Journal of Network and Computer Applications 80 (2017) 45–57

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Elimination of policy conflict to improve the PDP evaluation performance

Fan Deng[a,*], Li-Yong Zhang[b]

[a] School of Computer Science and Technology, Xi'an University of Science and Technology, Xi'an 710054, China
[b] School of Software, Xidian University, Xi'an 710071, China

ARTICLE INFO

Keywords:
XACML
Authorization
Access control
Policy conflict
Policy Decision Point (PDP)
Evaluation performance

ABSTRACT

In the authorization access control model, the Policy Decision Point (PDP) may make an inappropriate authorization decision or the operating efficiency of the network and information system may be influenced, because there might be conflicts in the policies loaded on the PDP. As a result, the PDP's evaluation performance is affected when it evaluates access requests. In order to detect and eliminate conflicts in a policy and achieve the goal that the PDP can evaluate access requests with high efficiency, a *form conflict* detecting and eliminating engine is presented. This engine can not only detect and eliminate *form conflicts* in a policy, but also evaluate access requests. In the *form conflict* detecting and eliminating engine, a *Resource Index Tree* is constructed to convert the rules in a policy defined by the XACML to the node information in the *Resource Index Tree*. On the basis of the dependent relationship of resources, the overlapping relationship of conditions and effect information, *form conflicts* in a policy are detected and eliminated. Experiments make comparisons of the evaluation performance of the *form conflict* detecting and eliminating engine with that of the Sun PDP, as well as XEngine and SBA-XACML. Experimental results show that the evaluation performance of the PDP can be greatly improved by eliminating *form conflicts* in the policies.

1. Introduction

In recent years, emerging network application techniques like SOA (Service-Oriented Architecture), Web Service and Cloud Computing are developing rapidly (Ching, 2013). However, with the continuous progress and development of new techniques, more and more security problems have been exposed. For example, a user without authorization can access system resources by some means and insert fraudulent data into the normal transmission of data. Access control has become the most important protection mechanism in network security and information security at the present stage, so that these security risks can be eliminated effectively (Ni et al., 2012).

In order to meet the needs of managing a large-scale information system, management based on the policy has become the main method chosen by organizations so that they could apply access control to large-scale internal networks and distributed systems (Gwan-Hwan et al., 2012). And the XACML (eXtensible Access Control Markup Language) is one of the widely accepted markup languages which can better describe policies (Bertolino et al., 2013). Policy can be defined as a group of descriptive principles implemented on the basis of practical requirements. These principles are the constraints authorizing the system to make decisions. The XACML is widely used in the service-oriented architecture environment of a distributed application system so that the security policy could be better described (Malek et al., 2012). With the rapid progress and development of information techniques, information systems tend to be diverse. The open system application environment and irregular addition and deletion of policies may lead to the threat that there might be conflicts in the security policies (Turkmen et al., 2013). The system may make an authorization decision inconsistent with the security requirement because of this threat. And systems based on policy management are confronted with a huge challenge (Ramli et al., 2014).

The XACML, which is an open-standard language based on the XML (eXtensible Markup Language), is used to standardize access control implemented in the XML (Chen et al., 2013). In brief, the XACML is a standard description of access requests and its characteristics are as follows (Gil and Artz, 2011):

- *Standard*: The XACML is a standard language that describes policies and can be widely applied to practical requirements in most applications.
- *Universality*: The XACML not only can be applied to specific environment, resources and application systems, but also has preferable universality. This universality is mainly reflected in the policy file which not only can be applied to different application systems, but also is easy for the administrator to enact and manage.

* Corresponding author.
E-mail address: deng_enya@126.com (F. Deng).

http://dx.doi.org/10.1016/j.jnca.2016.12.001
Received 1 April 2016; Received in revised form 3 November 2016; Accepted 2 December 2016
Available online 05 December 2016

Completed access control policies are universally applicable.

- *Supporting the application of the distributed system*: The XACML permits different administrators to enact policies on different occasions asynchronously.
- *Extensibility*: The XACML has a preferable support for multiple data types, functions and policies/rules combining algorithms. And it concentrates on the extension of some new fields so that it can be adapted to the need of developing special application systems.

Although the XACML has great adaptability, compatibility and expressive ability (Dan et al., 2013), its core specification thinks that all policies can be trusted (Adela and Richard, 2013) and lack the ability to self-analyze policies (Lunardelli et al., Mori). Therefore, the XACML cannot effectively detect conflicts in a policy. Next, the scale of the policy set grows fast with the development of access control techniques. In the same XACML policy set, the probability that rules in the same policy or rules between different policies contradict each other will increase, because the same policy set may be enacted and modified by different administrators. As a result, the same policy set might generate different process results for the same access request (Fatema and Chadwick, 2014). Again, if there are conflicts in a policy or between different policies, the following problems may occur:

- It permits unauthorized access (Sainan and Yu, 2013).
- It denies legal access (Wang et al., 2010).
- The PDP which loads the policy file will spend more substantial time evaluating access requests.

Therefore, the core and difficulty in this research is to find an efficient way to detect and eliminate conflicts in a policy and between policies accurately and then enhance the evaluation performance of the PDP.

This paper makes the following contributions.

- A *form conflict* detecting and eliminating engine is presented, which not only can detect and eliminate *form conflicts* in a policy and between policies, but also has the same ability as the PDP. This engine can evaluate the access requests by loading the policy whose *form conflicts* have been eliminated.
- A *Resource Index Tree* is constructed to convert the rules in a policy defined by the XACML to the node information in the *Resource Index Tree*.
- On the basis of the *Resource Index Tree,* a *common resource conflict* detecting and eliminating algorithm and a *dependent resource conflict* detecting and eliminating algorithm are proposed. It only compares a rule with the rule with which it is likely to conflict, which could avoid a large number of unnecessary comparisons with irrelative rules and improve complex and inefficient traversal conflict detecting. Thus, it can greatly improve the efficiency of detecting and eliminating conflicts. The elimination of policy conflicts could effectively enhance the evaluation performance of the PDP.
- The influence of the condition attribute in a policy's target attribute is also considered in this paper, which could avoid false detection results caused by ignoring the condition interval, and improve the correctness of conflict detecting.

The remainder of this paper is organized as follows. Section 2 reviews the related work on policy analysis, management, and high performance evaluation. Section 3 introduces definitions used in the study of *form conflict* detection and elimination. In Section 4, the *form conflict* detecting and eliminating engine is introduced. This engine can detect and eliminate *form conflicts* in a policy and between policies. In Section 5, we propose a *Resource Index Tree* in the *form conflict* detecting and eliminating engine to detect and eliminate *form conflicts*. Section 6 focuses on the method for detecting and eliminating *form conflicts*. A relevant detection and elimination algorithm for *form conflicts* is given in Section 7. Section 8 makes a comparison in evaluation performance between the *form conflict* detecting and eliminating engine and the Sun PDP, as well as XEngine and SBA-XACML. Finally, Section 9 presents some conclusions and directions for our future work.

2. Related work

In recent years, the research aiming at the methods for detecting and eliminating policy conflicts has increased a lot. Scholars around the world have extensively studied this problem (Barron et al., 2011; Fan et al., 2011; Mohan et al., 2011; Ngo et al., 2013; Mourad and Jebbaoui, 2015; Pina et al., 2012; Jebbaoui et al., 2015).

Barron et al. (2011) analyzed and discussed the classification and the consequences of conflict, and introduced a mechanism with which the administrator could take the initiative to detect the conflict related to the subject, resource and action attribute in the XACML policy. The mechanism simulates a set of distributed policies based on authorization and the XACML access control system, and conflicts in the policy could be recognized by the XACML conflict detection algorithm proposed in this paper.

Fan et al. (2011) indicated that conflicts between policies would cause delay or error when distributed systems exchange a large scale of data. To avoid this problem, they proposed a method of fine-grained access control and adopted the semantics-based method to detect conflicts in a policy.

Mohan et al. (2011) indicated that the XACML lacks the ability to analyze and classify conflicts in a policy. They analyzed and classified conflicts caused by operation on the attribute layer and proposed conflict detection methods based on different attributes and mechanisms.

Ngo et al. (2013) defined interval processing and MIDD operations to efficiently and precisely evaluate the complex logical expressions. They also introduce MIDD and X-MIDD, the new decision diagram data structures for XACML evaluation.

Mourad and Jebbaoui (2015) presented a semantics-based policy evaluation that provides better performance compared to the industrial standard Sun PDP and its corresponding ameliorations.

Pina et al. (2012) proposed an optimization for XACML policies evaluation based on two tree structures, which are created for a fast searching of applicable rules and are used for the evaluation of the applicable rules.

Jebbaoui et al. (2015) presented a semantics-based approach for detecting flaws, conflicts and redundancies in XACML policies by offering new mechanisms to analyze the meaning of policy rules through semantics verification by inference rule structure and deductive logic.

As we can see, the above research mainly analyzed the composition of the XACML or policy and proposed formal methods based on descriptive logic aiming at common problems in the XACML policy such as semantic representation, and conflict detection. Analysis results show that these methods can effectively extend the XACML policy and enhance the reasoning and expression ability of the XACML. However, most of these methods traverse every rule in the policy set when detecting conflicts, which greatly reduces the efficiency of conflict detection. Therefore, the key issue in optimizing the conflict detection algorithm is how to effectively reduce the frequency of traversing policy centralized rules. A *Resource Index Tree* is constructed in this paper and it only compares a rule with the possibly conflictive rules, which could effectively avoid the comparison with irrelevant rules and greatly improve the efficiency of conflict detection. On the other hand, if there are lots of conflicts in a policy file containing a large number of rules, the PDP which loads the policy file will spend much more time evaluating the access requests and its evaluation efficiency will drop significantly. Hence, an efficient conflict detection and elimination