Similarity Measure for Security Policies in Service Provider Selection

Yanhuang Li[1,2], Nora Cuppens-Boulahia[2], Jean-Michel Crom[1], Frédéric Cuppens[2], Vincent Frey[1], and Xiaoshu Ji[1]

[1] Orange Labs, 4 rue du Clos Courtel, 35510, Cesson-Sévigné, France
[2] Télécom Bretagne, 2 rue de la Châtaigneraie, 35510, Cesson-Sévigné, France
{yanhuang.li,jeanmichel.crom,vincent.frey,xiaoshu.ji}@orange.com
{nora.cuppens,frederic.cuppens}@telecom-bretagne.eu

Abstract. The interaction between different applications and services requires expressing their security properties. This is typically defined as security policies, which aim at specifying the diverse privileges of different actors. Today similarity measure for comparing security policies becomes a crucial technique in a variety of scenarios, such as finding the cloud service providers which satisfy client's security concerns. Existing approaches cover from semantic to numerical dimensions and the main work focuses mainly on XACML policies. However, few efforts have been made to extend the measure approach to multiple policy models and apply it to concrete scenarios. In this paper, we propose a generic and light-weight method to compare and evaluate security policies belonging to different models. Our technique enables client to quickly locate service providers with potentially similar policies. Comparing with other works, our approach takes policy elements' logic relationships into account and the experiment and implementation demonstrate the efficiency and accuracy of our approach.

Keywords: IT Security, Access Control, Policy Evaluation, Similarity Measure.

1 Introduction

Nowadays, data and service exchange across multiple actors becomes an emerging demand to provide dynamic ecosystems. This process involves a large number of actors such as cloud service provider (SP) and client. From customer's point of view, it is always difficult to decide whose service should be chosen so they use a broker to rank and select the suitable SPs based on user's requirement. However, most of the current service ranking technologies [1] do not consider the security aspect or they only measure security parameters such as encryption method [2] and security level [3,4]. Among various criteria that need to be considered for the service selection, security policy is a critical concern. Before a collaboration takes place between different actors, an actor A may need to know if the other actor guarantees a similar level of A's security policies. Policy comparison is one of the main mechanisms to that end. It consists in measuring the

"This paper has been accepted at the 11th International Conference on Information Systems Security (ICISS2015) (http://www.iciss.org.in/iciss-2015/default.htm). The final publication is available at Springer via: https://dx.doi.org/10.1007%2F978-3-319-26961-0_14 ."

similarity between two security policies and giving an evaluation score. A higher score between policies p_1 and p_2 indicates that they are more likely to share an equivalent security level and yield the same decisions. Unlike other measure criteria, security policies are usually based on first-order logic. For example, an access control policy consists of multiple elements and they collectively determine whether a user is allowed to take some actions on certain objects. Thus, the existing brokering technologies are difficult to apply on security policies.

In this paper, we propose a new algorithm to calculate the similarity score between two policies. The contribution is twofold. On one hand, our method is policy-agnostic and can be applied on various types of security policies. On the other hand, we propose integrating our policy similarity measure algorithm in SP selection process and the implementation proves that this integration can enrich the services offered with efficiency.

The rest of the paper is organized as follows: Section 2 reviews existing proposals on security policy models and policy similarity measure techniques. Section 3 proposes the policy similarity measure algorithm with an exhaustive calculation example. Section 4 illustrates an experiment in which the accuracy of our algorithm is demonstrated. Section 5 gives an implementation integrated with our algorithm. Section 6 concludes the paper and outlines future work.

2 Related Work

To present our policy evaluation method, we suggest, as a first step, to specify security policies which describe and control different exchanges within a dynamic environment of diverse applications. In this context, the administrator of these applications has to define what is permitted and what is prohibited during the execution in order to secure the use of the proposed services. To do that, he should specify the security policy to be implemented. Access control policy is one kind of such policies. An access policy governs access to protected resources by specifying which subjects can access which resources by which operations and under which circumstances. The specification of access control policy depends on different policy models. One widely used model is RBAC (Role-Based Access Control) [5]. In the RBAC model, access permissions are not assigned directly to the users but are abstracted as "roles" which correspond to different task descriptions. To apply RBAC, users should be assigned to different roles thus they possess indirectly the relevant permissions. The OrBAC (Organization Based Access Control) model [6] is an extension of the RBAC model. It defines a conceptual and industrial framework to meet the needs of information security and sensitive communications and allows the policy designer to define a security policy independently. With the development of web service, ABAC (Attribute Based Access Control) model [7] brings flexibility and interoperability for policy definition. The ABAC model defines permissions based on security-relevant attributes such as subject attributes, resource attributes and environment attributes.

To the best of our knowledge, most approaches to evaluate policy similarity

are based on XACML [8] policies. Lin et al. [9] propose an algorithm to evaluate policy similarity by calculating the similarity score between two XACML policies. This is indeed a pioneering work and it effectively distinguishes the categorical predicate and numerical predicate cases. The second version of the algorithm [10] advances the measure algorithm for numerical predicate and integrates ontology matching. However, the work has two limitations. Firstly, the algorithm only focuses on the literal level but not logic aspect of security policy. As a result, the similarity score computed may have a large difference with the test value in real cases (presented in Appendix A) . Secondly, the former algorithm contains 9 weight parameters which need to be configured and choosing the proper values is not easy to users. In addition, there are two variants of the former work. Bei et al. [11] investigate the contrary of the similarity: dissimilarity. In order to address the rule relationship comparison, they apply fuzzy theory to compute rule dissimilarity. Pham et al. [12] improve the similarity computing approach specified by Lin et al. [9] and also propose a mechanism to calculate a dissimilarity score by identifying related policies which are likely to produce different access decisions. Based on policy similarity measure, there exist some applications. Lin et al. [13] present a novel data protection framework in which policy similarity comparison approach is applied on policy ranking model. Cho et al. [14] propose a technique that allows similarity evaluation of encrypted policies. Shaikh et al. [15] suggest using similarity measure to select services in a distributed and heterogeneous environment. Bertolino et al. [16] put forward a new approach for access control test prioritization based on similarity.

3 Policy Similarity Measure (PSM)

The PSM assigns a similarity score S_{policy} for any two given policies, which approximates the percentage of the rule pairs having the same decision. The formal definition is given in Equation (1), where $Num(sameDecision(r_{1i}, r_{2j}))$ denotes the quantity of the rule pairs having the same decision and $Num(allDecision(r_{1i}, r_{2j}))$ denotes the amount of the total decision pairs.

$$S_{policy}(p_1, p_2) \approx \frac{Num(sameDecision(r_{1i}, r_{2j}))}{Num(allDecision(r_{1i}, r_{2j}))}, \qquad r_{1i} \in p_1,\ r_{2j} \in p_2 \quad (1)$$

The similarity score is a value between 0 and 1. Two equivalent policies are expected to obtain a similarity score which equals 1. We mention that the definition of policy similarity score in [10] focuses on the percentage of the requests obtaining the same decisions. Comparing with the former work, our definition for PSM is more fine-grained because the same decision from two policies can be derived from one or multiple rule pairs. Consequently, by considering decisions of rule pairs but not final policy decisions, our PSM is more accurate from both calculation and test aspects. More details are shown in Section 4.

3.1 Policy Structure

As a generic algorithm, our PSM can be applied on different policy models and this compatibility requires a transformation process before calculation. Policies are firstly split into different rules and each rule is expressed in the form of:

$$decision\_effect(attr\_name_1 \oplus attr\_value_1, ..., attr\_name_n \oplus attr\_value_n) \quad (2)$$

where *decision_effect* is a decision effect such as permit and deny; *attr_name* denotes the name of an attribute; $\oplus$ indicates a comparison operator and *attr_value* represents an attribute value. We define $(attr\_name_i \oplus attr\_value_i)$ as a policy element and it can be broadly classified into the following five categories [17,18]:

Category 1: One variable equality constraints. $x = c$, where x is a variable and c is a constant.
Category 2: One variable inequality constraints. $x \triangleright c$, where x is a variable, c is a constant and $\triangleright \in \{<, \leq, >, \geq\}$.
Category 3: Real valued linear constraints. $\sum_{i=1}^{n} a_i x_i \triangleright c_i$, where x_i is a variable, a_i, c_i are constants and $\triangleright \in \{=, <, \leq, >, \geq\}$. This category contains conjunctions of atomic boolean expressions defined by linear constraints in m-dimensional real space.
Category 4: Regular expression constraints. The general form of boolean expression in this category is any element formed using $\wedge$ and $\vee$ with expressions of the form either $s \in L(r)$ or $s \notin L(r)$, where s is a *string* variable and $L(r)$ is the language generated by regular expression r.
Category 5: Compound Boolean expression constraints. This category includes constraints obtained by combining elements belonging to the categories listed above. The combination operators can be $\wedge, \vee$ and $\neg$.

It is worth noting that elements in most security policies usually belong to category 1 2 and 3. In this paper, we are not going to address how to deal with category 4 because expressing security policy by generated language is out of scope of basic security policy definition. We would also like to mention that the categories listed above are not mutually exclusive. For example, the expression "$8:00 \leq Time \leq 18:00$" which belongs to category 3 can be also expressed by category 5: "$(8:00 \leq Time) \wedge (Time \leq 18:00)$". In our formalization, in order to minimize the expected computational burden, we avoid the use of category 5 by transforming the policy elements with Boolean combinations into category 3.

We would also like to note that each element in Form 2 after transformation should be atomic. An element is atomic if it does not contain explicitly compound logical operator $(\wedge, \vee, \neg)$. By this definition, an atomic element can belong to category 1, 2 and just one dimension of category 3. In category 5, an element whose attribute values are connected by "$\vee$" operator can be expressed by a set. Here we don't consider "$\neg$" operator for the reason that "$\neg$" relation can be converted into rules having contrary effects. Having different types of attribute values, atomic elements in security policies can be divided into the

following two types:

- **Categorical element**: The operator is " = " and the attribute value belongs to the *string* data type or be a set of *string*. For example "Role=admin" and "Action=[read,write,create]" are categorical atomic elements.
- **Numerical element**: The operator can be " = ", " $<$ ", " $\leq$ ", " $>$ ", " $\geq$ " and the attribute value can be integer, real, date/time data types. Operators and values can be combined into a set or an interval. For example, elements "time=$\{3\text{ pm}, 4\text{ pm}, 5\text{ pm}\}$", "$FileSize > 5$ GB", "$8:00 \leq Time \leq 18:00$" are numerical atomic elements.

In an example that we will use throughout the paper, we consider three XACML policies illustrated in [10]. These policies are defined for managing an information system of a research laboratory. The policies after transformation are:

policy1 (p_1)
$\mathbf{r_{11}}$: $Permit(Role = \{professor, postDoc, student, techStaff\}$,
$Resource = \{source, documentation, executable\}, Action = \{read, write\})$
$\mathbf{r_{12}}$: $Deny(Role = \{student, postDoc, techStaff\}$,
$Resource = \{source, documentation, executable\}, Action = write$,
$19:00 \leq Time \leq 21:00)$

policy2 (p_2)
$\mathbf{r_{21}}$: $Permit(Role = \{student, faculty, techStaff\}, Action = \{read, write\}$,
$FileSize \leq 120$ MB)
$\mathbf{r_{22}}$: $Permit(Role = techStaff, Action = \{read, write\}, 19:00 \leq Time \leq 22:00)$
$\mathbf{r_{23}}$: $Deny(Role = student, Action = write, 19:00 \leq Time \leq 22:00)$
$\mathbf{r_{24}}$: $Deny(Role = \{student, faculty, staff\}, Action = \{read, write\}$,
$Resource = media)$

policy3 (p_3)
$\mathbf{r_{31}}$: $Permit(Role = businessStaff, Resource = xls, Action = \{read, write\}$,
$8:00 \leq Time \leq 17:00, FileSize \leq 10$ MB)
$\mathbf{r_{32}}$: $Deny(Role = student, Action = \{read, write\})$

From a user's perspective, p_1 is more similar to p_2 than p_3 because most activities described by p_1 for the data owner are allowed by p_2. Our motivation is to quickly compute similarity scores $S_{policy}(p_1, p_2)$ and $S_{policy}(p_1, p_3)$ with expectation that the former is higher than the latter. The expected result is to indicate that the similarity between p_1 and p_2 is much higher than the similarity between p_1 and p_3.

3.2 Overview of PSM Algorithm

Shown in Figure 1, the PSM algorithm takes two policies as input and generates a similarity score as output. The calculation process can be divided into four steps.

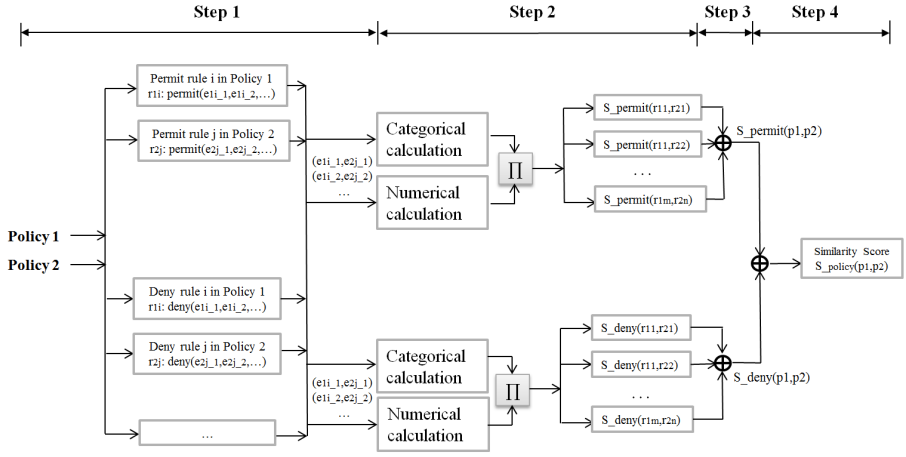

Fig. 1: The process of similarity score calculation

Step 1: Policy transformation. Two policies to be computed are split into rules in Form 2 which consist of atomic elements as follows.

$$r_{1i} : Permit(e_{1i\_1}, e_{1i\_2}, ...),\ r_{2j} : Permit(e_{2j\_1}, e_{2j\_2}, ...),\ ...$$

Step 2: Rule pair calculation. Scores of each rule pair in the same decision effect (d) between two policies are calculated. In Equation (3), the score for each rule pair is the product of the scores of all the element pairs. Product operation is chosen because any mismatch of element pair results different replies from two policies. Details for element pair calculation are shown in Section 3.3.

$$S_d(r_{1i}, r_{2j}) = \prod_k S(e_{1i\_k}, e_{2j\_k}), \qquad r_{1i} \in p_1,\ r_{2j} \in p_2,\ e_{1i\_k} \in r_{1i},\ e_{2j\_k} \in r_{2j} \tag{3}$$

Step 3: Decision effect calculation. Each $S_d(p_1, p_2)$ equals the sum of all the similarity scores of rule pairs in one decision effect (Equation (4)).

$$S_d(p_1, p_2) = \sum_i \sum_j S_d(r_{1i}, r_{2j}), \qquad r_{1i} \in p_1,\ r_{2j} \in p_2 \tag{4}$$

Step 4: Total score calculation. Shown in Equation (5), the total score is based on the scores from different decision effects $S_d(p_1, p_2)$ and the total amount of rule pairs from all the decision effects.

$$S_{policy}(p_1, p_2) = \frac{\sum_d S_d(p_1, p_2)}{\sum_d Num(d)}, \qquad d \in (permit, deny, ...) \tag{5}$$

3.3 Similarity Score of Rule Elements

The score of an element pair can be calculated when they share the same attribute name and in the same decision effect. In step 2 above, the score of a rule

pair is based on the rule elements having the same attribute name. When an element's attribute name does not appear in another rule, the access decisions from the two rules are not affected due to this difference. For this reason, we consider that the score of such element is 1. The calculation for similarity score of rule elements differs in element type.

Similarity Score for Categorical Elements. For categorical elements, we measure the exact match of two values. A higher score indicates that the two elements share more common attribute values. The formula for two elements e_1 and e_2 is defined as follows:

$$S_c(e_1, e_2) = \frac{num(v_1 \cap v_2)}{num(v_1 \cup v_2 \cup v_3 ... \cup v_n)} \tag{6}$$

$S_c(e_1, e_2)$ presents the exact percentage of the same decision for one element pair. $num(v_1 \cap v_2)$ denotes the quantity of common attribute values between element e_1 and e_2; $num(v_1 \cup v_2 \cup v_3 ... \cup v_n)$ is the quantity of common attribute values among all the elements in two policies and these elements should 1) have the same attribute name 2) belong to the rules of the same decision effect. Equation (6) is an extension of Jaccard similarity coefficient [19]. The difference is that the denominator in our equation covers two policies but not two rules because the aggregation of element scores in decision effect calculation(Equation 4) requires the same attribute space shared by different rule pairs.

Some policy models may use abstract element to represent a set of concrete values. For example, in RBAC, *Role* element is an abstraction of *Subjects*; in OrBAC, a *Role* is a set of *Subjects*, an *Activity* is a set of *Actions* and a $View$ is a set of *Objects*. In this case, the abstract values should be transformed to their related concrete values. For example, abstraction trees for *Role* and *Resource* elements of p_1, p_2, p_3 are shown in Figure 2, 3.

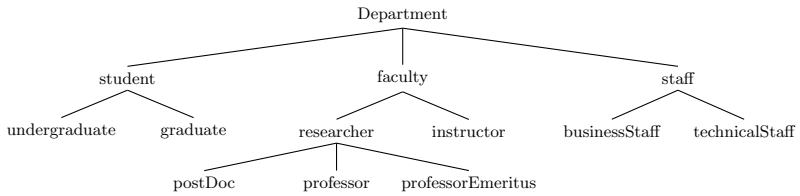

Fig. 2: Abstraction tree for Role element.

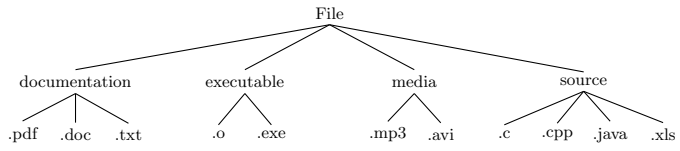

Fig. 3: Abstraction tree for Resource element.

To calculate the score of *Role* elements between r_{11} and r_{21}, as *student* and $faculty$ are two abstract values, they should be translated into concrete values which are leaves: $\{undergraduate, graduate\}$ and $\{postDoc, professor, professor - Emeritus, instructor\}$. After the transformation, we find that the two elements share 5 common attribute values. The disjunction of all the *Role* elements from policy 1 and policy 2 contains 8 attribute values. Applying Equation (6), $S_c(e_{r\_11(Role)}, e_{r\_21(Role)}) = 5/8 = 0.625$.

Another application of tree architecture is to represent the inheritance relation. The inheritance mechanism is defined in object-oriented programming as an efficient way to design an application. In Java, a class which is derived from another class is called a subclass. A similar mechanism for roles is used in RBAC [5] and the hierarchy of roles is associated with inheritance of permission. The role inheritance mechanism is extended in OrBAC model [20]: hierarchies of roles, views and activities are formally defined associated with inheritance relationships. In an inheritance tree, child elements can inherit the privileges of their parent elements. For example, the *Role* elements of a research laboratory may possess an inheritance tree for permission (Figure 4). When applying Equation (6), all the attribute values having inheritance relationship in the same inheritance tree should be treated as identical ones.

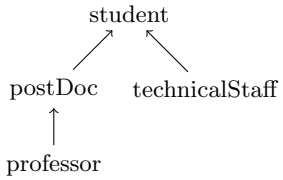

Fig. 4: Inheritance tree for *Role* element

Similarity Score for Numerical Elements. The calculation for numerical elements is more complex because numerical attribute values may have different forms such as single value, set, bounded interval and unbounded interval. Here we propose a unified method defined in Algorithm 1 for computing the similarity score between two numerical elements. The algorithm takes two numerical elements as input. Firstly, if two elements have the same attribute name, operator(s) and attribute value(s), the score is 1 (lines 1,2). Secondly, the two elements should be checked if their intersection is empty. An empty intersection returns 0 as similarity score (lines 4,5). Otherwise, there are three cases:

- **Bounded interval (lines 7,8)**: Two elements' values are both bounded interval. Length of an interval equals the difference between its endpoints. To compute the score, we divide the length of the conjunction of two intervals by the length of their disjunction. For example, the score for time elements in r_{12} and r_{23} is: $S_n(r\_12(time), r\_23(time)) = Len(21 - 19)/Len(22 - 19) = 0.67$.

Algorithm 1 $S_n(e_1, e_2)$: numerical similarity score calculation
Input: two numerical elements e_1 and e_2
Output: numerical similarity score

```
1: if e1 = e2 then
2:    return 1
3: end if
4: if e1 ∩ e2 = φ then
5:    return 0
6: else
7:    if both e1 and e2 are bounded intervals then
8:       return Len(e1∩e2) / Len(e1∪e2)
9:    else if both e1 and e2 are sets then
10:      return Num(e1∩e2) / Num(e1∪e2)
11:   else
12:      return 0.5
13:   end if
14: end if
```

- **Set (lines 9,10)**: Two elements' values are both sets. To compute the score, we divide the cardinality of the conjunction of two sets by the cardinality of their disjunction. For example, $Time_1 = [3\text{ am}, 4\text{ am}, 5\text{ am}]$, $Time_2 = [4\text{ am}, 5\text{ am}, 6\text{ am}]$, $S_n(Time_1, Time_2) = 2/4 = 0.5$.
- **Other cases**: As calculation between two different forms is difficult, we assign a fuzzy value 0.5 as the similarity score. 0.5 is chosen because it is the average value of similarity score.

3.4 Example of Calculation

Here we present an exhaustive example to illustrate how the PSM works. Continuing with the three policies p_1, p_2, p_3 defined in section 3.1 and their abstraction trees introduced in section 3.3, we illustrate the four steps of calculation.

1. **Policy transformation:** Shown in Section 3.1, the three policies have already been transformed from XACML policies to rules composed of atomic elements.
2. **Rule pair calculation:** Applying Equation (3), (6) and Algorithm 1, we calculate scores for different rule pairs in each decision effect:

$$
\begin{aligned}
&Permit: \\
&\qquad S_{rule}(r_{11}, r_{21}) = 0.625 \times 1 \times 1 \times 1 = 0.625 \\
&\qquad S_{rule}(r_{11}, r_{22}) = 0.125 \times 1 \times 1 \times 1 = 0.125 \\
&Deny: \\
&\qquad S_{rule}(r_{12}, r_{23}) = 0.25 \times 1 \times 0.5 \times 0.67 = 0.084 \\
&\qquad S_{rule}(r_{12}, r_{24}) = 0.5 \times 0 \times 0.5 \times 1 = 0
\end{aligned}
$$

3. **Decision effect calculation:** By Equation (4), scores of each decision effect are:
$$S_{permit} = S_{rule}(r_{11}, r_{21}) + S_{rule}(r_{11}, r_{22}) = 0.75$$
$$S_{deny} = S_{rule}(r_{12}, r_{23}) + S_{rule}(r_{12}, r_{24}) = 0.084$$
4. **Total score calculation:** The final score between two policies is calculated by Equation (5):

$$S_{policy}(p_1, p_2) = \frac{S_{permit} + S_{deny}}{Num(permit) + Num(deny)} = \frac{0.75 + 0.084}{2 + 2} = 0.209$$

Applying the same process, we can also calculate the similarity score between policies p_1 and p_3: $S_{policy}(p_1, p_3) = 0.083$. The two scores $S_{policy}(p_1, p_2)$ and $S_{policy}(p_1, p_3)$ indicates that policy p_1 is more similar to p_2 than p_3 in terms of the percentage of rule pairs having the same decision.

4 Experimental Results

In order to verify if our algorithm is applicable to real cases, we compare the percentage of the same decision pairs with the PSM score. Firstly, we implement a random policy generator which takes policy elements as input then generates access control policies in Form 2. Secondly, we extract policy elements from four policies with different models and each of them is related to a real scenario: RBAC for project management [21], Net-RBAC for firewall configuration [22], OrBAC for hospital management [23], ABAC for administration of research laboratory [10]. Thirdly, these policy elements are inputted to the policy generator and each policy pair generated obtains a similarity score by our algorithm. Finally, we input various combinations of elements as access control requests into the four policies and count the percentage of the same decision pair between rules from output. We mention that the test method which we used are brute-force based: for categorical element, we take all the combination of *string* values; for numerical element, enumerating all the numerical based attribute value in an interval (For example $19:00 \leq Time \leq 21:00$) is impossible. Without loss of generalization, we make equidistant sampling for bounded interval and bilateral sampling for unbounded interval. For example, inputs are all the integers from 1 to 24 for $0:00 \leq Time \leq 24:00$; for $FileSize > 10$ MB, inputs are $FileSize = 9$ MB and $FileSize = 11$ MB.

Table 1: Policies tested

Policy	Model	Categorical element	Numerical element	Effect
project-admin	RBAC	15	0	permit
firewall-admin	Net-RBAC	4	28	permit
hospital-admin	OrBAC	15	6	permit,deny
lab-admin	ABAC	19	0	permit,deny

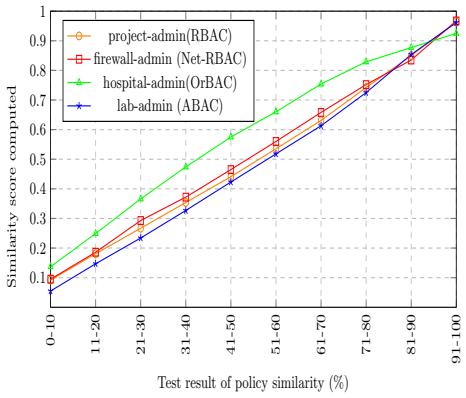

Fig. 5: Experiment of similarity score (*set*-4).

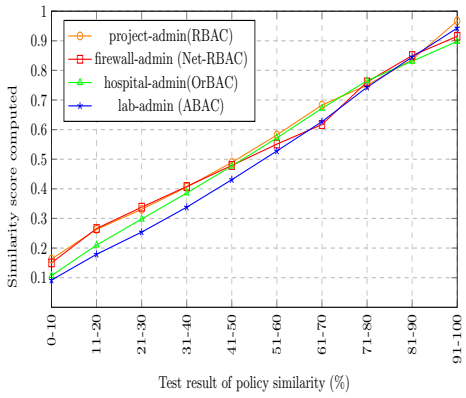

Fig. 6: Experiment of similarity score (*set*-8).

Figure 5, 6 show the policy similarity score (y-axis) and the same decision percentage for rule pairs (x-axis) in *set*-4 and *set*-8. Each test set contains 1000 pairs of policies. In *set*-4, each policy has four rules and in *set*-8 each policy has eight rules. The configurations of elements for each policy model are shown in Table 1. For example, laboratory administration policies are written by ABAC model and these policies contain 19 categorical elements with permit and deny effects. We observe that the score increases when the similarity between two policies increases. At the same time, the experimental values approach to the scores calculated and the quantity of test rules has no impact on the variation of curves. These data enable us to conclude that the PSM score well approximates the similarity between policies.

5 Application

Our PSM algorithm can be applied to different SP selection use cases such as network configuration, compute allocation and cloud storage. This section presents a concrete scenario.

5.1 Scenario Description

SUPERCLOUD [24] is a European project which aims to support user-centric deployments across multi-clouds and enables the composition of innovative trustworthy services. SUPERCLOUD will build a security management architecture and infrastructure to fulfill the vision of user-centric secure and dependable clouds of clouds. One use case is to build a middle-ware layer between cloud customer and cloud SPs and this middle-ware could select SP(s) according to the security requirement of client. Here we implement a scenario of cloud storage.

The subjects involved in the scenario are cloud client, cloud broker and SP. A cloud client wants to use the cloud storage service(s) provided by one or multiple SPs. At the same time, the client wishes that the security policies of SP meet his requirement. Otherwise, he may launch a negotiation process with SP(s) whose security policies are most approximate. To this end, the client chooses the SUPERCLOUD solution. It is worth noting that discovering SP(s) with client's similar security level is just a pre-selection phase. Other criteria such as price and performance will be taken into consideration in the final negotiation and decision steps.

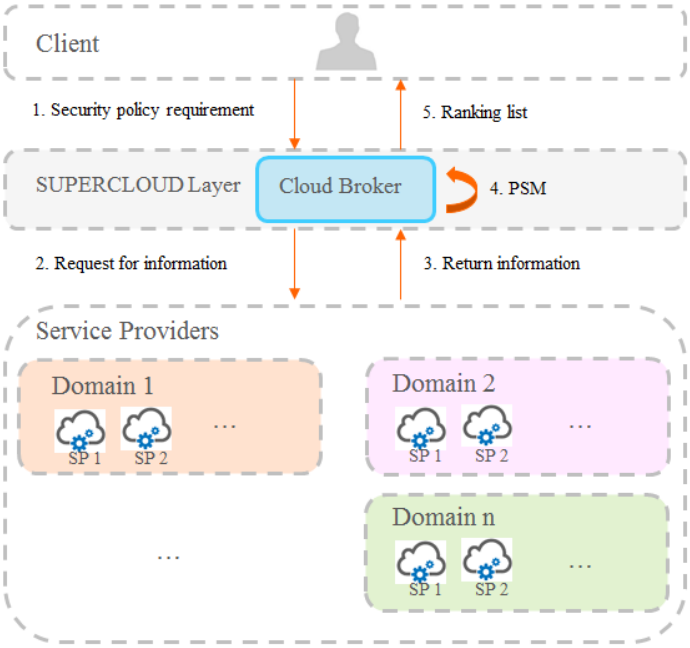

Fig. 7: Service provider selection for cloud storage

The implementation is based on CloudSim [22] simulation framework. Figure 7 illustrates the architecture of our implementation. Firstly, client expresses his requirement on cloud storage by security policies. For example, client may wish that he could have a space of 100 GB and he is allowed to upload files between 8:00 and 22:00. Then the client sends his requirement to the SUPERCLOUD layer where a cloud broker is deployed. The cloud broker obtains the information and security policy templates from SPs. Applying our PSM algorithm, the broker proposes a ranking list of SPs which meet client's requirement from storage space to security policies. PSM scores from SPs are ranked from high to low. When one SP's storage space is less than the requirement, broker

may also propose a composition of two SPs in the same domain[3]. In this case, two SPs' security policies should be combined and the policy after composition is also calculated by PSM and ranked. The composition operation depends on concrete use cases. Here we apply *Conjunction* (&) operation proposed in [25] for cloud storage policies. Consequently, there is more storage space and the security policy is stricter after composition. An example is as follows:

$\mathbf{SP_1}$: 50 $GB, Permit(Action : [upload, download], 8:00 \leq Time \leq 23:00)$

$\mathbf{SP_2}$: 50 $GB, Permit(Action : [upload, download, delete], 7:00 \leq Time \leq 22:00)$

$\mathbf{SP_1}$&$\mathbf{SP_2}$: 100 $GB, Permit(Action : [upload, download], 8:00 \leq Time \leq 22:00)$

5.2 Performance

The implementation is programmed in JAVA and is executed on an Intel machine having configuration: 2.2 GHz with 4 GB of RAM running Windows 8 and JDK 1.8. We measure the execution time needed until the client receives a SP ranking list. Figure 8 shows the execution time with the increase of SP quantity from 0 to 100 in each domain and there exist five domains. Blue line with triangles presents the execution time with the PSM and red line with stars shows the execution time without the PSM. In Figure 9, the domain number varies from 5 to 30. The higher surface presents the execution time with the PSM and the lower surface shows the execution time without the PSM. From the two figures, we remark that the introduction of the PSM does not cause much of performance loss and it proves that our PSM algorithm is light-weight.

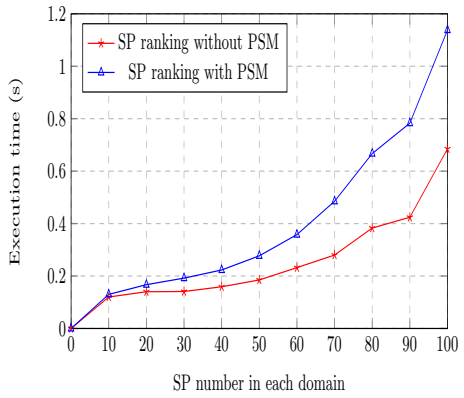

Fig. 8: Execution time of SP ranking (domain number=5)

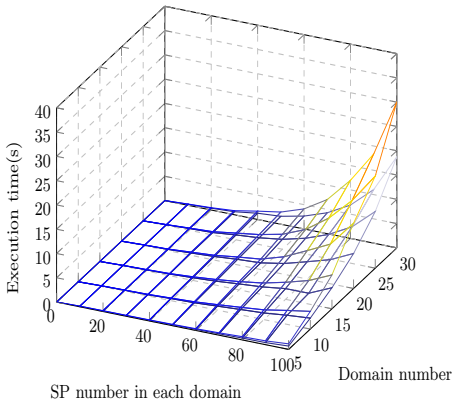

Fig. 9: Execution time of SP ranking (domain number=5∼30)

[3] We suppose that SPs in a cloud federation share the same domain and two SPs in the same domain can be composed as a virtual SP.

6 Conclusion and Future Work

The main objective of this paper is to expose our proposition to show how to measure the similarity between two security policies. The proposition gives mainly a generic and light-weight algorithm with which we can calculate a similarity score between two access control policies. After introducing the categorical measure and numerical measure, output of our algorithm approximates to the test result. In addition, our algorithm can be applied on policies with different models such as ABAC, RBAC and OrBAC.

We are planning to extend our work along the following directions. The first direction is related to policy negotiation between SP and client in a real distributed environment such as Grid'5000 [26]. The similarity evaluation may serve as a filter step to find out the SPs with similar security level. The second direction is to integrate our algorithm in some security policy negotiation frameworks [27]. The similarity score will be helpful in the negotiation process such as counter offer generation and decision making.

Acknowledgments. The work reported in this paper has been supported by ANRT [4] and Orange as CIFRE[5] thesis and the work of Nora Cuppens-Boulahia and Frédéric Cuppens has been partially carried out in the SUPERCLOUD project, funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 643964.

References

1. Li, A., Yang, X., Kandula, S., Zhang, M.: Cloudcmp: comparing public cloud providers. In: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, ACM (2010) 1–14
2. Yau, S.S., Yin, Y.: Qos-based service ranking and selection for service-based systems. In: Services Computing (SCC), 2011 IEEE International Conference on, IEEE (2011) 56–63
3. Luna, J., Ghani, H., Germanus, D., Suri, N.: A security metrics framework for the cloud. In: Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on, IEEE (2011) 245–250
4. Taha, A., Trapero, R., Luna, J., Suri, N.: Ahp-based quantitative approach for assessing and comparing cloud security. In: Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on, IEEE (2014) 284–291
5. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **29**(2) (1996) 38–47
6. Kalam, A.A.E., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., Trouessin, G.: Organization based access control. In: Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on, IEEE (2003) 120–131

[4] Association Nationale de la Recherche et de la Technologie
[5] Conventions Industrielles de Formation par la REcherche

7. Yuan, E., Tong, J.: Attributed based access control (abac) for web services. In: Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on, IEEE (2005)
8. Standard, O.: extensible access control markup language (xacml) version 2.0 (2005)
9. Lin, D., Rao, P., Bertino, E., Lobo, J.: An approach to evaluate policy similarity. In: Proceedings of the 12th ACM symposium on Access control models and technologies, ACM (2007) 1–10
10. Lin, D., Rao, P., Ferrini, R., Bertino, E., Lobo, J.: A similarity measure for comparing xacml policies. Knowledge and Data Engineering, IEEE Transactions on **25**(9) (2013) 1946–1959
11. Bei, W., Xing-yuan, C., Yong-fu, Z.: A policy rule dissimilarity evaluation approach based on fuzzy theory. In: Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on, IEEE (2009) 1–6
12. Pham, Q., Reid, J., Dawson, E.: Policy filtering with xacml. (2011)
13. Lin, D., Squicciarini, A.: Data protection models for service provisioning in the cloud. In: Proceedings of the 15th ACM symposium on Access control models and technologies, ACM (2010) 183–192
14. Cho, E., Ghinita, G., Bertino, E.: Privacy-preserving similarity measurement for access control policies. In: Proceedings of the 6th ACM workshop on Digital identity management, ACM (2010) 3–12
15. Shaikh, R.A., Sasikumar, M.: Dynamic parameter for selecting a cloud service. In: Computation of Power, Energy, Information and Communication (ICCPEIC), 2014 International Conference on, IEEE (2014) 32–35
16. Bertolino, A., Daoudagh, S., El Kateb, D., Henard, C., Le Traon, Y., Lonetti, F., Marchetti, E., Mouelhi, T., Papadakis, M.: Similarity testing for access control. Information and Software Technology **58** (2015) 355–372
17. Agrawal, D., Giles, J., Lee, K.W., Lobo, J.: Policy ratification. In: Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on, IEEE (2005) 223–232
18. Lin, D., Rao, P., Bertino, E., Li, N., Lobo, J.: Exam: a comprehensive environment for the analysis of access control policies. International Journal of Information Security **9**(4) (2010) 253–273
19. Jaccard, P.: Nouvelles recherches sur la distribution florale. (1908)
20. Cuppens, F., Cuppens-Boulahia, N., Miège, A.: Inheritance hierarchies in the or-bac model and application in a network environment. Proc. Foundations of Computer Security (FCS04) (2004) 41–60
21. `http://docs.openstack.org/developer/keystone/configuration.html`
22. Hachana, S., Cuppens-Boulahia, N., Cuppens, F.: Mining a high level access control policy in a network with multiple firewalls. Journal of Information Security and Applications **20** (2015) 61–73
23. Autrel, F., Cuppens, F., Cuppens-Boulahia, N., Coma, C.: Motorbac 2: a security policy tool. In: 3rd Conference on Security in Network Architectures and Information Systems (SAR-SSI 2008), Loctudy, France. (2008) 273–288
24. `http://www.supercloud-project.eu`
25. Bonatti, P., De Capitani di Vimercati, S., Samarati, P.: An algebra for composing access control policies. ACM Transactions on Information and System Security (TISSEC) **5**(1) (2002) 1–35
26. Bolze, R., Cappello, F., Caron, E., Daydé, M., Desprez, F., Jeannot, E., Jégou, Y., Lanteri, S., Leduc, J., Melab, N., et al.: Grid'5000: a large scale and highly reconfigurable experimental grid testbed. International Journal of High Performance Computing Applications **20**(4) (2006) 481–494

27. Li, Y., Cuppens-Boulahia, N., Crom, J.M., Cuppens, F., Frey, V.: Reaching agreement in security policy negotiation. In: Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on, IEEE (2014) 98–105

Appendix

A Brute-force based test for existing work

Figure 10 shows the brute-force test result of policy similarity score by using the same test environment illustrated in Section 4. The y-axis represents the PSM score computed by the algorithm proposed in [10]; the x-axis shows the test result of policy similarity defined by Equation (7) [10], where $Sreq$ denotes the set of the requests with the same decisions from p_1 and p_2 and Req is the set of the requests applicable to either p_1 or p_2:

$$S_{policy}(p_1, p_2) = |Sreq|/|Req| \tag{7}$$

We remark that the similarity score computed does not approximate to the test result. The main reason is that, firstly, as a brute-force based test method, our input requests are more exhaustive than ones generated by other test tools such as MTBDD [18]. Secondly, the PSM algorithm defined in [10] focuses only on the literal level but not logic aspect of security policy. As a result, two security rules sharing the majority of common elements are considered to hold a higher similarity score. However, the rest of elements may cause totally different decisions which indicates that the two rules are not similar in terms of output.

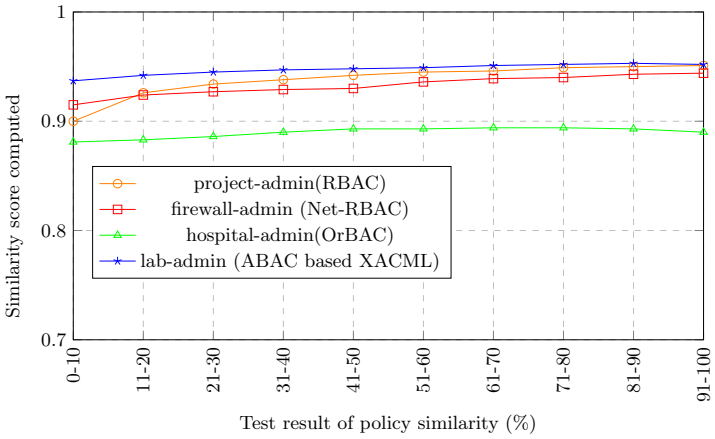

Fig. 10: Experiment of similarity score (set-4).