POSTER: Analyzing Access Control Policies with SMT

Fatih Turkmen
Eindhoven University of Technology
f.turkmen@tue.nl

Jerry den Hartog
Eindhoven University of Technology
j.d.hartog@tue.nl

Nicola Zannone
Eindhoven University of Technology
n.zannone@tue.nl

ABSTRACT

The flexibility and expressiveness of eXtensible Access Control Markup Language (XACML) allows the specification of a wide range of policies in different access control models. However, XACML policies are often verbose and, thus, prone to errors. Several tools have been developed to assist policy authors for the verification and analysis of policies, but most of them are limited in the types of analysis they can perform. In particular, they are not able to reason about predicates of non-boolean variables and, even if they do, they do it inefficiently. In this paper, we present the $X2S$ framework, a formal framework for the analysis of XACML policies that employs Satisfiability Modulo Theories (SMT) as the underlying reasoning mechanism. The use of SMT not only allows more fine-grained analysis of policies, but it also improves the performance of policy analysis significantly.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*access control, verification*

Keywords

Access control, policy analysis and verification, property checking, SAT modulo theories

1. INTRODUCTION

Modern IT systems usually rely on access control policies to regulate access to sensitive data such as health records or financial information. The separation of access control rules from the application logic allows an isolated view on authorization (i.e., who has access to what), but it requires dedicated languages for the specification of access control policies. XACML is a widely used access control policy language which has attracted considerable attention from both industry and the research community. XACML provides a standard XML-based syntax for policy specification and an architecture for policy enforcement.

However, the flexibility and expressiveness of XACML along with its verbose syntax make policy specification a complex and error-prone task. Policy errors can be dangerous and costly for organizations. On the one hand, they may allow illegitimate access to sensitive data and, thus, lead to data breaches. On the other hand, they may deny access to legitimate users causing service disruption.

In order to assist policy authors in the specification and off-line analysis of access control policies, several techniques and tools have been proposed to verify whether a policy satisfies a certain property using an automated reasoner [1, 3, 5, 6, 7]. These properties vary from expressing certain policy behavior (e.g., checking whether a policy correctly denies a user under certain circumstances) to comparison of policies (e.g., checking whether a policy is as restrictive or permissive as another). Most existing policy analysis tools, however, either are limited in the properties they can analyze, or do not provide sufficient granularity in the analysis due to the limitations of the underlying reasoner.

In this paper, we present the $X2S$ framework, a formal analysis tool for the verification of XACML policies. $X2S$ allows policy authors to verify their policies against a large variety of properties from the literature. In contrast to existing tools, $X2S$ employs propositional satisfiability (SAT) modulo theories (SMT) [2] as the underlying reasoning mechanism. SMT enables the use of background theories, such as linear arithmetic and equality, when reasoning about the satisfiability of first order formulas with variables of different sorts such as integers. By employing SMT, $X2S$ allows reasoning over non-boolean attributes and functions which frequently appear in XACML policies and are usually left uninterpreted [7] in the existing tools. $X2S$ not only enables a more fine-grained analysis of policies, but it is also more efficient than existing SAT-based policy analysis tools. Therefore, $X2S$ provides a practical tool to assist policy authors in analyzing their policies and fixing policy errors before deployment. This minimizes the risk of security incidents due to incorrect policy specifications and thus reduces costs for organizations.

2. THE $X2S$ FRAMEWORK

This section provides an overview of the $X2S$ framework.

2.1 Ingredients

The main goal of the $X2S$ framework is to support the formal analysis of XACML policies. It formally represents

CCS'14, November 3–7, 2014, Scottsdale, Arizona, USA.
ACM 978-1-4503-2957-6/14/11.
http://dx.doi.org/10.1145/2660267.2662366 .

XACML policies and uses this formalization to verify whether the policies satisfy certain security properties. There are three main concepts underlying $X2S$:

- *Formal Model of Policies*: XACML policies are formally represented as many-sorted first order formulas whose variables denote policy attributes. In particular, XACML policies are transformed into policy formulas in a normal form, which define the conditions under which a policy evaluate to a certain access decision. Intuitively, policy formulas consist of set predicates encoding constraints on the applicability of policy elements to requests and policy combining algorithms.
- *Expressive Query Specification Language*: Properties are specified in $X2S$ using a simple yet expressive query language built on top of policy formulas [8]. The query language allows the specification of several security properties discussed in the literature such as policy refinement and attribute hiding as well as the specification of ad-hoc properties to verify whether a policy behaves as expected under certain circumstances.
- *Satisfiability Modulo Theories*: What distinguishes $X2S$ from other policy analysis tools is the use of SMT [2] as underlying reasoning mechanism. SMT allows the verification of the satisfiability of a many-sorted first order formula containing non-boolean variables with respect to background theories such as linear arithmetic or uninterpreted functions with equality. SMT not only enables more expressiveness in policy formulas, but it also solves their satisfiability more efficiently thanks to the use of tailored background theory solvers.

2.2 Overview

Figure 1 presents the architecture of the $X2S$ framework. The framework consists of two main components: *SMT Translator* which encodes XACML policies and properties into SMT specifications and the *Report Generator* which serves as a front-end for the interaction with the SMT solver.

The *SMT Translator* builds an internal model of the XACML policies given as input by extracting the relevant information and policy structure. This model represents the applicability constraints for each policy element (i.e., rules, policies and policy sets) as well as the effect of rules (*Permit* or *Deny*) and the combining algorithm for policies and policy sets. The constraints for a policy element are built from the XACML functions, attributes and their values occurring in its target and conditions. Their evaluation partitions the policy space (all possible requests) into three disjoint subspaces (i.e., *Applicable*, *Indeterminate*, *NotApplicable*).

The internal model is used to generate policy formulas which encode the decision spaces of the policy, i.e. the conditions under which an access decision is returned by the policy evaluation (i.e., *Permit*, *Deny*, *Indeterminate* and *Not Applicable*). Policy formulas in a normal form are generated from the internal model in two steps: First, the applicability constraints of each policy element are recursively propagated to the child policy elements. The propagation results in formulas encoding the applicability of each rule with the applicability of its parents considered. These applicability formulas together with the effect of the rules are combined into policy formulas that encode the decision space of the policy. This combination is performed in a bottom-up fashion according to the semantics of the combining algorithms specified in the policy elements and results in a many-sorted first-order formula for each access decision.

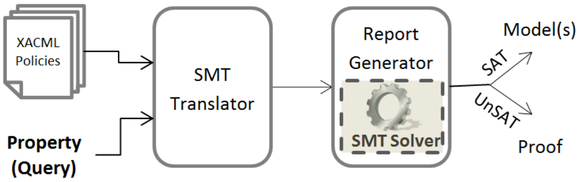


Figure 1: Prototype Architecture

The *SMT Translator* also allows for the specification of properties against which the policies are analyzed. Properties are specified in $X2S$'s query language [8], a first order logical language built over terms representing the decision spaces of policies. For instance, term P_p refers to the policy formula encoding the *Permit* space of policy p. By substituting the terms denoting the decision spaces with their respective formulas, the *SMT Translator* generates a query formula that encodes the property over the selected policies.

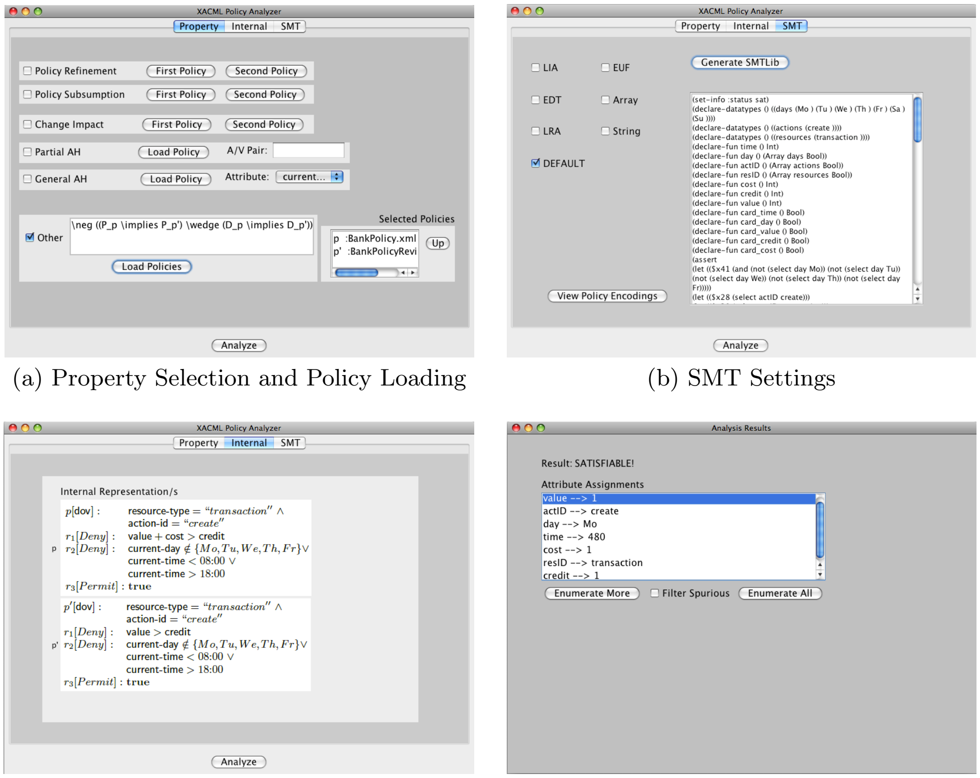
The *Report Generator* is a solver front-end that feeds the query formulas obtained from *SMT Translator* to an SMT solver and presents the analysis results in a user-friendly format. As underlying solver we use Z3 [4], an efficient SMT solver with a rich set of supported background theories. The analysis results can be either variable assignments satisfying the query formula (called models) or a proof of the formula unsatisfiability. For example, for query P_p an access request permitted by p will be generated. If multiple solutions are required by a property (e.g., change-impact), the *Report Generator* enumerates solutions by adding the negation of the found solution to the original formula iteratively. The solver may return a large (possibly infinite) number of solutions which are "uninteresting" for the problem at hand, the so called spurious models. To eliminate spurious models the *Report Generator* fixes the value of variables with unbounded domain (e.g., integers) to the first assignment obtained from the solver.

2.3 Demonstration

We demonstrate the use of $X2S$ by considering a simple access control policy that regulates the creation of a domestic payment transaction in a bank. The bank applies a commission to each transaction. The policy states that the *creation* of a *transaction* is only allowed during week days and within working hours. In addition, a user cannot create a transaction if her balance (credit) is less than the sum of transaction amount (value) and the commission (cost). Due to a change in the law, the bank decides to eliminate the commissions, which should also be reflected in the access control policy. The bank must ensure the updated policy does not introduce errors in the overall authorizations of users.

The policy analyst uses $X2S$ to analyze the policies and chooses the property to check for by either selecting one of the predefined queries encoding properties from the literature or by specifying his own queries (Figure 2a). He chooses to specify a query that checks for cases where the permit and deny decisions of the old policy p are not preserved by the new policy p'.

In order to reason about the non-boolean attributes contained in the policies of our scenario, several background



□ Policy Refinement

□ Policy Subsumption

□ Change Impact

□ Partial AH — A/V Pair:

□ General AH — Attribute: current...

☑ Other: \neg ((P_p \implies P_p') \wedge (D_p \implies D_p'))

Selected Policies:
p :BankPolicy.xml
p' :BankPolicyRevi

(a) Property Selection and Policy Loading

□ LIA □ EUF

□ EDT □ Array

□ LRA □ String

☑ DEFAULT

```
(set-info :status sat)
(declare-datatypes () ((days (Mo ) (Tu ) (We ) (Th ) (Fr ) (Sa )
(Su ))))
(declare-datatypes () ((actions (create ))))
(declare-datatypes () ((resources (transaction ))))
(declare-fun time () Int)
(declare-fun day () (Array days Bool))
(declare-fun actID () (Array actions Bool))
(declare-fun resID () (Array resources Bool))
(declare-fun cost () Int)
(declare-fun credit () Int)
(declare-fun value () Int)
(declare-fun card_time () Bool)
(declare-fun card_day () Bool)
(declare-fun card_value () Bool)
(declare-fun card_credit () Bool)
(declare-fun card_cost () Bool)
(assert
(let (($x41 (and (not (select day Mo)) (not (select day Tu))
(not (select day We)) (not (select day Th)) (not (select day
Fr)))))
(let (($x28 (select actID create)))
```

(b) SMT Settings

Internal Representation/s

p
p[dov] : resource-type = "$transaction$" ∧
action-id = "$create$"
$r_1[Deny]$: value + cost > credit
$r_2[Deny]$: current-day $\notin \{Mo, Tu, We, Th, Fr\} \vee$
current-time $<$ 08:00 ∨
current-time $>$ 18:00
$r_3[Permit]$: **true**

p'
p'[dov] : resource-type = "$transaction$" ∧
action-id = "$create$"
$r_1[Deny]$: value > credit
$r_2[Deny]$: current-day $\notin \{Mo, Tu, We, Th, Fr\} \vee$
current-time $<$ 08:00 ∨
current-time $>$ 18:00
$r_3[Permit]$: **true**

(c) Internal Policy Representation

Result: SATISFIABLE!

Attribute Assignments

```
value --> 1
actID --> create
day --> Mo
time --> 480
cost --> 1
resID --> transaction
credit --> 1
```

□ Filter Spurious

(d) Policy Analysis Results

Figure 2: The *X2S* Framework

theories including linear arithmetic (LIA) and enumerated data types (EDT) are needed. By default (Figure 2b) the solver automatically chooses which background theories to use based on the XACML functions used in the policy. However, the analyst can manually override this, for example, to improve performance when certain theories are not really needed. The analyst can now choose to examine the internal model of the policies (Figure 2c) and/or view the analysis results. As the query is satisfiable here, *X2S* shows a sample request, extracted from the solver output, for which the two policies produce different decisions (Figure 2d). The analyst can also obtain additional requests for deeper analysis.

3. CONCLUSIONS

We have presented *X2S*, a XACML policy analysis framework that allows checking of policies before they are deployed. The framework converts policy and properties to SMT formulas and employs off-the-shelf SMT solvers to verify them. In contrast to available tools, it supports reasoning about non-boolean attributes of policies increasing the depth and efficiency of the analysis. Analysis of a large range of properties from the literature such as policy refinement or attribute-hiding is supported. We demonstrated the usage of *X2S* with a user-defined query over an example policy that contains linear arithmetic expressions. The internal policy representation provides a compact overview of the policies, and the intuitive query language allows easy expression of policy properties to be checked. Thus, *X2S* can bring large savings by enabling essential sanity checks on policies which help in preventing costly mistakes.

Acknowledgments This work has been partially funded by the EDA project IN4STARS2.0, the ITEA2 project FedSS, and the ARTEMIS project ACCUS, the NWO CyberSecurity programme under the PriCE project, and the Dutch national program COMMIT under the THeCS project.

4. REFERENCES

[1] M. Backes, G. Karjoth, W. Bagga, and M. Schunter. Efficient comparison of enterprise privacy policies. In *SAC*, pages 375–382. ACM, 2004.

[2] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, pages 825–885. IOS Press, 2008.

[3] J. Crampton and C. Morisset. PTaCL: A Language for Attribute-Based Access Control in Open Systems. In *POST*, pages 390–409. Springer, 2012.

[4] L. M. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *TACAS*, pages 337–340. Springer, 2008.

[5] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and change-impact analysis of access-control policies. In *ICSE*, pages 196–205. ACM, 2005.

[6] W. Fitzgerald, F. Turkmen, S. Foley, and B. O'Sullivan. Anomaly analysis for physical access control security configuration. In *CRiSIS*, pages 1–8, 2012.

[7] G. Hughes and T. Bultan. Automated verification of access control policies using a SAT solver. *STTT*, 10(6):503–520, 2008.

[8] F. Turkmen, J. den Hartog, and N. Zannone. Analysis of XACML Policies with SMT, 2014. In Submission.