Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 610278, 14 pages
http://dx.doi.org/10.1155/2014/610278

Research Article

Policy Decomposition for Evaluation Performance Improvement of PDP

Fan Deng,[1,2] Ping Chen,[1,2] Li-Yong Zhang,[2] Xian-Qing Wang,[2] Sun-De Li,[1] and Hui Xu[1]

[1] *School of Computer Science and Technology, Xidian University, Xi'an 710071, China*
[2] *School of Software, Xidian University, Xi'an 710071, China*

Correspondence should be addressed to Fan Deng; dengfan916@gmail.com

Received 30 December 2013; Accepted 10 April 2014; Published 7 May 2014

Academic Editor: Manyu Xiao

In conventional centralized authorization models, the evaluation performance of policy decision point (PDP) decreases obviously with the growing numbers of rules embodied in a policy. Aiming to improve the evaluation performance of PDP, a distributed policy evaluation engine called XDPEE is presented. In this engine, the unicity of PDP in the centralized authorization model is changed by increasing the number of PDPs. A policy should be decomposed into multiple subpolicies each with fewer rules by using a decomposition method, which can have the advantage of balancing the cost of subpolicies deployed to each PDP. Policy decomposition is the key problem of the evaluation performance improvement of PDPs. A greedy algorithm with $O(nlgn)$ time complexity for policy decomposition is constructed. In experiments, the policy of the LMS, VMS, and ASMS in real applications is decomposed separately into multiple subpolicies based on the greedy algorithm. Policy decomposition guarantees that the cost of subpolicies deployed to each PDP is equal or approximately equal. Experimental results show that (1) the method of policy decomposition improves the evaluation performance of PDPs effectively and that (2) the evaluation time of PDPs reduces with the growing numbers of PDPs.

1. Introduction

In the service-oriented architecture (SOA) [1–5] environment, access control [6–9] is one significant part of security requirement [10–15]. Confronted with the requests of numerous users, all issued resources are required for access control protection by some predominant techniques, such as identity authentication [16–20] and dynamic authorization [21–25].

In conventional centralized authorization models, an XACML (eXtensible Access Control Markup Language) [26–28] policy evaluation engine contains one single *policy decision point* (PDP) that is responsible for granting/denying the access requests of users. The PDP needs to load a policy set which contains a large number of policies, with each policy consisting of a large number of rules. The evaluation performance of PDP will decrease significantly when the number of rules in a policy increases considerably [29].

Meanwhile, when users try to access resources concurrently, the *policy enforcement point* (PEP) calls PDP to retrieve an authorization decision. This authorization decision is made through the evaluation of rules in the policy, which is loaded in PDP. Finally, PEP receives the authorization decision (permit/deny). If the number of users who are originating requests concurrently is very large, the time spent on accessing resources will grow evidently. It is because the later users have to wait for the former users before their authorization operations are completed.

The following bottlenecks will occur in the evaluation performance improvement of PDP.

(i) The number of rules in a policy and access requests in a concurrent operation are extremely large.

(ii) All the access requests have to be sent to one single PDP.

(iii) When evaluating an access request, PDP needs to search which rule is applicable among all the rules in a policy.

Recently, researches on the evaluation performance improvement of PDP can be categorized into two groups. (1) The first uses distributed authorization models to improve the evaluation performance of PDP. According to the attributes contained within the targets in a policy or a rule, Kateb et al. [29] decomposed the global policy into several subpolicies. Alzahrani et al. [30] proposed an XACML distributed authorization model. In this model, the global policy in the centralized authorization model was decomposed into several subpolicies, which were deployed to corresponding PDPs. In this way, different PDPs can cooperate with each other. Decat et al. [31] proposed to decompose and distribute the tenant policies across provider and tenant in order to evaluate as much parts of the policy near the data they required while keeping the tenant access control data confidential. Craven et al. [32] described a method for policy refinement, in which policy decomposition is the first stage. They achieved policy decomposition by the application of decomposition rules, relating domain elements represented more abstractly to the components and implementations of those elements at a more concrete level. (2)The second group uses new techniques to improve the evaluation performance of PDP, such as adopting advanced data structures [33] and founding indices or caches based on access records [34, 35]. Liu et al. [33] presented an XEngine, in which the rules embodied in a policy were expressed as integers before requests were evaluated. XEngine transformed the hierarchical tree structure of the XACML policy to a flat structure so that the time of evaluating requests can be effectively reduced. According to the access records of users, Marouf et al. [34] proposed that the relative order of policies or rules was adjusted in order to speed up the time of evaluating requests. Wang et al. [35] presented an XACML policy evaluation engine termed MLOBEE (multilevel optimization based evaluation engine), in which a multilevel optimization technology was adopted. Before evaluating requests, MLOBEE simplified the rules embodied in a policy and reduced the size of policies. In evaluation procedure, MLOBEE adopted a variety of cache mechanisms, such as evaluation result caches, attribute caches, and policy caches. This method can decrease the communication cost between PDP and other modules.

However, the existing studies scarcely consider the internal structure of a policy in decomposition. The cost of subpolicies deployed to each PDP cannot be guaranteed to be equal or approximately equal. This problem may lead to the fact that there might exist an appreciable difference in the evaluation time among PDPs. For example, some subpolicies might be relatively large and some relatively small, which might affect the evaluation performance improvement of PDPs. Therefore, we present a novel distributed policy evaluation engine and propose a decomposition method. In this method, a policy should be decomposed into multiple subpolicies each with fewer rules so that the cost of subpolicies deployed to each PDP is equal or approximately equal. Policy decomposition is the key problem of improving the evaluation performance of PDPs.

This paper makes the following contributions.

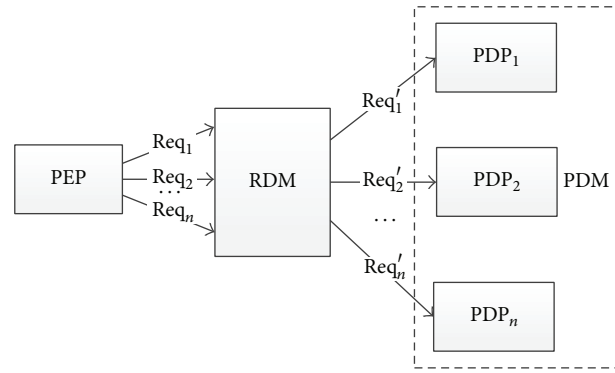


Figure 1: XDPEE with PDM and RDM.

(i) A novel distributed policy evaluation engine (called XDPEE) is proposed, which has abilities of decomposing policies and distributing requests.

(ii) A discrete optimization model of policy decomposition is presented, whose properties are analyzed.

(iii) A greedy algorithm with a favorable time complexity for solving the optimization model is constructed.

(iv) Comparisons of the evaluation performance of PDPs in XDPEE with that of PDPs in the Sun PDP are made. Also, the evaluation time of XDPEE with different numbers of PDPs is measured. Experimental results show that the method of policy decomposition improves the evaluation performance of PDPs substantially.

The remainder of this paper is organized as follows. Section 2 describes a novel distributed policy evaluation engine. A discrete optimization model of policy decomposition is shown and its properties are analyzed in Section 3. In Section 4, we construct a greedy algorithm for solving the optimization model. Section 5 shows experimental results of the evaluation performance improvement of PDPs. Finally, Section 6 presents some conclusions and directions for our future work.

2. Distributed Policy Evaluation Engine

Our proposed distributed policy evaluation engine termed XDPEE is shown in Figure 1, where a *policy decomposition module (PDM)* and a *request distribution module (RDM)* are introduced on the basis of conventional centralized authorization models. Multiple PDPs are founded in order to cooperate with PDM when running. PEP issues access requests to the RDM, which can transmit these requests to the corresponding PDPs according to the information of access requests. If a policy decision process fails, the RDM can retransmit the request to the backup PDP for authorization in order to ensure the robustness of the authorization system.

Two schemes for policy decomposition are as follows.

(i) Every PDP loads the same policy set, and the RDM distributes access requests to the idle PDP for processing.

Table 1: Criteria for policy decomposition.

Category	Combination
1	<Subject>, <Action>, <Resource>
2	<Subject, Action>, <Subject, Resource>, <Action, Resource>
3	<Subject, Action, Resource>

(ii) According to its internal structure, a policy (handled by single PDP) is decomposed into multiple corresponding subpolicies (handled by multiple PDPs), each of which contains fewer rules than the original policy. These subpolicies are deployed to PDPs, each of which loads fewer subpolicies.

The second scheme is adopted for better improving the evaluation performance of PDPs.

3. Policy Decomposition

In this section, we will discuss policy decomposition in detail. First of all, the decomposition criteria based on attributes are addressed. Secondly, a discrete optimization model of policy decomposition is presented. Finally, the properties of policy decomposition are analyzed.

3.1. Decomposition Criteria. A policy is configured by specific subjects, actions, and resources, so the number of rules in a policy with the same subjects, actions, and resources may be extremely large. Therefore, the basis of policy decomposition can be the combination of these three attributes. For example, if a policy is decomposed on the basis of the subject attribute in the target element, the rules with the same subject attribute will be distributed to the identical policy.

A policy can be decomposed by the decomposition criteria based on the combination of three attributes: subject, action, and resource [29], as shown in Table 1. The decomposition criteria do not alter policy behaviors of the centralized architectures. For category 1, the basis of decomposition is each of the three attributes separately. For category 2, the basis is any two of the three attributes, and, for category 3, the basis is all the three attributes. We adopt the first category to decompose a policy according to the subject attribute, for the reason that the rules configured for the same user can be distributed to the identical policy. Therefore, the rules which are corresponding to the user (subject) can be obtained efficiently when requests are evaluated.

If a policy is not properly decomposed, the evaluation performance cannot be improved substantially. In this situation, some subpolicies might contain many more rules than other subpolicies. Suppose that a policy is decomposed on the basis of subject and that each subject may be accessed with the same probability, and how to efficiently decompose a policy and make each PDP spend approximately equal time on evaluating requests becomes a key problem.

3.2. Optimization Model. If a policy is decomposed on the basis of subject, both the number of rules and the cost of evaluation corresponding to each subject embodied in a policy are determined. Suppose that the rule set corresponding to the attribute Subject_i embodied in a policy is $\{r_{i1}, r_{i2}, \ldots, r_{im}\}$ and that the cost set corresponding to the rules is $\{c_{i1}, c_{i2}, \ldots c_{im}\}$; the total cost of the attribute Subject_i is shown in

$$\text{Cost}_i = \sum_{j=1}^{m} c_{ij}. \tag{1}$$

If a policy contains n subjects, the cost set corresponding to these attributes is $\text{Cost} = \{\text{Cost}_1, \text{Cost}_2, \ldots, \text{Cost}_n\}$. If there are k PDPs, then we will need k-partition for Cost and require that the cost of the k subsets after decomposition be equal or approximately equal. For example, if 10 subjects are given, the cost set corresponding to these attributes is $\text{Cost}' = \{5, 10, 6, 8, 7, 7, 9, 7, 5, 6\}$. If there are two PDPs, we will need 2 partitions for Cost' and desire that the cost (or sum value) of each subset after decomposition be equal or approximately equal. The optimal result of decomposition should be $\text{Cost}'_1 = \{5, 6, 7, 8, 9\}$ and $\text{Cost}'_2 = \{5, 6, 7, 7, 10\}$.

The core idea of policy decomposition is that a policy is decomposed into multiple subpolicies according to one of the three categories of decomposition criteria discussed above and that these subpolicies are then deployed, respectively, to several PDPs. We aim at making the cost of subpolicies deployed to each PDP equal or approximately equal; thus, the average evaluation time of these PDPs can be effectively shortened.

The problem of policy decomposition can be summarized as an optimization model as follows. We suppose that a policy is decomposed for k-partition on the basis of subject and that the cost set corresponding to the ith subject in this policy is Cost_i $(i = 1, 2, \ldots, n)$. K PDPs can be regarded as k disjoint sets, expressed in C_j $(j = 1, 2, \ldots, k)$. Cost_i $(i = 1, 2, \ldots, n)$ needs to be distributed, respectively, to C_j $(j = 1, 2, \ldots, k)$. We let $p_{ij} = 1$ if Cost_i $(i = 1, 2, \ldots, n)$ is distributed to C_j $(j = 1, 2, \ldots, k)$; otherwise we let $p_{ij} = 0$. In (2), S_j $(j = 1, 2, \ldots, k)$ stands for the sum value of the elements in C_j. Formula (3) is the constraint condition for (2) and indicates that Cost_i $(i = 1, 2, \ldots, n)$ can be distributed to only one of C_j $(j = 1, 2, \ldots, k)$:

$$S_j = \sum_{i=1}^{n} p_{ij} \text{Cost}_i, \tag{2}$$

$$\sum_{j=1}^{k} p_{ij} = 1 \quad (i = 1, 2, 3, \ldots, n). \tag{3}$$

It is significant that the maximum of S_j $(j = 1, 2, \ldots, k)$ must be as small as possible for policy decomposition. Accordingly, the cost of subpolicies deployed to each PDP is equal or approximately equal so that the purpose of improving the evaluation performance of PDP can be achieved. The optimal solution to the optimization model must satisfy the mathematical expression, as shown in

$$\min \max_{1 \le j \le k} S_j. \tag{4}$$

Input: Cost, k
Output: k subsets

(1) Cost$'$ = Heap_Sort(Cost)
(2) **for** the first k elements **do**
(3) distributed respectively to one of k subsets
(4) **end for**
(5) A min-heap *minHeap* with the first k elements is built
(6) **for** the remainder elements of Cost$'$ **do**
(7) distributed successively to one of k elements in *minHeap*, whose sum value is always kept the smallest
(8) *minHeap* is adjusted
(9) **end for**

Algorithm 1: Greedy algorithm of policy decomposition.

3.3. Properties. The model of policy decomposition is a discrete optimization model, which has the following properties.

Property 1. C_j $(j = 1, 2, \ldots, k)$, corresponding to the optimal solution of policy decomposition, is not unique.

Here is an example. Given Cost $= \{10, 20, 30, 40, 60, 80\}$ for 2 partitions, we can obtain two results of policy decomposition. One is that $C_1 = \{10, 20, 30, 60\}$ and that $C_2 = \{40, 80\}$, with $S_1 = S_2 = 120$, and the other is that $C_1' = \{10, 30, 80\}$ and that $C_2' = \{20, 40, 60\}$, with $S_1 = S_2 = 120$.

Property 2. The optimal solution S_j $(j = 1, 2, \ldots, k)$ of policy decomposition is unique.

Proof. Suppose that the optimal solution S_j $(j = 1, 2, \ldots, k)$ of policy decomposition is not unique and that there is another optimal solution S_p' $(p = 1, 2, \ldots, k)$. If $\max_{1 \le j \le k} S_j < \max_{1 \le p \le k} S_p'$, S_p' $(p = 1, 2, \ldots, k)$ cannot meet formula (4). If $\max_{1 \le j \le k} S_j > \max_{1 \le p \le k} S_p'$, S_j $(j = 1, 2, \ldots, k)$ contradicts the optimal solution. □

Property 3. C_j $(j = 1, 2, \ldots, k)$ corresponds to the optimal solution S_j $(j = 1, 2, \ldots, k)$. For any C_p $(C_p \in \{C_j \mid j = 1, 2, \ldots, k\})$, if any one of the elements of Cost_p $(\text{Cost}_p \in \{\text{Cost}_i \mid i = 1, 2, \ldots, n\})$ is extracted from C_p, a new set C_p' is obtained. The sum value of C_p' is $S_p' = S_p - \text{Cost}_p$, and then $S_p' \le S_j$ $(j = 1, 2, \ldots, k)$.

Proof. Suppose that there exists S_t $(S_t \in \{S_j \mid j = 1, 2, \ldots, k\})$ and $S_t < S_p'$; then $S_t + \text{Cost}_p < S_p' + \text{Cost}_p = S_p$, which indicates that a better solution results if Cost_p is added to C_t, which is corresponding to S_t. This better solution contradicts the optimal solution S_j $(j = 1, 2, \ldots, k)$. □

4. Greedy Algorithm of Policy Decomposition

Since the model of policy decomposition is a discrete optimization model, there is not a specific algorithm that can apply to the problem directly. In what is to follow, we construct a greedy algorithm for solving the problem of policy decomposition effectively.

In practical applications, if a policy is decomposed on the basis of the subject attribute, there exist the following two facts.

(i) The difference between values for each Cost_i $(i = 1, 2, \ldots, n)$ is not too much.

(ii) The number of the elements embodied in C_j $(j = 1, 2, \ldots, k)$ is not truly very large.

In view of these two existing facts, the main idea of the proposed greedy algorithm is as follows.

(i) The cost set Cost should be sorted in a descending order for obtaining a new set Cost$'$.

(ii) If k-partition is needed, the first k elements of Cost$'$ are distributed, respectively, to one of k null sets.

(iii) The remainder elements of Cost$'$ are distributed successively to one of the k sets, whose sum value of the elements present is the smallest.

(iv) The algorithm ends when the cost set Cost$'$ is empty.

The complete greedy algorithm of policy decomposition is shown in Algorithm 1, where the input is the cost set Cost and k (the number of subsets after policy decomposition) and the output is the k subsets. Our demand is that the maximum sum value of these subsets be as small as possible. In step (1), heap sort is used to sort the elements in the cost set Cost to obtain a new set Cost$'$. In steps (2)~(4), the first k elements in Cost$'$ are distributed, respectively, to the k subsets firstly. In step (5), A min-heap *minHeap* is then built on the basis of the k subsets. In steps (6)~(9), the remainder elements of Cost$'$ are distributed successively to one of the k subsets, whose sum value is always kept the smallest in each distribution. Meanwhile, *minHeap* also needs to maintain the min-heap property in each distribution.

The time complexity of the algorithm is analyzed as follows.

(i) Heap sort takes $O(nlgn)$ time.

(ii) The time that the first k elements in Cost$'$ are distributed, respectively, to R array is $O(k)$.

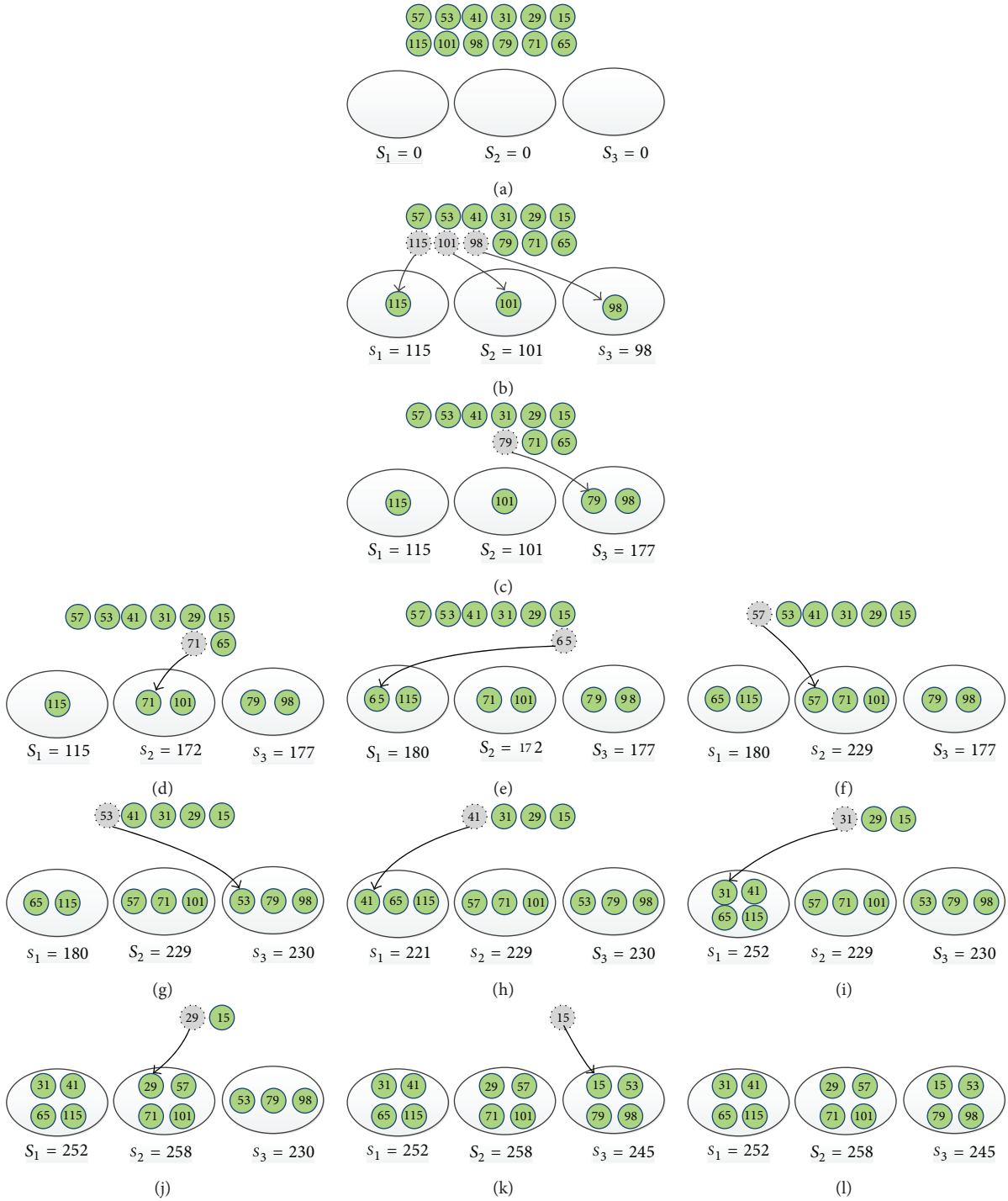


Figure 2: An example of decomposition procedure.

(iii) The time of building *minHeap* with k elements is $O(k)$.

(iv) The time that the remainder elements of Cost$'$ are distributed successively to *minHeap* with *minHeap* adjusted is $O((n-k) * lgk)$.

Based on the analyses above, the total time complexity of the greedy algorithm is $O(nlgn + 2 * k + (n-k) * lgk)$, or $O(nlgn)$.

An example is presented here. The greedy algorithm is used to decompose a cost set Cost = {79, 41, 29, 31, 65, 98, 53, 71, 15, 57, 101, 115} into three subsets, and the procedure of decomposition is shown in Figure 2 from (a) to (l). The decomposition result is that the three sum values of the subsets are $S_1 = 252$, $S_2 = 258$, and $S_3 = 245$. This result is not the optimal solution $S_1' = 252$, $S_2' = 252$, and $S_3' = 251$, but the difference between values for them is not truly very large.

Table 2: 20 cost sets with different numbers of elements.

Number	Cost set
Cost-1	{79, 41, 22, 31, 65, 96, 53, 71, 15, 101, 115, 57}
Cost-2	{44, 41, 50, 31, 65, 79, 67, 56, 69, 120, 189, 67}
Cost-3	{59, 5, 26, 32, 66, 123, 42, 27, 17, 79, 136, 78, 90}
Cost-4	{88, 91, 33, 1, 65, 98, 53, 23, 15, 101, 156, 34}
Cost-5	{56, 67, 29, 12, 65, 98, 53, 71, 15, 89, 167, 11}
Cost-6	{34, 41, 26, 31, 65, 122, 53, 67, 15, 23, 124, 56, 78}
Cost-7	{79, 65, 20, 25, 65, 45, 53, 90, 15, 45, 115, 89, 78}
Cost-8	{77, 23, 46, 65, 98, 53, 47, 54, 18, 29, 23, 84, 98, 70}
Cost-9	{56, 41, 16, 31, 65, 22, 53, 37, 66, 23, 91, 56, 55}
Cost-10	{79, 94, 99, 81, 71, 22, 63, 48, 95, 47, 34, 78}
Cost-11	{21, 51, 65, 95, 75, 84, 62, 34, 16, 12, 75, 110}
Cost-12	{5, 98, 49, 60, 37, 36, 110, 174, 164, 105, 87}
Cost-13	{94, 28, 67, 9, 18, 49, 76, 38, 60, 142, 80, 7}
Cost-14	{134, 29, 91, 76, 48, 25, 86, 42, 30, 5, 31, 88}
Cost-15	{58, 72, 39, 64, 13, 24, 51, 99, 14, 55, 101, 96}
Cost-16	{18, 90, 99, 69, 79, 99, 186, 208, 73, 48, 61}
Cost-17	{68, 75, 94, 18, 37, 90, 86, 43, 19, 37, 5, 66}
Cost-18	{37, 48, 98, 79, 35, 84, 19, 57, 27, 82, 64, 73}
Cost-19	{43, 96, 32, 87, 77, 16, 41, 9, 67, 39, 13, 25}
Cost-20	{71, 30, 91, 28, 64, 53, 79, 47, 83, 123, 62, 11}

In order to make further efforts to observe and compare the results of policy decomposition using the greedy algorithm, we randomly generate 20 cost sets with different numbers of elements, as shown in Table 2.

Without loss of generality, these 20 cost sets are decomposed, respectively, for 3, 4, and 5 partitions and their decomposition results are shown in Figure 3 from (a) to (c), which depicts the comparisons of the maximum sum values of the subsets obtained by decomposition with that of the theoretical optimal solutions. As shown in Figure 3, experimental results using the greedy algorithm for solving the problem of policy decomposition are close to the theoretical optimal solutions with a high degree of approximation.

Each of the experimental results using the greedy algorithm is unique and nonrandom. All the results cannot be guaranteed to be the optimal solutions, but the obtained approximate solutions are extremely close to the theoretical optimal solutions or even equal to the theoretical optimal solutions sometimes. This fact can meet the needs of the practical applications.

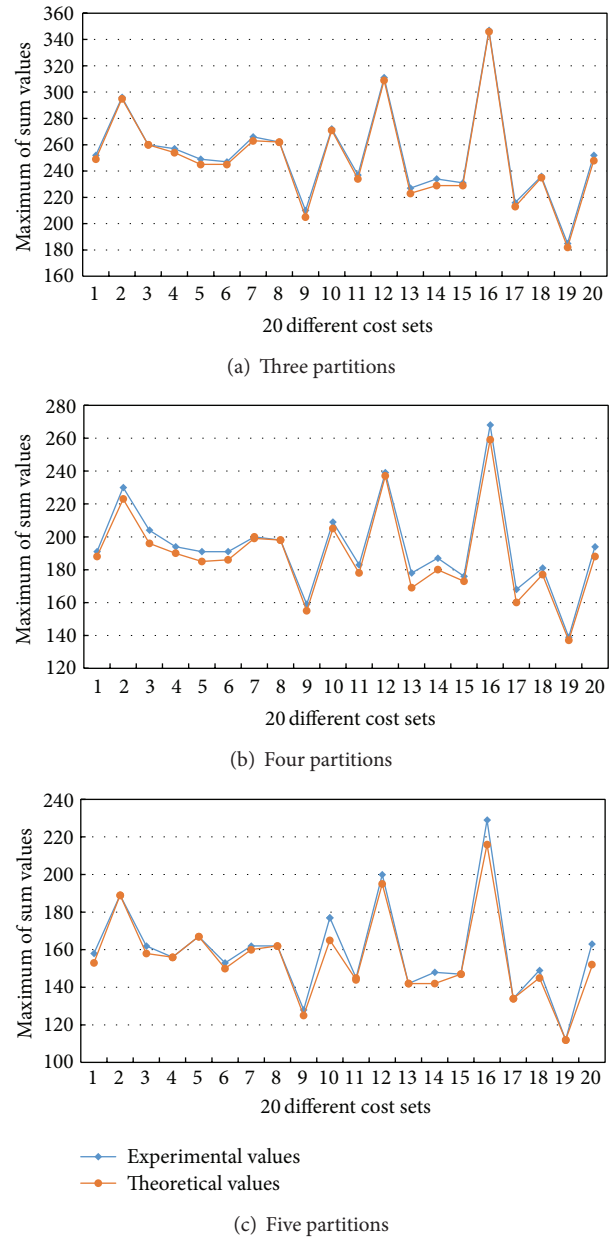


Figure 3: Comparisons of experimental values using greedy algorithm with theoretical values.

5. Experimental Results

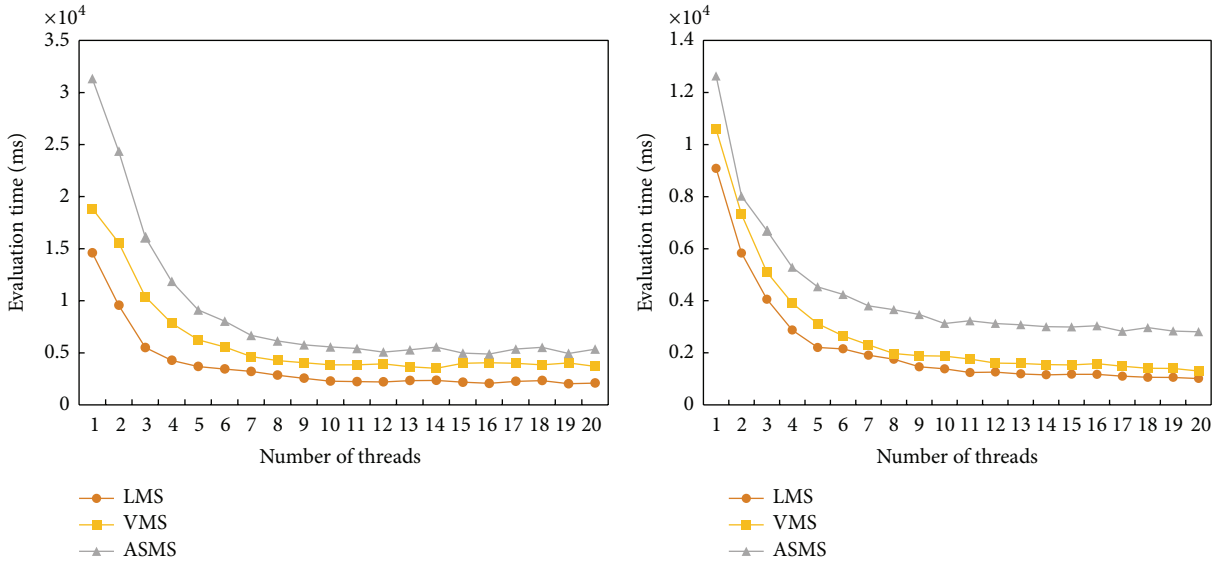
In order to assess the evaluation performance improvement of PDPs in XDPEE, the test policies and the generation of test requests are introduced firstly. We do the experiments as follows.

(i) The optimum number of threads is determined.

(ii) Comparisons of the evaluation performance of PDPs in XDPEE with that of PDPs in the Sun PDP are made. (Sun PDP [29] is a widely used policy decision point, which is adopted to evaluate requests as a decision engine in our experiments.)

(iii) The evaluation time of XDPEE with different numbers of PDPs is measured.

5.1. Test Policies. In order to simulate practical application scenarios, we select policies from practical systems [36–38].



(a) Sun PDP (b) XDPEE

Figure 4: Variations of evaluation time of 4 PDPs with number of threads.

Three adopted XACML access control policies in practical systems are as follows:

(i) *library management system (LMS):* the LMS provides access control policies by which a public library can use web services to manage books;

(ii) *virtual meeting system (VMS):* the VMS provides access control policies by which web conference services can be managed. The VMS allows users to organize online meetings in a distributed platform. When a user connects to the server, he/she can enter or exit a meeting, make a statement and ask questions at the meeting, and so forth. Every meeting has an administrator, whose responsibilities are initializing the meeting information and setting some parameters (such as the meeting's title and organization). The administrator can also assign to every meeting a host, who is capable of selecting a user to make a statement;

(iii) *auction sale management system (ASMS):* the ASMS provides access control policies by which items can be bought or sold online. A seller initializes the lowest price of and the description of an item which are allowed to be submitted when at auction. A user can participate in the bidding process by bidding the item. The restriction to a user is that there must be enough money in his/her account before bidding.

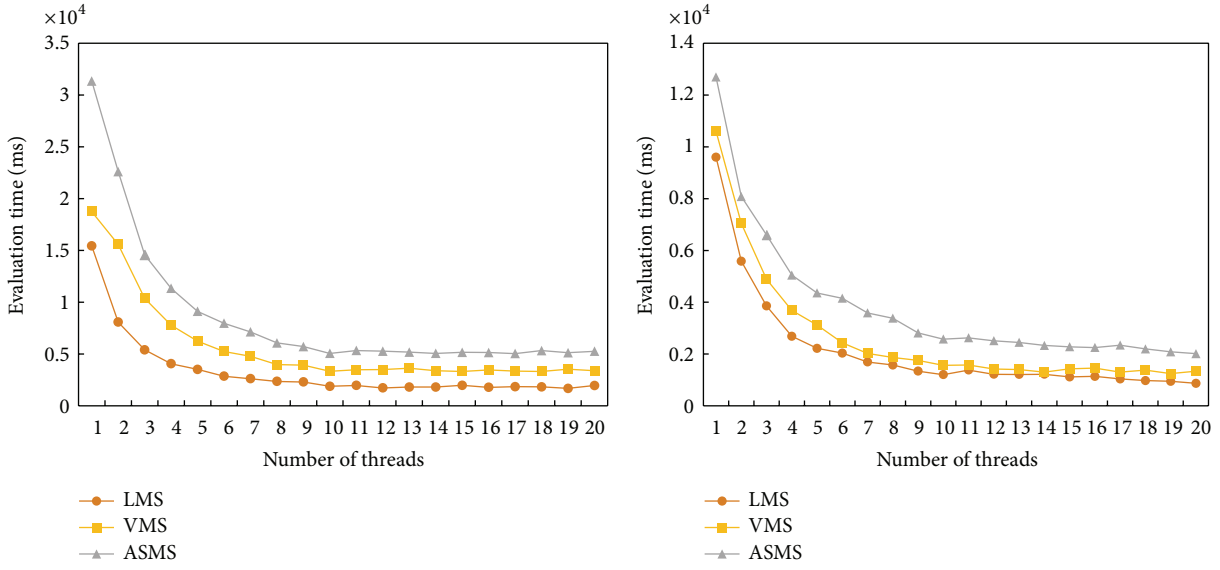
The policy of the LMS contains 720 rules, the VMS 945 rules, and the ASMS 1760 rules.

5.2. Generation of Test Requests. Martin and Xie [39] put forward that policies are analyzed by *Change-Impact* in order to automatically generate access requests that conform to Change-Impact. The purpose is to improve the coverage of test. The main idea is that conflicting policies or rules can be obtained by conflicting detection tools according to the fact that different policies or different rules in the same policy could make inconsistent results of evaluation for the same request and that correlative access requests can be constructed for testing according to the conflicting policies or rules.

Bertolino et al. [40] proposed that access requests can be automatically generated to test the correctness of PDP as well as the configured policies. They pointed out that the *Context Shema* which is defined by the XML Schema of XACML describes all the structures of the access requests that might be accepted by PDP or all the valid input requests. This paper indicates that their developed X-CREATE can generate possible structures of access requests according to the Context Shema of XACML. The policy analyzer obtains possible input values of every attribute from a policy. The policy manager adopts the method of random allocation to distribute the obtained input values into structures of access requests. Another test scheme is *Simple Combinatorial*, which can generate access requests according to all the possible combinations of attribute values of subject, action, and resource in XACML policies. This paper also analyzes the advantages of these two schemes in debugging.

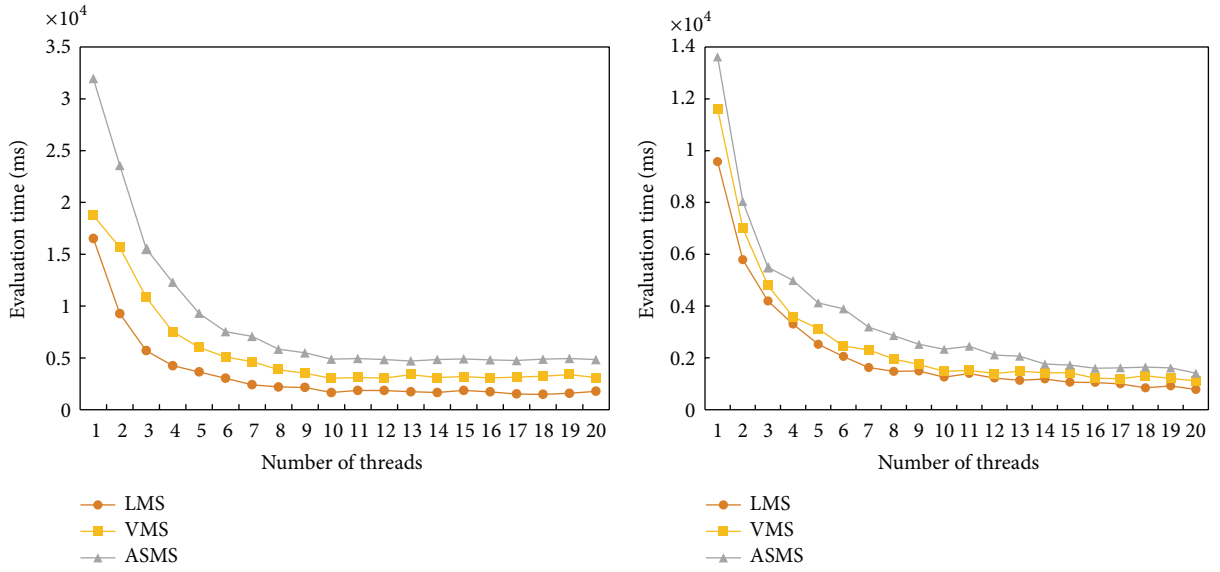
According to the actual requirement of performance test, the method for combining *Change-Impact*, *Context Shema*, and *Simple Combinatorial* is adopted to simulate the practical access requests.

5.3. Performance Tests and Comparisons. In the light of actual requirement, the policy of the LMS, VMS, and ASMS needs to be expanded separately to be three bigger policies, which contain more rules. According to the Cartesian product of different subjects, actions, resources, and conditions in all



(a) Sun PDP (b) XDPEE

Figure 5: Variations of evaluation time of 5 PDPs with number of threads.



(a) Sun PDP (b) XDPEE

Figure 6: Variations of evaluation time of 6 PDPs with number of threads.

rules of a policy, we conduct new rules and add them to the original policy. The number of rules in the policy of the LMS, VMS, and ASMS is expanded separately to 3000, 6000, and 9000. Finally, the policy of the LMS, VMS, and ASMS is decomposed separately into multiple subpolicies each with fewer rules. Policy decomposition guarantees that the cost of subpolicies deployed to each PDP is equal or approximately equal.

5.3.1. Determination of Optimum Number of Threads. The evaluation time of PDPs is related to the number of threads in the RDM. Accordingly, it is essential to determine the optimal number of threads in the experiment firstly. When the number of access requests is fixed (1000 access requests are generated randomly here), the variations of evaluation time of PDPs with number of threads are shown in Figures 4, 5, and 6. These figures describe the tests for XDPEE and the Sun PDP each with 4, 5, and 6 PDPs, respectively.

In Figures 4, 5, and 6, we observe that

(i) the evaluation time of PDPs reduces with the growing numbers of threads,

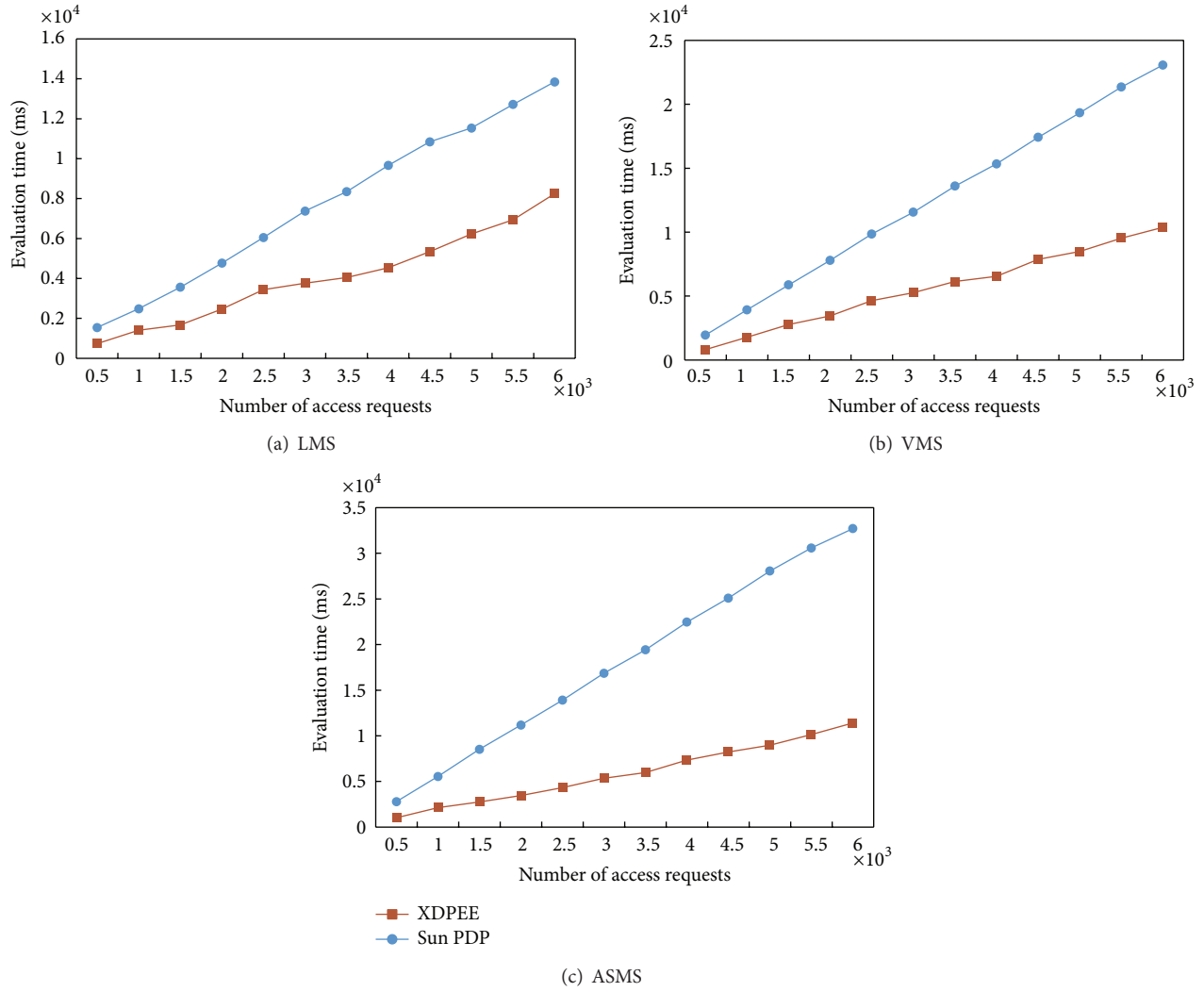


Figure 7: Variations of evaluation time of 4 PDPs with number of access requests.

(ii) when the number of threads is greater than 10, the evaluation time of PDPs tends to be a constant value.

Therefore, the number of threads in the RDM is set to 10 in the following experiments.

5.3.2. Performance Comparisons of XDPEE with Sun PDP. In order to assess the evaluation performance improvement of PDPs by using the method of policy decomposition, performance comparisons of XDPEE with the Sun PDP each with 4, 5, and 6 PDPs are made. We generate 500, 1000,..., 6000 access requests randomly to measure the evaluation time of PDPs. For the policy of the LMS, VMS, and ASMS, the variations of evaluation time of PDPs with number of access requests are shown in Figures 7, 8, and 9.

In Figures 7, 8, and 9, we observe that

(i) the evaluation time of PDPs increases when the number of access requests grows,

(ii) the growth rate of the evaluation time of PDPs in XDPEE is less than that of PDPs in the Sun PDP,

Table 3: Improvement (percentage) of evaluation performance.

	LMS	VMS	ASMS
4 PDPs	40.46%	55.05%	65.11%
5 PDPs	41.34%	55.97%	65.79%
6 PDPs	41.94%	56.79%	66.58%

(iii) when the number of access requests reaches 6000, the improvement (percentage) of the evaluation performance of PDPs in XDPEE compared with that of PDPs in the Sun PDP is shown in Table 3.

5.3.3. Measurements of Evaluation Time of XDPEE with Different Numbers of PDPs. In order to further assess the evaluation performance improvement of multiple PDPs, the evaluation time of XDPEE with 1, 2, 4, and 6 PDPs is measured. We generate 500, 1000,..., 6000 access requests

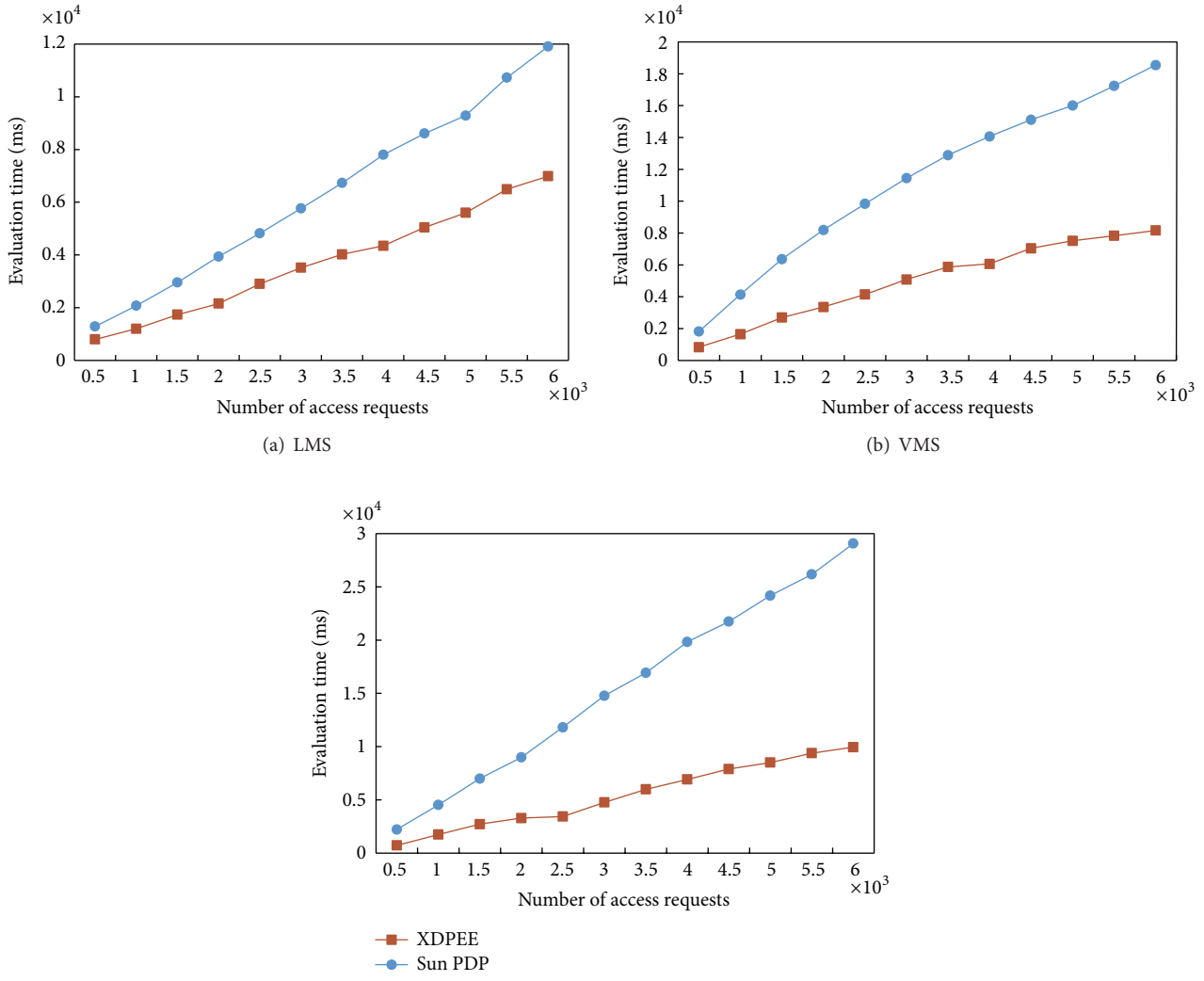


Figure 8: Variations of evaluation time of 5 PDPs with number of access requests.

randomly to measure the evaluation time of PDPs. For the policy of the LMS, VMS, and ASMS, the evaluation time of XDPEE with different numbers of PDPs is shown in Figure 10.

In Figure 10, we observe that

(i) the evaluation time of PDPs in XDPEE increases when the number of access requests grows,

(ii) the evaluation time of PDPs in XDPEE reduces with the growing numbers of PDPs,

(iii) when the number of access requests reaches 6000, the improvement (percentage) of evaluation time of 6, 4, and 2 PDPs compared separately with that of one single PDP is shown in Table 4.

Table 4: Improvement (percentage) of evaluation time.

	LMS	VMS	ASMS
6 PDPs	43.21%	49.11%	50.37%
4 PDPs	37.33%	38.31%	38.93%
2 PDPs	27.12%	28.42%	28.88%

6. Conclusions

When requests are evaluated, the evaluation performance improvement of PDP is affected by some factors, such as the number of rules embodied in a policy, the relative order of rules, and the number of access requests, etc. Based on these influencing factors and studies available, this paper proposes a distributed policy evaluation engine with several PDPs, where a policy decomposition module (PDM) and a request distribution module (RDM) are introduced. The pattern where there is only one single PDP in a centralized authorization model is changed. A policy should be decomposed into multiple subpolicies each with fewer

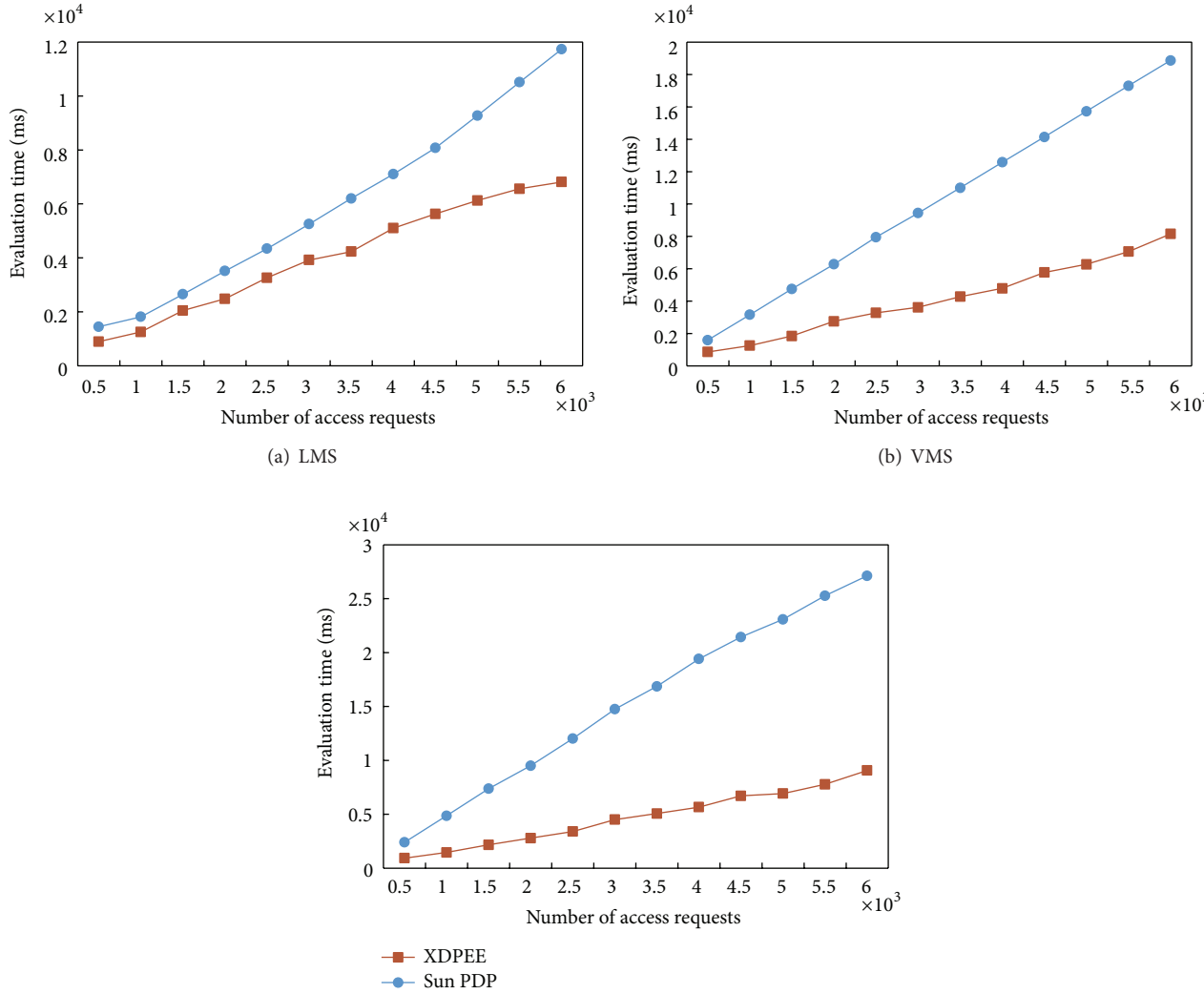


FIGURE 9: Variations of evaluation time of 6 PDPs with number of access requests.

rules so that the cost of subpolicies deployed to each PDP is equal or approximately equal. As a result, the average evaluation time of PDPs can be effectively shortened, so that the evaluation performance of PDPs can be improved substantially.

This paper also presents a discrete optimization model of policy decomposition. According to the properties of the optimization model, a greedy algorithm with a favorable time complexity is constructed for solving the model.

In experiments, the optimal number of threads in the RDM is determined. Comparisons of the evaluation performance of PDPs in XDPEE with that of PDPs in the Sun PDP are made. Also, the evaluation time of XDPEE with different numbers of PDPs is measured. For the policy of the LMS, VMS, and ASMS, experimental results show that

(i) the evaluation time of PDPs in XDPEE is less than that of PDPs in the Sun PDP, no matter how great the number of access requests is,

(ii) the more the numbers of PDPs are, the shorter the evaluation time of PDPs in XDPEE will be.

In our simulation of a considerable number of access requests, the evaluation performance of PDPs in XDPEE is improved effectively, which can meet the needs of authorization services preferably in SOA environment. Meanwhile, we have to focus attention on ensuring the correctness of authorization services. At present, the authorization is carried out on the basis of the policies configured by administrators. Future research will concentrate on how to adopt the security and risk assessment mechanism of authorization to improve its correctness.

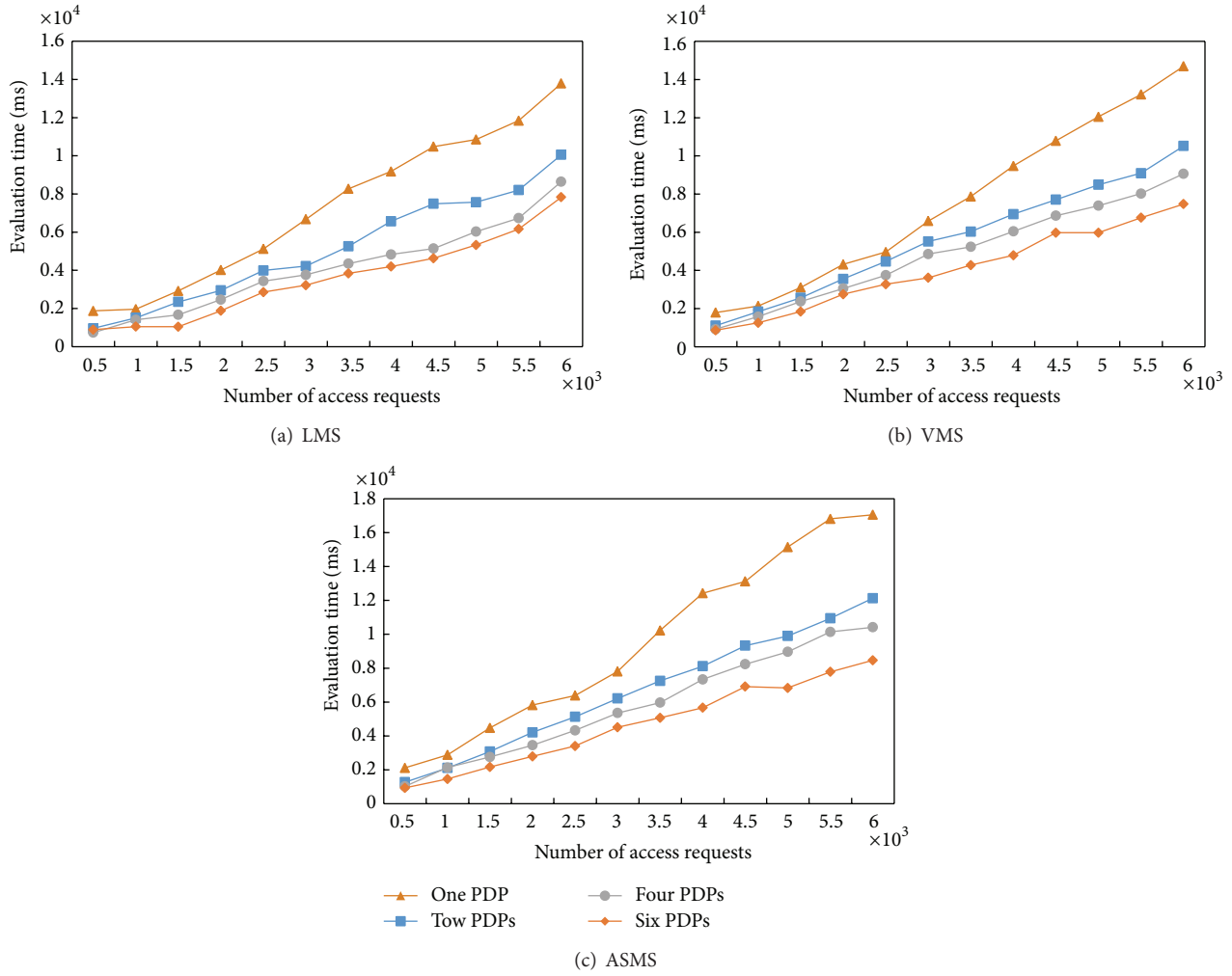


Figure 10: Evaluation time of XDPEE with different numbers of PDPs.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is supported by the State-funded project of the Higher Education and Fundamental Scientific Research in China.

References

[1] M. Bell, *SOA Modeling Patterns for Service-Oriented Discovery and Analysis*, John Wiley & Sons, Hoboken, NJ, USA, 2010.

[2] F. Hojaji and M. R. A. Shirazi, "Developing a more comprehensive and expressive SOA governance framework," in *Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME '10)*, pp. 563–567, Chengdu, China, April 2010.

[3] Y. C. Zhou, X. P. Liu, X. N. Wang, L. Xue, C. Tian, and X. X. Liang, "Context model based SOA policy framework," in *Proceedings of the IEEE 8th International Conference on Web Services (ICWS '10)*, pp. 608–615, Miami, Fla, USA, July 2010.

[4] Y. I. Nurhasan, B. Dabarsyah, and H. Fakhrurroja, "Information model design as model-driven for Service Oriented Architecture (SOA) implementation in PK-BLU institution using SOA Ontology: case study: financial administration bureau Padjadjaran University," in *Proceedings of the International Conference on ICT for Smart Society (ICISS '13)*, pp. 1–9, Jakarta, Indonesia, 2013.

[5] F. Hojaji and M. R. A. Shirazi, "A comprehensive SOA governance framework based on COBIT," in *Proceedings of the 6th World Congress on Services (Services-1 '10)*, pp. 407–414, Miami, Fla, USA, July 2010.

[6] N. Dan, S. H. Ji, C. Yuan, and G. J. Hu, "Attribute based access control (ABAC)-based cross-domain access control in service-oriented architecture (SOA)," in *Proceedings of the International Conference on Computer Science & Service System (CSSS '12)*, pp. 1405–1408, Nanjing, China, 2012.

[7] W. She, I.-L. Yen, F. Bastani, B. Tran, and B. Thuraisingham, "Role-based integrated access control and data provenance for SOA based net-centric systems," in *Proceedings of the 6th IEEE*

International Symposium on Service-Oriented System Engineering (SOSE '11), pp. 225–234, Irvine, Calif, USA, December 2011.

[8] M. Jung, T. Hofer, S. Dobelt, G. Kienesberger, F. Judex, and W. Kastner, "Access control for a Smart Grid SOA Internet," in *Proceedings of the International Conferece for Technology and Secured Transactions*, pp. 281–287, London, UK, 2012.

[9] G. H. Hwang, C. W. Lee, and Z. X. Jiang, "Workflow-based dynamic access control in a service-oriented architecture," in *Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA '12)*, pp. 47–52, Fukuoka, Japan, 2012.

[10] M. Kassou and L. Kjiri, "A maturity metric based approach for eliciting SOA security requirements," in *Proceedings of the National Days of Network Security and Systems (JNS '12)*, pp. 7–11, Marrakech, Morocco, 2012.

[11] E. Sonchaiwanich, J. Zhao, C. Dowin, and M. McRoberts, "Using AOP to separate SOA security concerns from application implementation," in *Proceedings of the IEEE Military Communications Conference (MILCOM '10)*, pp. 470–474, San Jose, Calif, USA, November 2010.

[12] N. Kabbani, S. Tilley, and L. Pearson, "Towards an evaluation framework for SOA security testing tools," in *Proceedings of the 4th International Systems Conference*, pp. 438–443, San Diego, Calif, USA, April 2010.

[13] N. Shahgholi, M. Mohsenzadeh, M. A. Seyyedi, and S. H. Qorani, "A new SOA security framework defending web services against WSDL attacks," in *Proceedings of the 3rd IEEE International Conference on Social Computing (socialcom '11)*, pp. 1259–1262, Boston, Mass, USA, October 2011.

[14] N. Ahmed, R. Gamble, M. Linderman, and B. Bhargava, "Analysis of End-to-End SOA Security Protocols with Mobile Devices," in *Proceedings of the 14th IEEE International Conference on Mobile Data Management (MDM '13)*, pp. 166–170, Milan, Italy, 2013.

[15] N. Kabbani and S. Tilley, "Evaluating the capabilities of SOA security testing tools," in *Proceedings of the 5th IEEE International Systems Conference (SysCon '11)*, pp. 129–134, Montreal, Canada, April 2011.

[16] Z. Liu, L. Gu, Y. Yang, and G. Xing, "An identity authentication scheme based on USB key for trusted network connect," in *Proceedings of the IEEE International Conference on Information Theory and Information Security (ICITIS '10)*, pp. 203–207, Beijing, China, December 2010.

[17] Y. N. Sun, X. H. Guan, T. Liu, and Y. Qu, "An identity authentication mechanism based on timing covert channel," in *Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '12)*, pp. 832–836, Liverpool, UK, 2012.

[18] B. Li, "On identity authentication technology of distance education system based on voiceprint recognition," in *Proceedings of the 30th Chinese Control Conference (CCC '11)*, pp. 5718–5721, Yantai, China, July 2011.

[19] K. Peng, J. Zheng, and J. Yang, "An identity authentication system based on mobile phone token," in *Proceedings of the IEEE International Conference on Network Infrastructure and Digital Content*, pp. 570–575, Beijing, China, November 2009.

[20] Y. G. Yang, H. Y. Wang, X. Jia, and H. Zhang, "A quantum protocol for (t, n)-threshold identity authentication based on greenberger-horne-zeilinger states," *International Journal of Theoretical Physics*, vol. 52, no. 2, pp. 524–530, 2013.

[21] N. C. N. Chu and K. E. Barker, "Dynamic Role Lease Authorization for a Grid/Cloud," in *Proceedings of the 10th International Conference on ICT and Knowledge Engineering*, pp. 63–70, Bangkok, Thailand, 2012.

[22] C. Liu and L. Z. Liu, "A trust evaluation model for dynamic authorization," in *Proceedings of the International Conference on Computational Intelligence and Software Engineering (CiSE '10)*, pp. 1–4, Wuhan, China, December 2010.

[23] H. Xie, B. Zhang, and D. Y. Hu, "A role-based dynamic authorization model and its implementation in PMI," in *Proceedings of the International Conference on Computer Science and Software Engineering (CSSE '08)*, pp. 661–664, Wuhan, China, December 2008.

[24] D. Tang, J. Guo, and Q. Zhang, "A dynamic workflow authorization method based on participant expression rules," in *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering (CSAE '11)*, pp. 345–349, Shanghai, China, June 2011.

[25] J. J. Wang, J. P. Li, Y. F. Li, and J. Peng, "Review of key-based dynamic trust authorization mechanism," in *Proceedings of the International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP '12)*, pp. 263–267, Chengdu, China, 2012.

[26] M. Xu, D. Wijesekera, and X. Zhang, "Runtime administration of an RBAC profile for XACML," *IEEE Transactions on Services Computing*, vol. 4, no. 4, pp. 286–299, 2011.

[27] A. Mourad, H. Otrok, H. Yahyaoui, and L. Baajour, "Toward an abstract language on top of XACML for web services security," in *Proceedings of the International Conference for Internet Technology and Secured Transactions (ICITST '11)*, pp. 254–259, Abu Dhabi, United Arab Emirates, December 2011.

[28] C. Ran and G. Guo, "Security XACML access control model based on SOAP encapsulate," in *Proceedings of the International Conference on Computer Science and Service System (CSSS '11)*, pp. 2543–2546, Nanjing, China, June 2011.

[29] D. E. Kateb, T. Mouelhi, Y. L. Traon, J. Y. Hwang, and T. Xie, "Refactoring access control policies for performance improvement," in *Proceedings of International Conference on Performance Engineering (ICPE '12)*, pp. 323–334, New York, NY, USA, 2012.

[30] A. Alzahrani, H. Janicke, and S. Abubaker, "Decentralized XACML overlay network," in *Proceedings of the 10th IEEE International Conference on Computer and Information Technology*, pp. 1032–1037, Bradford, UK, July 2010.

[31] M. Decat, B. Lagaisse, and W. Joosen, "Toward efficient and confidentiality-aware federation of access control policies," in *Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing (MW4NG '12)*, New York, NY, USA, 2012.

[32] R. Craven, J. Lobo, E. Lupu, A. Russo, and M. Sloman, "Policy refinement: decomposition and operationalization for dynamic domains," in *Proceedings of the 7th International Conference on Network and Service Management (CNSM '11)*, pp. 115–123, Laxenburg, Austria, October 2011.

[33] A. X. Liu, F. Chen, J. Y. Hwang, and T. Xie, "Designing fast and scalable XACML policy evaluation engines," *IEEE Transactions on Computers*, vol. 60, no. 12, pp. 1802–1817, 2011.

[34] S. Marouf, M. Shehab, A. Squicciarini, and S. Sundareswaran, "Adaptive reordering and clustering-based framework for efficient XACML policy evaluation," *IEEE Transactions on Services Computing*, vol. 4, no. 4, pp. 300–313, 2011.

[35] Y. Z. Wang, D. G. Feng, L. W. Zhang, and M. Zhang, "XACML policy evaluation engine based on multi-level optimization technology," *Journal of Software*, vol. 22, no. 2, pp. 323–338, 2011.

[36] Y. le Traon, T. Mouelhi, A. Pretschner, and B. Baudry, "Test-driven assessment of access control in legacy applications," in *Proceedings of the 1st International Conference on Software Testing, Verification and Validation (ICST '08)*, pp. 238–247, Lillehammer, Norway, April 2008.

[37] T. Mouelhi, F. Fleurey, B. Baudry, and Y. le Traon, "A model-based framework for security policy specification, deployment and testing," in *Model Driven Engineering Languages and Systems*, vol. 5301 of *Lecture Notes in Computer Science*, pp. 537–552, Springer, Berlin, Germany, 2008.

[38] T. Mouelhi, Y. L. Traon, and B. Baudry, "Transforming and selecting functional test cases for security policy testing," in *Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation (ICST '09)*, pp. 171–180, Denver, Colo, USA, April 2009.

[39] E. Martin and T. Xie, "Automated test generation for access control policies," in *Proceedings of the IEEE International Symposium on Software Reliability Engineering (ISSRE '06)*, pp. 752–753, New York, NY, USA, November 2006.

[40] A. Bertolino, S. Daoudagh, F. Lonetti, and E. Marchetti, "Automatic XACML requests generation for policy testing," in *Proceedings of the 5th IEEE International Conference on Software Testing, Verification and Validation*, pp. 842–849, Montreal, Canada, 2012.