39(2):25, 2006.

[5] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. Atl: A model transformation tool. *Science of Computer Programming*, 72(1):31–39, 2008.

[6] Object Management Group. OCL 2.2 Specification, feb 2010.

[7] Yair Bartal, Alain Mayer, Kobbi Nissim, and Avishai Wool. Firmato: A novel firewall management toolkit. *ACM Trans. Comput. Syst.*, 22(4):381–420, November 2004.

[8] Timothy Nelson, Christopher Barratt, Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. The margrave tool for firewall analysis. In *Proceedings of the 24th international conference on Large installation system administration*, LISA'10, pages 1–8. USENIX Association, 2010.

[9] Hal Lockhart, Bill Parducci, and Anne Anderson. OASIS XACML TC, 2013.