Available online at www.sciencedirect.com
ScienceDirect
journal homepage: www.elsevier.com/locate/cose
Computers & Security

Formal analysis of XACML policies using SMT

Fatih Turkmen [a,*], *Jerry den Hartog* [b], *Silvio Ranise* [c], *Nicola Zannone* [b]

[a] *University of Amsterdam, Amsterdam, Netherlands*
[b] *Eindhoven University of Technology, Eindhoven, Netherlands*
[c] *Fondazione Bruno Kessler, Trento, Italy*

ARTICLE INFO

Article history:
Received 3 October 2016
Received in revised form 4 January 2017
Accepted 24 January 2017
Available online 31 January 2017

Keywords:
XACML policy analysis
Logic encoding of policy analysis problems
First-order logic
Satisfiability modulo theories
Experimental evaluation

ABSTRACT

The eXtensible Access Control Markup Language (XACML) has attracted significant attention from both industry and academia, and has become the de facto standard for the specification of access control policies. However, its XML-based verbose syntax and rich set of constructs make the authoring of XACML policies difficult and error-prone. Several automated tools have been proposed to analyze XACML policies before their actual deployment. However, most of the existing tools either cannot efficiently reason about non-Boolean attributes, which often appear in XACML policies, or restrict the analysis to a small set of properties. This work presents a policy analysis framework for the verification of XACML policies based on SAT modulo theories (SMT). We show how XACML policies can be encoded into SMT formulas, along with a query language able to express a variety of well-known security properties, for policy analysis. By being able to reason over non-Boolean attributes, our SMT-based policy analysis framework allows a fine-grained policy analysis while relieving policy authors of the burden of defining an appropriate level of granularity of the analysis. An evaluation of the framework shows that it is computationally efficient and requires less memory compared to existing approaches.

1. Introduction

Data and other digital resources have become a valuable asset for most organizations. Their protection is thus of utmost importance. Access control is a widely adopted technology for information security and, in particular, to ensure that sensitive information can only be accessed by authorized users.

In the last decades, several access control models and languages have been proposed for the specification and enforcement of access control policies. Among these languages, the eXtensible Access Control Markup Language (XACML) (OASIS XACML Technical Committee, 2013) provides an expressive and extensible syntax in XML for the specification of attribute-based access control policies as well as means to combine policies possibly specified by independent authorities. XACML has been widely used in academia and adopted by many enterprises such as IBM (Buecker et al., 2009), becoming the de facto standard for access control.

However, due to its rich set of constructs and XML-based verbose syntax, policy specification in XACML is known to be difficult and error-prone (Hughes and Bultan, 2008; Nelson et al., 2010). For instance, when a policy is updated to address new requirements, it becomes difficult to determine whether the revised policy works as intended. Even small errors can lead to large data breaches. Ensuring the correctness of access control policies, especially in the error-prone setting of XACML policy specification, is thus a crucial task for protecting sensitive data.

To assist security administrators in the definition of their policies, several methods and tools have been developed for the verification of access control policies at design time using formal

* *Corresponding author.*
E-mail addresses: f.turkmen@uva.nl (F. Turkmen), j.d.hartog@tue.nl (J. den Hartog), ranise@fbk.eu (S. Ranise), n.zannone@tue.nl (N. Zannone).

http://dx.doi.org/10.1016/j.cose.2017.01.009
0167-4048/© 2017 Published by Elsevier Ltd.

reasoning (Backes et al., 2004; Crampton and Morisset, 2012; Hu et al., 2013; Hughes and Bultan, 2008; Nelson et al., 2010; Turkmen et al., 2013). These tools aim to verify whether an access control policy (or a set of policies) satisfies certain properties. A property can vary from checking the (types of) access requests that should be allowed (or denied) by a policy to the analysis of the relation between two policies such as being as permissive/restrictive as another policy (i.e., policy refinement (Backes et al., 2004). However, many of the existing approaches can only analyze a restricted set of security properties due to limits of the expressiveness of the policy formalization used. Moreover, exiting policy analysis tools often do not naturally support reasoning over non-Boolean variables and functions, which often appear in XACML policies. As a consequence, they are not able to analyze access control policies at a fine level of granularity or the performance of the analysis deteriorates very quickly.

To address these issues, in a previous work (Turkmen et al., 2015), we have introduced a framework that employs SAT modulo theories (SMT) (Barrett et al., 2008) as the underlying reasoning method for the formal analysis of XACML policies. SMT is a natural extension to propositional satisfiability (SAT) (Gomes et al., 2008) in which SMT solvers employ tailored reasoners when solving non-Boolean predicates in the input formula. In particular, SMT enables the use of background theories, such as linear arithmetic and equality, to reason about the satisfiability of many-sorted first order formulas. In Turkmen et al. (2015), we provided the intuition of how XACML policies can be encoded into SMT formulas and presented a powerful query language that allows the specification and analysis of a vast range of security properties that have been proposed in the literature. However, given the complex syntax of XACML, it is desirable to have an automated translation of XACML policies into SMT formulas while preserving the semantics of the original policy. Moreover, although SMT provides a powerful approach to problem verification including policy analysis, the problem of checking the satisfiability of arbitrary many-sorted first order logic formulas can be undecidable.

In this paper, we extend the work in Turkmen et al. (2015) by providing the following contributions:

- We provide a complete procedure for the automated translation of XACML policies into SMT formulas for policy analysis. Specifically, we present an encoding of XACML policies that flattens the hierarchical structure of a policy. To support the translation, we provide an encoding of XACML combining algorithms and a mapping between the most common XACML functions and the available SMT background theories.
- We provide a proof of the correctness of the proposed encoding, thus guaranteeing that the semantics of the original policy is preserved.
- We confirm the expressive power of our query language by encoding a new set of properties, namely separation of duty constraints.
- We study under which conditions SMT solvers are capable of tacking policy analysis problems. To the best of our knowledge, this is the first work that provides a detailed study of the complexity of policy analysis in SMT.
- We complement the study of the complexity with an evaluation of the framework through a more extensive set of experiments compared to Turkmen et al. (2015). In particular, we compare our SMT-based approach with SAT-based approaches using different SAT solvers, thus providing a more comprehensive comparison between the two approaches. Moreover, we evaluate our framework using additional realistic policies.

The paper is structured as follows. The next section provides background about XACML and SMT. Section 3 presents our encoding of XACML policies as SMT formulas. Section 4 introduces a query language for the specification of security properties and demonstrates this language by encoding a number of well-known security properties from the literature. Section 5 discusses the complexity of policy analysis in SMT. Section 6 presents an experimental evaluation of our framework. Finally, Section 7 discusses related work, and Section 8 concludes the paper providing directions for future work.

2. Preliminaries

This section introduces the basic notions underlying XACML and SMT.

2.1. XACML

XACML is an OASIS standard for the specification of access control policies. It provides an attribute-based language that allows the specification of composite policies. In this work, we focus on the core specification of XACML v3 (OASIS XACML Technical Committee, 2013) (without obligations) unless it is indicated otherwise.

XACML policies are expressed using three policy elements: *policysets*, *policies* and *rules*. A policyset consists of policysets and policies; policies in turn consist of rules. If a policy element p_1 is nested (i.e., textually contained) in a policy element p_2, we say that p_1 is a *child policy element* of p_2 or, equivalently, that p_2 is the *parent policy element* of p_1. Each policy element has a (possibly empty) *target* which defines (restricts) the applicability of the policy element, i.e. the set of access requests that the policy element applies to. In addition, rules specify an *effect* element that defines whether an access request should be allowed (*Permit*) or denied (*Deny*), and can be associated with a *condition* to further restrict their applicability. A condition is a predicate that must be satisfied for the rule to be applicable.

The evaluation of an access request against a policy element results in an access decision. XACML v3 uses two access decision sets: a four-valued decision set (i.e., `Permit`, `Deny`, `Indeterminate` and `NotApplicable`) and six-valued decision set in which the `Indeterminate` decision is extended to record the decision that would have been returned if an error in the policy evaluation did not occur (the so-called *Indeterminate extended set*). In particular, the Indeterminate extended set consists of decisions `Indeterminate`{`P`}, `Indeterminate`{`D`} and `Indeterminate`{`PD`} where `Indeterminate`{`P`} indicates an `Indeterminate` decision from a policy element which could have evaluated to `Permit` (but not `Deny`), `Indeterminate`{`D`} indicates an `Indeterminate` decision from a policy element which could have evaluated to `Deny` (but not `Permit`) and