A Toolchain for Designing and Testing XACML Policies

Antonia Bertolino*, Marianne Busch†, Said Daoudagh*, Nora Koch† and Francesca Lonetti*, Eda Marchetti*
*Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR
via G. Moruzzi 1, 56124, Pisa, Italy
{firstname.lastname}@isti.cnr.it
†Ludwig-Maximilians-Universität München
Oettingenstraße 67, 80538 Müunchen, Germany
{busch, kochn}@pst.ifi.lmu.de

I. Introduction

Security aspects are critical issues for many application domains such as Service Oriented Architectures (SOAs) and Peer-to-Peer (P2P) systems. Justified confidence in the implemented security mechanisms is a key point for assuring proper data access. In the last years XACML has become the de facto standard for specifying policies for access control decisions in many application domains. Briefly, a XACML policy defines the constraints and conditions that a subject needs to comply with for accessing a resource and doing an action in a given environment. However, due to the complexity of the language, the XACML policy specification is a difficult and error prone process that requires specific knowledge and big effort to be managed correctly.

In recent years, model-driven approaches have been proposed for improving the definition of XACML policies [1], to overcome intrinsic XACML language difficulties. Many methods are able to capture the access control peculiarities by abstracting from the complexity of the language. However, a simplified view could hide some security inaccuracies, due to an inappropriate use of model constructs. These weaknesses can only be tackled by validating the final XACML policy.

Model-driven proposals rarely provide facilities for verifying the compliance of the derived policy with the requirements expressed in the model [2], [3]. In this proposal, we make a step in this direction by presenting an integrated framework not only for designing security policies, but also for testing the compliance of the derived XACML sources with the initial models. Inspired by the conformance testing process we use some XACML-based testing strategies for generating appropriate test cases which are able to test functional aspects, constraints, permissions and prohibitions. The execution of these test cases, generated independently from the security model, provides (partial) input/output traces of the XACML policy execution. These data can be exploited for the construction of an additional model representing the XACML policy behavior, called *traces model*. The compliance of the *traces model* with the associated security model is then assessed against some specific criterion.

II. Using a Tool Chain

The available proposals for verifying the consistency of access control policies with security requirements rely on the definition of specific properties of design (see for instance [2], [3]), which in some case could be very complex or difficult to express. The innovation of our proposal is based on the exploitation of a testing process for outlining the actual policy behavior and the use of well-known model assessment techniques for discovering possible gaps between security and trace models. However, assisting users in design and testing of XACML policies is not possible without tool support. Therefore, we introduce a tool chain which includes the following components:

- *Model-driven Policy Design*: offers a tool for the graphical specification of security requirements and converts the model into a XACML policy;
- *Test Case Generation and Execution*: provides testing strategies to derive test cases and an engine for executing them on the XACML policy;
- *Trace Analysis and Model Compliance*: implements a methodology for analyzing the execution traces and for deriving the *traces model*. An oracle assesses the compliance of the *traces model* with the graphical security model.

Technical details about the "Model-driven Policy design" and "Test Case Generation and Execution" components are provided in Sections II-A and II-B. Currently we are finalizing the "Trace Analysis and Model Compliance" component. The underlining ideas are presented in Section II-C.

A. Model-driven Policy Design

Modeling access control policies on a high level of abstraction has the advantage that policies are easy to understand and to maintain. UML-based Web Engineering (UWE) [4] is a notation for secure web applications which supports this approach. UWE uses the extension mechanisms provided by UML via the definition of a UML profile. UWE proposes to build different models for views such as content and navigation. (1) The Content Model, representing the domain concepts that are relevant for the web application

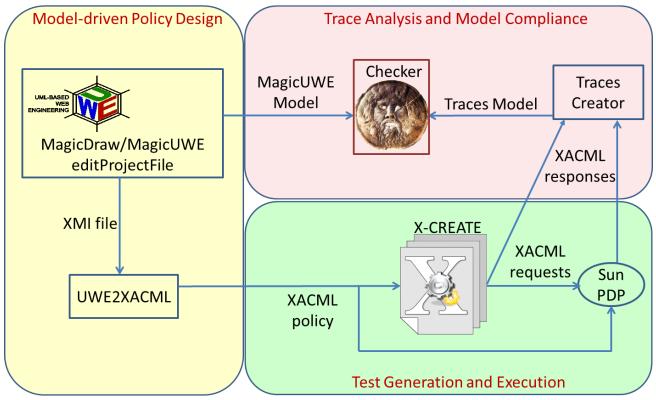

Figure 1. Tool chain

and the relationships between them. (2) The Role Model, defining a hierarchy of user groups with the purpose of authorization and access control. (3) The Basic Rights Model, expressing role based access control on the domain concepts. (4) The Navigation Model, providing a graphical representation of the path the user can navigate in the web system. This model also includes security features as, e.g., authentication and secure connections.

For each view, UWE selects an appropriate type of UML diagram and provides a set of stereotypes, tag definitions and constraints. For instance, the Basic Rights Model is based on a UML class diagram which can be exported as XACML policy file. This file is used for the test case generation in the next step of our tool chain.

B. Test Case Generation and Execution

X-CREATE (XaCml REquests derivAtion for TEsting) is a tool for systematic generation of XACML requests [5]. It takes into account the XACML policy structure, which basically consists of a Target and a set of Rules, specifying the access constraints and conditions. Specifically, X-CREATE implements four testing strategies that are based on a combinatorial approach of the values taken from the Target and Rules of a policy. In addition, random values are considered for negative testing purposes. The first testing strategy (called Simple Combinatorial) uses values combinations for deriving simple requests containing one subject, one resource, one action and one environment. The main advantage of this strategy is that it is simple and achieves the coverage of the policy input domain represented by the policy values combinations. The second testing strategy (called XPT-based) exploits the XACML Context Schema for systematically deriving structurally more complex requests that are suitable where the PDP decision depends simultaneously on the values of more than one subject, resource, action and/or environment. The third strategy (called Incremental XPT) is an improvement of the XPT-based strategy allowing to reduce the number of generated test cases. The last strategy (called Multiple Combinatorial) allows deriving requests having more than one subject, resource, action and environment.

C. Trace Analysis and Model Compliance

The Traces Creator derives the Traces Model by analyzing the XACML requests and the corresponding responses. In particular, for each request it maps: the subjects with the roles of the Role Model; resources with concepts of the Content Model and actions with action stereotypes of the Basic Rights Model. A positive response is translated into a stereotyped dependency between roles and concepts expressed in the request.

The Checker validates the compliance of the Trace Model against the original UWE model by considering domain specific assessment criterions. Therefore, it analyzes if the access permissions for each role are equally defined in both, the Trace Model and the original Basic Rights Model.

III. Conclusions and Future Work

We introduced three components consisting of six tools which can be used as a tool chain as well as separately. Used as a tool chain they allow to semi-automatically test access control policies modeled with UWE.

Future work comprises the full implementation of the "trace analysis and model compliance" component and the evaluation of some further case studies. We plan to use case studies of the NESSoS project, which are dealing with e-health and smart-grids. Additionally, we are working on the specialization of our oracle and think about a feedback loop for the generation of further requests.

Acknowledgment

This work has been partially sponsored by the EU projects NESSoS, NoE 256980 and ASCENS, FP7 257414.

References

[1] OASIS, "eXtensible Access Control Markup Language (XACML) Version 2.0," http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, February 1 Feb 2005.

[2] F. Massacci and N. Zannone, "A model-driven approach for the specification and analysis of access control policies," in *Proc. of the OTM Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE*, 2008, pp. 1087–1103.

[3] A. Pretschner, T. Mouelhi, and Y. Le Traon, "Model-based tests for access control policies," in *Proc. of ICST*, 2008, pp. 338–347.

[4] M. Busch, A. Knapp, and N. Koch, "Modeling Secure Navigation in Web Information Systems," in *BIR*, ser. LNBIP 90. Springer, 2011, pp. 239–253.

[5] A. Bertolino, S. Daoudagh, F. Lonetti, and E. Marchetti, "The X-CREATE framework: a comparison of XACML policy testing strategies," in *Proc. of WEBIST*, 2012, pp. 155–160.