Available online at www.sciencedirect.com

ScienceDirect

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

Computers & Security

Decision Diagrams for XACML Policy Evaluation and Management

Canh Ngo[*], *Yuri Demchenko*, *Cees de Laat*

Informatics Institute, University of Amsterdam, Science Park 904, 1098XH Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 27 January 2014
Received in revised form
28 September 2014
Accepted 10 November 2014
Available online 21 November 2014

Keywords:
Access control
Authorization
Policy evaluation
Decision diagram
Interval partition processing
XACML

ABSTRACT

One of the primary challenges to apply the XACML access control policy language in applications is the performance problem of policy evaluation engines, particularly when they experience a great number of policies. Some existing works attempted to solve this problem, but only for some particular use-cases: either supporting simple policies with equality comparisons or predefined attribute values. Due to the lack of carefully checking the XACML model, they did not have original policy evaluation semantics. Therefore, they cannot handle errors containing indeterminate decisions, or ignore the critical attribute setting that leads to potential missing attribute attacks. In this paper, we build up the XACML logical model and propose a decision diagram approach using the data interval partition aggregation. It can parse and transform complex logical expressions in policies into decision tree structures, which efficiently improve the policy evaluation performance. Our approach can also be applied to solve other policy management problems such as policy redundancy detection, policy testings and comparisons, or authorization reverse queries.

1. Introduction

XACML (Extensible Access Control Mark-up Language) is an authorization policy language in XML format based on the Attribute-Based Access Control (ABAC) model. It composes policies from set of attribute criteria joined by logical operators to decide if authorization requests are granted. XACML is scalable in arranging policies in the hierarchical order in the repository. The policy language also supports delegations, obligations and advices, that makes it applicable in many areas such as networking, grids, clouds, enterprise organization and management. However, expansions of policies to address system scales will increase the complexity of the repository, which drops the policies evaluation performance.

XACML policies has complex structures containing a sophisticated logical model as follows:

- Policies are organized hierarchically in a policy-tree with rules, policies and policy-sets elements. The tree contains internal nodes and external nodes. An internal node can either be a policyset or a policy. Children of a policyset node can be other policysets or policies. Children of a policy are rules, which are external nodes. Because children can produce conflicting decisions, parent nodes can resolve them by predefined combining algorithms.
- Policy decisions are not only *permit* and *deny*, but also other intermediate values to handle error and un-matched situations such as *not-applicable*, *indeterminate* decisions (see Section 3). It means that operations on combining policies'

* *Corresponding author.*
E-mail addresses: t.c.ngo@uva.nl (C. Ngo), y.demchenko@uva.nl (Y. Demchenko), delaat@uva.nl (C. de Laat).
http://dx.doi.org/10.1016/j.cose.2014.11.003

decisions cannot be derived from binary logical operators. They should be defined in multi-valued logical domains.
- Not all attributes are processed equally, some of them are marked as critical (with the flag *"MustBePresent = true"*): during the evaluation, the missing of these attributes should yield *indeterminate* values rather than the *not-applicable*.
- Because policies have their own predicates to match with requests, attribute comparisons are scattered in the policy-tree. Thus, typical implementations discussed in Turkmen and Crispo (2008) often have redundancies in evaluations: an attribute may be compared multiple times in different policy nodes.

With these characteristics, there are challenges to propose high performance policy evaluation solutions or resolve policy analysis and management problems. We need practical mechanisms that not only can gather predicates and efficiently reduce them in aware of combining algorithms, but also guarantee multi-valued logical semantics of the XACML.

Motivated by the need of the high performance policy evaluation in designing access control systems for Clouds using XACML (Ngo et al., 2011) and related work on decision diagrams (Bryant, 1986; Strehl and Thiele, 2000; Christiansen and Fleury, 2004), policy evaluation approaches (Liu et al., 2011; Pina Ros et al., 2012) as well as implementations reviewed in Turkmen and Crispo (2008), we analyze the logic behind XACML standard and propose the Multi-data-type Interval Decision Diagram (MIDD) approach. Its data structures and operators can transform policies into decision trees which boosts the policy evaluation performance while they keep original semantics. We presented a preliminary work in Ngo et al. (2013) containing the basic formulation of MIDDs. However it still contains some drawbacks in logical analysis and related algorithms. In this paper, we improve and fix as follows:

- Analyze the logic of XACML components evaluation, which essentially is a many-valued logic system with equivalent operators on different domains. The related work analyzed for simple cases with partial error handling coverage (e.g. target expressions returned only either 'matched' or 'unmatched' without errors). Our algorithms in this paper are improved based on formulations from a complete logical analysis of XACML components. Section 3 will clarify this contribution.
- Based on the logical analysis, we classify the MIDD definition in the preliminary work (Ngo et al., 2013) into the MIDD for expressions having the V_M as the target domain, and the X-MIDD for expressions having the target domain V_R. They facilitate improvements of algorithms in Section 5.3.
- Algorithms in Section 6 support the flexible critical attribute settings in different match expressions. This is a drawback of Ngo et al. (2013).

The proposed mechanisms can also be applied to solve XACML policy management problems: transforming a complex policy tree into a unified decision tree which can facilitate to solve policy management problems, such as policy comparisons, policy redundancy detection, policy testings or authorization reverse queries.

The rest of the paper is organized as follows. Section 2 reviews the related work on policy analysis, management, integration and high performance evaluation. Section 3 analyzes XACML logic that provides the basis for the proposed solution. Section 4 formulates the approach to evaluate the complete logical expressions using interval decision diagrams. Section 5 defines fundamental operations to process intervals, partitions and decision diagrams. These materials are used in our solution to transform XACML policies in Section 6. The proposed mechanism is then analyzed and validated in Section 7. Finally Section 8 concludes our paper.

2. Related work

There are numerous prior works on access control policies that mainly focus on policy verification, analysis and testing to detect and remove redundancy (Fisler et al., 2005; Li and Tripunitara, 2006; Kolovski et al., 2007; Hu and Ahn, 2008). Fisler et al. (2005) used propositional logic in XACML to identify properties of given policies and analyze the change-impact of two policies to summarize their differences. The proposal was implemented in the Margrave project, using Multi-Terminal Binary Decision Diagram (MTBDD) (Fujita et al., 1997) as the underlying mechanism. Because of using one binary variable for each attribute–value pair, their approach is only applicable for policies containing all predefined attribute values. Due to modeling decisions as binary values, the MTBDD in Fisler et al. (2005) could not process intermediate decisions, thus ignored error use-cases handling in XACML evaluations. Li and Tripunitara (2006) and Hu and Ahn (2008) proposed methodologies to verify and correct policies under the Role-Based Access Control (RBAC) model. Kolovski et al. (2007) used description logic (DL) to represent XACML policies and use DL reasoners for analysis tasks such as policy comparisons, verification and querying. However, because description logic could only covers a subset of XACML, this approach did not handle complex comparisons, *indeterminate* decisions handling as well as left out *one-applicable* combining algorithm. Masi et al. (2012) formalized the XACML 2.0 semantics and proposed an alternative syntax supporting policy composition. They implemented a tool to compile policies into Java classes following the proposed semantic rules, where these classed are executed to compute policy decisions.

Policy integration and composition were introduced firstly by Bonatti et al. (2002). They defined an algebra with constraints to compose and translate policies into logic programs. However the algebra did not bind with any practical policy language. Mazzoleni et al. (2006) proposed the policy integration preferences which is an XACML extension specified how to integrate policies from different parties. However they did not show any applicable mechanisms for such integration. Bruns et al. (2007) attempted to use Belnap logic to formalize XACML 2.0, in which they mapped four logic values to XACML policy decisions. Actually the logic of XACML is different from Belnap logic because the *indeterminate* values cannot map to any Belnap logical value. Subsequently, Ni et al. (2009) used