An Approach to Evaluate Policy Similarity

Dan Lin[†] Prathima Rao[†] Elisa Bertino[†] Jorge Lobo[‡]

[†]Department of Computer Science
Purdue University, USA
{*lindan,prao,bertino*}*@cs.purdue.edu*

[‡]IBM T.J. Watson Research Center
USA
jlobo@us.ibm.com

ABSTRACT

Recent collaborative applications and enterprises very often need to efficiently integrate their access control policies. An important step in policy integration is to analyze the similarity of policies. Existing approaches to policy similarity analysis are mainly based on logical reasoning and boolean function comparison. Such approaches are computationally expensive and do not scale well for large heterogeneous distributed environments (like Grid computing systems). In this paper, we propose a policy similarity measure as a filter phase for policy similarity analysis. This measure provides a lightweight approach to pre-compile a large amount of policies and only return the most similar policies for further evaluation. In the paper we formally define the measure, by taking into account both the case of categorical attributes and numeric attributes. Detailed algorithms are presented for the similarly computation. Results of our case study demonstrates the efficiency and practical value of our approach.

Categories and Subject Descriptors

D.4.6 Security and Protection—Access controls

General Terms

Algorithms, Measurement,Design

Keywords

Policy similarity measure, XACML policies, Access control policies

1. INTRODUCTION

A key goal for collaborative applications is to share resources, such as services, data and knowledge. Such applications may have different objectives, such as provisioning some complex service to third parties or performing some collaborative data analysis, and may adopt different collaboration mechanisms and tools. However, a common requirement is represented by the need to assure security for the shared resources. It is important that collaboration does not undermine security of the collaborating parties and their resources. Security however should not drastically reduce the benefits deriving from the collaboration by severely restricting the access to the resources by the collaborators. An important question that a party P thus may need to answer when deciding whether to share a resource with other parties is whether these other parties guarantee the similar level of security as P. This is a complex question and approaches to it require developing adequate methodologies and processes, and addressing several issues. One relevant issue is the comparison of access control policies; access control policies govern access to protected resources by stating which subjects can access which data for which operations and under which circumstances. Access control represents a key component of any security mechanism. A party P may decide to release some data to a party P' only if the access control policies of P' are very much the same as its own access control policies. It is important to notice that an approach under which P just sends its policies together with the data to P' so that P' can directly enforce these policies may not always work. The evaluation of P's policies may require accessing some additional data that may not be available to P' for various reasons, for example for confidentiality, or P may not just be able to share, for confidentiality reasons, its policies with P'.

More complex situations arise when several alternative resources and services, each governed by its own independently-administered access control policies, may have to be selected and combined in a complex service. In order to maximize the number of requests that can be satisfied by the complex service, and also satisfy the access control policies of each participating resource and service, one would like to select a combination of resources and services characterized by access control policies that are very much similar. As an example consider the case of a grid computing system consisting of data owners and resource owners, each with its own access control policies [10]. For a subject to be able to run a query on the data, this subject must verify both the access control policy associated with the queried data and the access control policy of the resource to be used to process the query. It is often the case that such parties do not have exactly the same access control policies; therefore in order to maximize the access to the data, the data for processing should be stored at the resource having access control poli-

SACMAT'07, June 20-22, 2007, Sophia Antipolis, France.

cies similar to the access control policies associated with the data.

A brute force approach is to simply evaluate both policies for any request and any assignment, and then compare the results. Obviously, such an approach is very inefficient and even infeasible when the request domain is infinite. Therefore, two recent approaches have been proposed to address the problem of policy comparison. The first approach is based on model checking [5], whereas the second is based on logical reasoning [1]. However, both these approaches are computationally expensive, especially when dealing with large scale heterogeneous distributed environments and time constraints. The problem is equivalent to solving Boolean satisfiability, which is NP-complete. In this paper, we thus propose an alternative approach, based on principles from the information retrieval field. Our approach uses the notion of *policy similarity measure*, based on which a similarity score can be quickly computed for two policies. Specifically, if the similarity score of policy P_1 and P_2 is higher than that of policy P_1 and P_3, it means P_1 and P_2 may yield same decisions to a larger common request set than P_1 and P_3 will do. The policy similarity measure can serve as a filter before applying any additional logical reasoning or boolean function comparison. It provides a useful lightweight approach to pre-compile a list of policies and return the most similar policies for further exploration. Such exploration could be a fine-grained policy analysis which spots the common and different parts of two policies; it can also be a visualization phase where users can easily identify the similar policies and make their own decisions. Our approach parallels the quick filter approach for multimedia data querying. In multimedia data querying, a first quick filtering of the data is executed; such filtering phase discards the data that are certainly not part of the query reply. The data that are not discarded by the quick filter are then analyzed by using specialized algorithms to determine the actual query results.

In the current paper, we focus on a similarity measure for policies written in XACML (Extensible Access Control Mark-up Language) [15], because of its widespread adoption. The similarity measure takes into account the policy structure typical of XACML. Given two polices, our algorithm for computing the similarity score first groups the same components of the two policies, and evaluate their similarity by using hierarchy distance and numerical distance. Then the scores obtained for the different components of the policies are combined according to a weighted combination in order to produce an overall similarity score. As our case study shows, our approach can successfully identify similar policies.

The rest of the paper is organized as follows. Section 2 introduces preliminary notions concerning XACML. Section 3 introduces the proposed policy similarity measure, the algorithm for computing the similarity, and a case study. Section 4 discusses related works on access control policy analysis. Finally, Section 5 concludes the paper and outlines future work.

2. PRELIMINARY NOTIONS AND AN ILLUSTRATIVE EXAMPLE

An XACML policy [15] consists of three major components, namely a *Target*, a *Rule set*, and a *rule combining algorithm* for conflict resolution. The *Target* specifies some

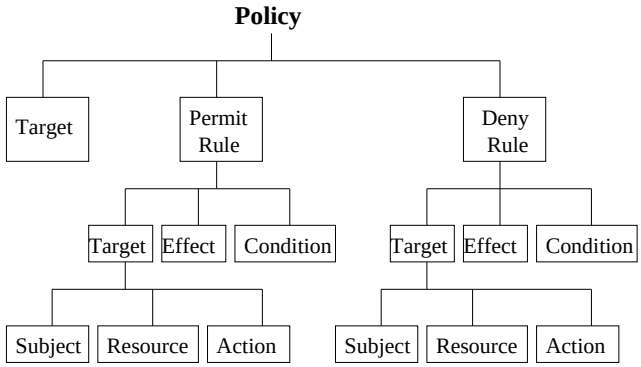

Figure 1: Policy Structure

predicates on the attribute values in a request, which must be held in order for the policy to be applicable to the request. The attributes in the *Target* element are categorized into *Subject*, *Resource* and *Action*. A *Rule set* consists of one or more *Rules*. Each *Rule* in turn consists of *Target*, *Condition* and *Effect* elements. The rule *Target* has the same structure of the policy *Target*. The only difference is that the rule *Target* specifies the situation when the rule can be applied. A *Condition* element specifies some restrictions on request attribute values that must be satisfied in order to yield a *Permit* or *Deny* decision as specified by the *Effect* element. Figure 1 gives an overview of a policy structure.

As an example that we will use throughout the paper, we consider three policies P_1, P_2 and P_3, in the context of data and resource management for a grid computing system in a university domain. In particular, P_1 is a data owner policy, whereas P_2 and P_3 are resource owner policies. Specifi-

PolicyId=P1
<PolicyTarget
GroupName=IBMOpenCollaboration>
<RuleId=R11 Effect=Permit>
<Target>
<Subject Designation *belong_to*{Professor, PostDoc, Student, TechnicalStaff} >
<Resource FileType *belong_to*{Source, Documentation, Executable} >
<Action AccessType *belong_to*{Read, Write} >
</Target>
<Condition FileSize $\leq$ 100MB >
</Rule>
<RuleId=R12 Effect=Deny>
<Target>
<Subject Designation *belong_to*{Student, PostDoc, TechnicalStaff} >
<Resource FileType *belong_to*{Source, Documentation, Executable}>
<Action AccessType=Write>
</Target>
<Condition $19:00 \leq$ Time $\leq 21:00$>
</Rule>

Figure 2: Data Owner Policy P_1

PolicyId=P2
<RuleId=R21 Effect=Permit>
<PolicyTarget GroupName *belong_to*{IBMOpen-Collaboration, IntelOpenCollaboration}>
<Target>
<Subject Designation *belong_to*{Student, Faculty, TechnicalStaff} >
<Action AccessType *belong_to*{Read, Write}>
</Target>
<Condition FileSize $\leq$ 120MB >
</Rule>
<RuleId=R22 Effect=Permit>
<Target>
<Subject Designation=TechnicalStaff>
<Action AccessType *belong_to*{Read, Write}>
</Target>
<Condition $19:00 \leq$ Time $\leq 22:00$ >
</Rule>
<RuleId=R23 Effect=Deny>
<Target>
<Subject Designation=Student>
<Action AccessType=Write>
</Target>
<Condition {$19:00 \leq$ Time $\leq 22:00$>
</Rule>
<RuleId=R24 Effect=Deny>
<Target>
<Subject Designation *belong_to*{Student, Faculty, Staff}>
<Resource FileType=Media>
<Action AccessType *belong_to*{Read, Write}>
</Target>
</Rule>

Figure 3: Resource Owner Policy P_2

cally, P_1 states that professors, postdocs, students and technical staff in the IBM project group are allowed to read or write source, documentation or executable files of size less than 100MB. P_1 denies the write operations for postdocs, students and technical staff between 19:00 and 21:00 because professors may want to check and make changes to the project files without any distraction. P_2 is an access control policy for a project machine. P_2 allows students, faculty and technical staff in the IBM or Intel project group to read or write files of size less than 120MB. P_2 gives a special permission to technical staff between time 19:00 and 22:00 so that technical staff can carry out system maintenance and backup files, and denies students the permission to write any file when technical staff is possibly working on maintenance. Moreover, P_2 does not allow any user to operate on media files on the machine. P_3 is an access control policy for another machine, mainly used by business staff. P_3 states that only business staff in the group named "Payroll" can read or write .xls files of size less than 10MB from 8:00 to 17:00, and it clearly denies students the access to the machine. Figure 2, 3 and 4 report the XACML specification for these policies.

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

PolicyId=P3
<PolicyTarget GroupName = Payroll >
<RuleId=R31 Effect=Permit>
<Target>
<Subject Designation=BusinessStaff>
<Resource FileType=".xls">
<Action AccessType *belong_to*{Read, Write}>
</Target>
<Condition $8:00 \leq$ Time $\leq 17:00$, FileSize $\leq$ 10MB >
</Rule>
<RuleId=R32 Effect=Deny>
<Target>
<Subject Designation=Student>
<Action AccessType *belong_to*{Read, Write}>
</Target>
</Rule>

Figure 4: Resource Owner Policy P_3

similarity scores S_1 between P_1 and P_2, and S_2 between P_1 and P_3, where we would expect that S_1 be larger than S_2 to indicate that the similarity between P_1 and P_2 is much higher than the similarity between P_1 and P_3.

3. POLICY SIMILARITY MEASURE

Our proposed policy similarity measure is based on the comparison of each corresponding component of the policies being compared. Here, the corresponding component means the policy targets and the same type of elements belonging to the rules with the same effect.

We would like the policy similarity measure between any two given policies to assign a *similarity score* that **approximates** the relationship between the sets of requests permitted (denied) by the two policies. The similarity score is a value between 0 and 1, which reflects how similar these rules are with respect to the targets they are applicable to and also with respect to the conditions they impose on the requests. For example, in a scenario where a set of requests permitted (denied) by a policy P_1 is a subset of requests permitted (denied) by a policy P_2, the similarity score for policies P_1 and P_2 must be higher than the score assigned in a scenario in which the set of requests permitted (denied) by P_1 and P_3 have very few or no request in common.

We now proceed to introduce how to obtain the similarity score of two policies. Given two policies P_1 and P_2, the rules in these policies are first grouped according to their effects, which results in a set of Permit Rules (denoted as PR) and a set Deny Rules (denoted as DR). Each single rule in P_1 is then compared with a rule in P_2 that has the same effect, and a similarity score of two rules is obtained. The similarity score obtained between the rules is then used to find one-many mappings (denoted as Φ) for each rule in the two policies. For clarity, we choose to use four separate Φ mappings Φ_1^P, Φ_1^D, Φ_2^P and Φ_2^D. The mapping $\Phi_1^P(\Phi_1^D)$ maps each PR(DR) rule r_{1i} in P_1 with one or more PR(DR) rules r_{2j} in P_2. Similarly the mapping $\Phi_2^P(\Phi_2^D)$ maps each PR(DR) rule r_{2j} in P_2 with one or more PR(DR) rules r_{1i} in P_1. For each rule in a policy $P_1(P_2)$, the Φ mappings give similar rules in $P_2(P_1)$ which satisfy certain similarity

threshold. The computation of the Φ mapping will be addressed in the Section 3.1.

By using the Φ mappings, we compute the similarity score between a rule and a policy. We aim to find out how similar a rule is with respect to the entire policy by comparing the single rule in one policy with a set of similar rules in the other policy. The notation $rs_{1i}(rs_{2j})$ denotes the similarity score for a rule $r_{1i}(r_{2j})$ in policy $P_1(P_2)$. The rule similarity score $rs_{1i}(rs_{2j})$ is the average of the similarity scores between a rule $r_{1i}(r_{2j})$ and the rules similar to it given by the Φ mapping. rs_{1i} and rs_{2j} are computed according to the following expressions:

$$rs_{1i} = \begin{cases} \dfrac{\sum_{r_j \in \Phi_1^P(r_{1i})} S_{rule}(r_{1i}, r_j)}{|\Phi_1^P(r_{1i})|}, r_{1i} \in PR_1 \\ \dfrac{\sum_{r_j \in \Phi_1^D(r_{1i})} S_{rule}(r_{1i}, r_j)}{|\Phi_1^D(r_{1i})|}, r_{1i} \in DR_1 \end{cases} \quad (1)$$

$$rs_{2j} = \begin{cases} \dfrac{\sum_{r_i \in \Phi_2^P(r_{2j})} S_{rule}(r_{2j}, r_i)}{|\Phi_2^P(r_{2j})|}, r_{2j} \in PR_2 \\ \dfrac{\sum_{r_i \in \Phi_2^D(r_{2j})} S_{rule}(r_{2j}, r_i)}{|\Phi_2^D(r_{2j})|}, r_{2j} \in DR_2 \end{cases} \quad (2)$$

where S_{rule} is a function that assigns a similarity score between two rules.

Next, we compute the similarity score between the permit(deny) rule sets $PR_1(DR_1)$ and $PR_2(DR_2)$ of policies P_1 and P_2 respectively. We use the notations $S^P_{rule-set}$ and $S^D_{rule-set}$ to denote the similarity scores for permit and deny rule sets respectively. The similarity score for a permit(deny) rule set is obtained by averaging the rule similarity scores (equations 1 and 2) for all rules in the set. The permit and deny rule set similarity scores are formulated as follows:

$$S^P_{rule-set} = \frac{\sum_{i=1}^{N_{PR_1}} rs_{1i} + \sum_{i=1}^{N_{PR_2}} rs_{2j}}{N_{PR_1} + N_{PR_2}} \quad (3)$$

$$S^D_{rule-set} = \frac{\sum_{i=1}^{N_{DR_1}} rs_{1i} + \sum_{i=1}^{N_{DR_2}} rs_{2j}}{N_{DR_1} + N_{DR_2}} \quad (4)$$

where N_{PR_1} and N_{PR_2} are the numbers of rules in PR_1 and PR_2 respectively, N_{DR_1} and N_{DR_2} are the numbers of rules in DR_1 and DR_2 respectively.

Finally, we combine the similarity scores for permit and deny rule sets between the two policies along with a similarity score between the *Target* elements of the two policies, to develop an overall similarity score, S_{policy}. The formulation of S_{policy} is given by the following equation:

$$S_{policy}(P_1, P_2) = w_T S_T(P_1, P_2) + w_p S^P_{rule-set} + w_d S^D_{rule-set} \quad (5)$$

where S_T is a function that computes a similarity score between the *Target* elements of any two given policies; w_p and w_d are weights that can be chosen to reflect the relative importance to be given to the similarity of permit and deny rule sets respectively. For normalization purpose, the weight values should satisfy the constraint: $w_T + w_p + w_d = 1$. An example is given in Section 3.5 (refer to step 5–9).

Notation	Meaning
P	Policy
PR	Permit rule set
DR	Deny rule set
r	Rule
a	Attribute
v	Attribute value
H	Height of a hierarchy
S_{policy}	Similarity score of two policies
S_{rule}	Similarity score of two rules
$S^P_{rule-set}$	Similarity score of two sets of permit rules
$S^D_{rule-set}$	Similarity score of two sets of deny rules
$S_{\langle Element \rangle}$	Similarity score of elements, $\langle Element \rangle \in \{'T','t','c','s','r','a'\}$
s_{cat}	Similarity score of two categorical values
S_{cat}	Similarity score of two categorical predicates
s_{num}	Similarity score of two numerical values
S_{num}	Similarity score of two numerical predicates
rs	Similarity score between a rule and a policy
Φ	Rule mapping
$\mathcal{M}_a$	Set of pairs of matching attribute names
$\mathcal{M}_v$	Set of pairs of matching attribute values
N_{PR}	Number of permit rules in a policy
N_{DR}	Number of deny rules in a policy
N_a	Number of attributes in an element
N_v	Number of values of an attribute
$SPath$	Length of shortest path of two categorical values
$w_{\langle Element \rangle}$	Weight of similarity scores of elements, $\langle Element \rangle \in \{'T','t','c','s','r','a'\}$
ϵ	Rule similarity threshold
δ	Compensating score for unmatched values

Table 1: Notations

The intuition behind the similarity score assigned to any two policies is derived from the fact that two policies are similar to one another when the corresponding policy elements are similar.

In the following sections, we introduce the detailed algorithms for the computation of Φ mappings and rule similarity score S_{rule}. Table 1 lists main notations used throughout the paper.

3.1 Computation of Φ Mappings

In this section, we discuss the procedure to determine the Φ mappings for each rule in the permit and deny rule sets in a policy.

The one-many Φ mappings determine for each PR(DR) rule in $P_1(P_2)$ which PR(DR) rules in $P_2(P_1)$ are *very similar*. Intuitively, two rules are *similar* when their targets and the conditions they specify are similar. Thus we define a Φ mapping as follows:

$$\Phi(r_i) = \{r_j | S_{rule}(r_i, r_j) \geq \epsilon\} \quad (6)$$

where S_{rule} is computed by equation 7 and ϵ is a threshold. The threshold term is important here, since it allows us to calibrate the quality of the similarity approximation. We expect that the actual value of the threshold will be very specific to the policy domain. Figure 5 summarizes the procedure for calculating a Φ mapping. This procedure takes two rule sets R' and R'' as inputs and computes a mapping

Procedure ComputePhiMapping(R', R'', ϵ)
Input : R' and R'' are sets of rules and ϵ is a *threshold* value.
1. **foreach** rule $r' \in R'$
2. $\quad \Phi(r') = \emptyset$
3. $\quad$ **foreach** rule $r'' \in R''$
4. $\quad\quad$ **if** $S_{rule}(r', r'') \geq \epsilon$ **then**
5. $\quad\quad\quad \Phi(r') = \Phi(r') \cup \{r''\}$
6. **return** Φ

end ComputePhiMapping.

Figure 5: Procedure for computing a Φ mapping

for each rule in R' based on equation 6. An example of the computation of Φ mapping is shown by steps 3 and 4 in Section 3.5.

3.2 Similarity Score between Rules

Since our similarity measure serves as a lightweight filter phase, we do not want to involve complicated analysis for boolean expressions. Our similarity measure is developed based on the intuition that rules r_i and r_j are similar when both apply to similar targets and both specify similar conditions on request attributes. Specifically, we compute the rule similarity function S_{rule} between two rules r_i and r_j as follows:

$$S_{rule}(r_i, r_j) = w_t S_t(r_i, r_j) + w_c S_c(r_i, r_j) \quad (7)$$

w_t and w_c are weights that can be used for emphasizing the importance of the target or condition similarity respectively. For example, if users are more interested in finding policies applied to similar targets, they can increase w_t to achieve this purpose. The weights satisfy the constraint $w_t + w_c = 1$. S_t and S_c are functions that compute a similarity score between two rules based on the comparison of their *Target* and *Condition* elements respectively.

As the *Target* element in each rule contains the *Subject*, *Resource* and *Action* elements, each of these elements in turn contains predicates on the respective category of attributes. Thus, the *Target* similarity function S_t is computed as follows:

$$S_t(r_i, r_j) = w_s S_s(r_i, r_j) + w_r S_r(r_i, r_j) + w_a S_a(r_i, r_j) \quad (8)$$

In equation 8, w_s, w_r, w_a represent weights that are assigned to the corresponding similarity scores. Like in the previous equations, weight values need to satisfy the constraint $w_s + w_r + w_a = 1$. S_s, S_r and S_a are functions that return a similarity score based on the *Subject*, *Resource* and *Action* attribute predicates respectively in the *Target* elements of the two given rules.

The computation of functions S_c, S_s, S_r and S_a involves the comparison of pairs of predicates in the given pair of rule elements, which we discuss in detail in the next subsection.

3.3 Similarity Score of Rule Elements

Each of the rule elements *Subject*, *Resource*, *Action* and *Condition* is represented as a set of predicates in the form of $\{attr\_name_1 \oplus_1 attr\_value_1, attr\_name_2 \oplus_2 attr\_value_2, ...\}$, where *attr_name* denotes the attribute name, $\oplus$ denotes a comparison operator and *attr_value* represents an attribute value. We assume that there are no syntactic variations for the same attribute name. For example, there cannot exist attribute names "emp-name", "EmpName" in different policies all of which refer to the employee name attribute. The unification of the attribute names is out of the scope of this paper as many existing approaches [11, 13, 16] developed for schema matching can be adopted to handle this problem.

Based on the type of attribute values, predicates are divided into two categories, namely *categorical predicate* and *numerical predicate*.

- *Categorical predicate*: The attribute values of this type of predicate are categorical data that belong to some domain-specific ontology. Predicates like "Designation = Professor" and "FileType = Documentation" belong to the categorical type.
- *Numerical predicate*: The attribute values of this type of predicate belong to *integer*, *real*, or *date/time* data types. For example, predicates "FileSize < 10MB", "Time=12:00" are of numerical type.

The similarity score between two rules r_i and r_j regarding the same element is denoted as $S_{\langle Element \rangle}$, where $\langle Element \rangle$ refers to 'c' (condition), 's' (subject), 'r' (resource) or 'a' (action). The $S_{\langle Element \rangle}$ is computed by comparing the corresponding predicate sets in two rules. There are three steps. First, we cluster the predicates for each rule element according to the attribute names. It is worth noting that one attribute name may be associated with multiple values. Second, we find the predicates in the two rules whose attribute names match exactly and then proceed to compute a similarity score for their attribute values. The way we compute similarity score between attribute values differs, depending on whether the attribute value is of categorical type or numerical type (details of computation is covered in the following subsection). Finally, we summarize the scores of each pair of matching predicates and obtain the similarity score of the rule element. Since not all attributes in one rule can find a matching in the other, we include a penalty for this case by dividing the sum of similarity scores of matching pairs by the maximum number of attributes in a rule. In addition, there is a special case when the element set is empty in one rule, which means no constraint exists for this element. For this case, we consider the similarity of the elements of the two rules to be 0.5 due to the consideration that one rule is a restriction of the other and the 0.5 is the estimation of the average similarity.

The formal definition of $S_{\langle Element \rangle}$ is given by equation 9.

$$S_{\langle Element \rangle}(r_i, r_j) = \begin{cases} \dfrac{\sum_{(a_{1k}, a_{2l}) \in \mathcal{M}_a} S_{\langle attr\_typ \rangle}(a_{1k}, a_{2l})}{max(N_{a_1}, N_{a_2})}, & N_{a_1} > 0 \text{ and } N_{a_2} > 0; \\ 1, & \text{otherwise.} \end{cases} \quad (9)$$

In equation 9, $\mathcal{M}_a$ is a set of pairs of matching predicates with the same attribute names; a_{1k} and a_{2l} are attributes of rules r_{1i} and r_{2j} respectively; $S_{\langle attr\_typ \rangle}$ is the similarity score of attribute values of the type *attr_typ*; and N_{a_1} and N_{a_2} are the numbers of distinct predicates in the two rules respectively.

In addition, the computation of the similarity score of two policy targets S_T is the same as that for the rule targets i.e. S_t.

3.3.1 Similarity Score for Categorical Predicates

For the categorical values, we not only consider the exact match of two values, but also consider their semantics similarity. For example, policy P_1 is talking about the priority of professors, policy P_2 is talking about faculty members, and policy P_3 is talking about business staff. In some sense, policy P_1 is more similar to policy P_2 than to policy P_3 because "professors" is a subset of "faculty members" which means that policy P_1 could be a restriction of policy P_2. Based on this observation, our approach assumes that a hierarchy relationship exists for the categorical values. The similarity between two categorical values (denoted as S_{cat}) is then defined according to the shortest path of these two values in the hierarchy. The formal definition is shown below:

$$s_{cat}(v_1, v_2) = 1 - \frac{SPath(v_1, v_2)}{2H} \tag{10}$$

where $SPath(v_1, v_2)$ denotes the length of the shortest path between two values v_1 and v_2, and H is the height of the hierarchy. In the equation 10, the length of the shortest path of two values is normalized by the possible maximum path length which is $2H$. The closer the two values are located in the hierarchy, the more similar the two values will be, and hence a higher similarity score s_{cat} will be obtained.

Figure 6 gives an example hierarchy, where each node represents a categorical value (specific values are given in Figure 11). The height of the hierarchy is 3, and the length of maximum path of two values is estimated as $2 \times 3 = 6$ (the actual maximum path in the figure is 5 due to the imbalance of the hierarchy). The $SPath(E, B)$ is 1, and the $SPath(E, F)$ is 2. According to the equation 10, the similarity score of nodes E and B is $1 - 1/6 = 0.83$, and the similarity score of nodes E and F is $1-2/6 = 0.67$. From the obtained scores, we can observe that E is more similar to B than to F. The underlying idea is that the parent-child relationship (B and E) implies one rule could be a restriction for the other and this would be more helpful than the sibling relationship (E and F) especially in the rule integration.

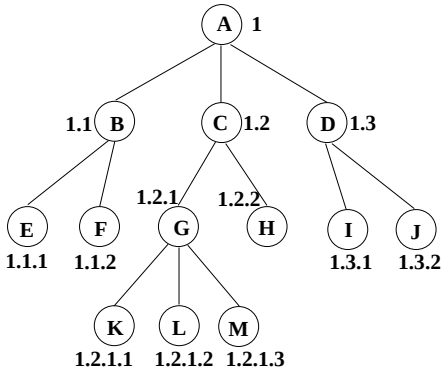

Figure 6: An Example Hierarchy

To avoid repeatedly searching the hierarchy tree for the same value during the shortest path computation, we propose to assign each node a *hierarchy code* (Hcode), indicating the position of each node. In particular, the root node is assigned an Hcode equal to '1', and its children nodes are named in the order from left to right by appending their position to the parent's Hcode with a separator '.', where we will have Hcodes like '1.1' and '1.2'. Then the process continues till the leaf level. The number of elements separated by '.' is equal to the level at which a node is located. From such Hcodes we can easily compute the length of shortest path between two nodes. We compare two Hcodes element by element until we reach the end of one Hcode or there is a difference. The common elements correspond to the same parent nodes they share, and the number of different elements correspond to the levels that they need to be generalized to their common parent node. Therefore, the shortest path is the total number of different elements in two Hcodes. For example, the length of the shortest path from node '1.1' to '1.2' is 2, as there are two different elements in the Hcodes.

Note that our definition of s_{cat} can also be applied to categorical values which do not lie in a hierarchy. In that case, if two values are matched, their shortest path $SPath$ is 0 and their similarity score will be 1; otherwise, $SPath$ is infinity and their similarity score becomes 0.

Having introduced our approach to compare two single values, we now extend the discussion to two sets of values. Suppose there are two attributes $a_1 : \{v_{11}, v_{12}, v_{13}, v_{14}\}$ and $a_2 : \{v_{21}, v_{22}, v_{23}\}$, where a_1 and a_2 are the attribute names belonging to policy P_1 and P_2 respectively, and values in the brackets are corresponding attribute values. Note that the listed values belonging to the same attribute are different from one another. The similarity score of the two attribute value sets is the sum of similarity scores of pairs $\langle v_{1k}, v_{2l} \rangle$ and a compensating score δ (for non-matching attribute values). Obviously, there could be many combinations of pairs. Our task is to find a set of pairs (denoted as $\mathcal{M}_v$) which have the following properties:

1. If $v_{1k} = v_{2l}$, then $(v_{1k}, v_{2l}) \in \mathcal{M}_v$.
2. For pairs $v_{1k} \neq v_{2l}$, pairs contributing to the maximum sum of similarity scores belong to $\mathcal{M}_v$.
3. Each attribute value v_{1k} or v_{2l} occurs at most once in $\mathcal{M}_v$.

The process of finding the pair set $\mathcal{M}_v$ is the following. First, we obtain the hierarchy code for each attribute value. See Figure 7 for an example of these values for the example hierarchy from Figure 6. Then we compute the similarity between pairs of attribute values with the help of the hierarchy code. Figure 8 shows the resulting scores for the

Policy P_1

Attr	Hcode
v_{11}	1.1
v_{12}	1.2.1.1
v_{13}	1.2.1.2
v_{14}	1.3.2

Policy P_2

Attr	Hcode
v_{21}	1.1
v_{22}	1.2
v_{23}	1.3.2

Figure 7: Hierarchy Code

example. Next, we pick up exactly matched pairs, which are $\langle v_{11}, v_{21} \rangle$ and $\langle v_{14}, v_{23} \rangle$ in the example. For the remaining attribute values, we find pairs that maximize the sum of similarity scores of pairs. In this example, $\langle v_{12}, v_{22} \rangle$ has the same similarity score as $\langle v_{13}, v_{22} \rangle$, and hence we need to

P_2 \ P_1	v_{11} (1.1)	v_{12} (1.2.1.1)	v_{13} (1.2.1.2)	v_{14} (1.3.2)
v_{21} (1.1)	**1**	0.33	0.33	0.5
v_{22} (1.2)	0.67	**0.67**	0.67	0.5
v_{23} (1.3.2)	0.5	0.17	0.17	**1**

Figure 8: Similarity Score of Two Sets of Attributes

further consider which choice can lead to a bigger compensating score. The compensating score δ is for attribute values which do not have matchings when two attributes have different number of values. δ is computed as average similarity scores between unmatched values with all the values of the other attribute. For this example, no matter which pair we choose, the compensating score is the same. Suppose we choose the pair $\langle v_{12}, v_{22}\rangle$, and then one value v_{13} is left whose compensating score δ is $(0.33 + 0.67 + 0.17)/3 = 0.39$. Finally, the similarity score for the two attribute a_1 and a_2 takes into account both the similarity of attribute names and attribute values. Specifically, the similarity score for attribute names is 1 since they are exactly matched, and the similarity score for attribute values is the average scores of pairs and the compensating score. The final score is $\frac{1}{2}[1 + (1 + 1 + 0.67 + 0.39)/4] = 0.88$.

The similarity score of two categorical predicates is finally defined as below:

$$S_{cat}(a_1, a_2) = \frac{1}{2}\Big[1 + \frac{\sum_{(v_{1k}, v_{2l}) \in \mathcal{M}_v} s_{cat}(v_{1k}, v_{2l}) + \delta}{max(N_{v_1}, N_{v_2})}\Big] \quad (11)$$

$$\delta = \begin{cases} \dfrac{\sum_{(v_{1k}, \_) \notin \mathcal{M}_v} \sum_{l=1}^{N_{v_2}} s_{cat}(v_{1k}, v_{2l})}{N_{v_2}}, & N_{v_1} > N_{v_2}; \\ \dfrac{\sum_{(\_, v_{2l}) \notin \mathcal{M}_v} \sum_{k=1}^{N_{v_1}} s_{cat}(v_{1k}, v_{2l})}{N_{v_1}}, & N_{v_2} > N_{v_1}. \end{cases} \quad (12)$$

where N_{v_1} and N_{v_2} are the total numbers of values associated with attributes a_1 and a_2 respectively.

3.3.2 Similarity Score for Numerical Predicates

Unlike categorical values, numerical values do not have any hierarchical relationship. For computation efficiency, the similarity of two numerical values v_1 and v_2 is defined based on their difference as shown in equation 13.

$$s_{num}(v_1, v_2) = \frac{|v_1 - v_2|}{max(v_1, v_2)} \quad (13)$$

The s_{num} tends to be large when the difference between two values is small.

The computation of the similarity score of two numerical value sets is similar to that for the two categorical value sets; we thus have the following similarity definition for numerical predicates:

$$S_{num}(a_1, a_2) = \frac{1}{2}\Big[1 + \frac{\sum_{(v_{1k}, v_{2l}) \in \mathcal{M}_v} s_{num}(v_{1k}, v_{2l}) + \delta}{max(N_{v1}, N_{v2})}\Big] \quad (14)$$

$$\delta = \begin{cases} \dfrac{\sum_{(v_{1k}, \_) \notin \mathcal{M}_v} \sum_{l=1}^{N_{v_2}} s_{num}(v_{1k}, v_{2l})}{N_{v_2}}, & N_{v_1} > N_{v_2}; \\ \dfrac{\sum_{(\_, v_{2l}) \notin \mathcal{M}_v} \sum_{k=1}^{N_{v1}} s_{num}(v_{1k}, v_{2l})}{N_{v_1}}, & N_{v_2} > N_{v_1}. \end{cases} \quad (15)$$

3.4 Overall Algorithm

In this section, we summarize the steps involved in the computation of a similarity score between two policies P_1 and P_2. Figure 9 presents the pseudo-code of the complete algorithm, which consists of five phases. First, we categorize rules in P_1 and P_2 based on their *effects* (line 1). Second, we compute the similarity score S_{rule} for each pair of rules in P_1 and P_2 (line 2-7). Third, based on the S_{rule}, we compute the Φ mappings (line 8-11). Fourth, we use the Φ mappings to calculate the rule set similarity scores (line 12-23). Finally, the overall similarity score is obtained (line 24).

Algorithm PolicySimilarityMeasure(P_1, P_2)
Input : P_1 is a policy with n rules $\{r_{11}, r_{12}, ..., r_{1n}\}$ and P_2 is a policy with m rules $\{r_{21}, r_{22}, ..., r_{2m}\}$

1. Categorize rules in P_1 and P_2 based on their *effects*. Let $PR_1(PR_2)$ and $DR_1(DR_2)$ denote the set of permit and deny rules respectively in $P_1(P_2)$.

/* Compute similarity scores for each rule in P_1 and P_2 */
2. **foreach** rule $r_{1i} \in PR_1$
3. **foreach** rule $r_{2j} \in PR_2$
4. $S_{rule}(r_{1i}, r_{2j})$ //compute similarity score of rules
5. **foreach** rule $r_{1i} \in DR_1$
6. **foreach** rule $r_{2j} \in DR_2$
7. $S_{rule}(r_{1i}, r_{2j})$ //compute similarity score of rules

/* Compute Φ mappings */
8. $\Phi_1^P \leftarrow$ComputePhiMapping(PR_1, PR_2, ϵ)
9. $\Phi_2^P \leftarrow$ComputePhiMapping(PR_2, PR_1, ϵ)
10. $\Phi_1^D \leftarrow$ComputePhiMapping(DR_1, DR_2, ϵ)
11. $\Phi_2^D \leftarrow$ComputePhiMapping(DR_2, DR_1, ϵ)

/* Compute the rule set similarity scores */
12. **foreach** rule $r_{1i} \in P_1$
13. **if** $r_{1i} \in PR_1$ **then**
14. $rs_{1i} \leftarrow$ ComputeRuleSimilarity(r_{1i}, Φ_1^P)
15. **elsif** $r_{1i} \in DR_1$ **then**
16. $rs_{1i} \leftarrow$ ComputeRuleSimilarity(r_{1i}, Φ_1^D)
17. **foreach** rule $r_{2j} \in P_2$
18. **if** $r_{2j} \in PR_2$ **then**
19. $rs_{2j} \leftarrow$ ComputeRuleSimilarity(r_{2j}, Φ_2^P)
20. **elsif** $r_{1i} \in DR_1$ **then**
21. $rs_{2j} \leftarrow$ ComputeRuleSimilarity(r_{2j}, Φ_2^D)
22. $S_{rule-set}^P \leftarrow$ average of rs of permit rules
23. $S_{rule-set}^D \leftarrow$ average of rs of deny rules

/* Compute the overall similarity score */
24. $S_{policy}(P_1, P_2) = S_T(P_1, P_2) + w_p S_{rule-set}^P + w_d S_{rule-set}^D$

end PolicySimilarityMeasure.

Figure 9: Algorithm for Policy Similarity Measure

Procedure ComputeRuleSimilarity(r', Φ)
Input : r' is a rule and Φ is a mapping between rules
1. **foreach** rule $r'' \in \Phi$
2. sum = sum $+S_{rule}(r', r'')$
3. rs = $\frac{sum}{|\Phi|}$
4. **return** rs
end ComputeRuleSimilarity.

Figure 10: Procedure for Computing Rule Similarity

The most computationally expensive part of the algorithm is to compute the S_{rule}. We analyze its complexity as follows. S_{rule} is the sum of similarity scores of corresponding elements. Suppose the average number of attributes in one element is n_a. To find matching attributes with the same name, it takes $O(n_a \log n_a)$ to sort and compare the list of attribute names. For each pair of matching attributes, we further compute the similarity scores of attribute values. Generally speaking, one attribute name is associated with one or very few number of values (e.g. ≤ 10). Therefore, we estimate the time for the attribute value computation to be a constant time c. Then the complexity of computing a similarity score of two elements is $O(n_a \log n_a + n_a c)$. For each rule, there are at most 5 elements, and the computation complexity of S_{rule} is still $O(n_a \log n_a)$.

It is worth noting that n_a is usually not a big value. For an entire policy, the total number of attribute-value pairs tested in [5] is 50. The maximum number of attribute-value pairs in one policy we have seen so far is about 500 [17]. Considering that the average number of attributes in one policy component is even smaller, our similarity score computation is very efficient.

3.5 Case Study

In this section we provide a detailed example to illustrate how our policy similarity measure algorithm works. Continuing with the policy examples P_1, P_2 and P_3 introduced in section 2, we show how our policy similarity algorithm assigns a similarity score between these policies. We further show that our similarity algorithm assigns a higher similarity score between the data owner policy P_1 and resource owner policy P_2 than between the data owner policy P_1 and resource owner policy P_3, adequately representing the relationship between the sets of requests permitted(denied) by the corresponding policies. Thus using the similarity score computed by our algorithm, the data owner is notified that P_2 is more compatible to his own policy. The data owner only need to further check one policy P_2 instead of testing two policies before sending his data to the resource owner.

In the following discussion we refer to the policies shown in Figures 2, 3 and 4. We also refer to two attribute hierarchies in the domain, namely the user hierarchy (Figure 11) and file type hierarchy (Figure 12). Without having any additional knowledge of the application, we assume that each rule component is equally important and hence assign the same weight to all computations.

The similarity score between P_1 and P_2 is calculated as follows:

1. We categorize rules in P_1 and P_2 based on their effects and find the permit and deny rule sets, $PR_1(PR_2)$ and $DR_1(DR_2)$. These sets are

$$PR_1 = \{R11\}$$
$$PR_2 = \{R21, R22\}$$
$$DR_1 = \{R12\}$$
$$DR_2 = \{R23, R24\}$$

2. We compute the rule similarity scores between pairs of rules with the same effect in both policies.

$$S(R11, R21) = 0.81$$
$$S(R11, R22) = 0.56$$
$$S(R12, R23) = 0.81$$
$$S(R12, R24) = 0.76$$

3. For policy P_1, we find the Φ mappings Φ_1^P and Φ_1^D using the **ComputePhiMapping** procedure. We use 0.7 as the value of the threshold for this example when computing the mappings. The Φ mappings obtained for policy P_1 are as follows:

$$\Phi_1^P = \{R11 \rightarrow \{R21\}\}$$
$$\Phi_1^D = \{R12 \rightarrow \{R23, R24\}\}$$

4. The Φ mappings Φ_2^P and Φ_2^D are calculated similarly for policy P_2.

$$\Phi_2^P = \{R21 \rightarrow \{R11\}, R22 \rightarrow \{\}\}$$
$$\Phi_2^D = \{R23 \rightarrow \{R12\}, R24 \rightarrow \{R12\}\}$$

5. For each rule r_{1i} in P_1, the corresponding rule similarity score rs_{1i} is computed:

$$rs_{11} = S_{rule}(R11, R21) = 0.81$$
$$rs_{12} = \frac{1}{2}\big[S_{rule}(R12, R23) + S_{rule}(R12, R24)\big] = 0.79$$

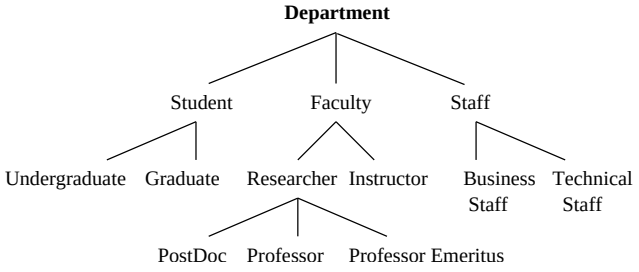

Figure 11: User Hierarchy in the University Domain

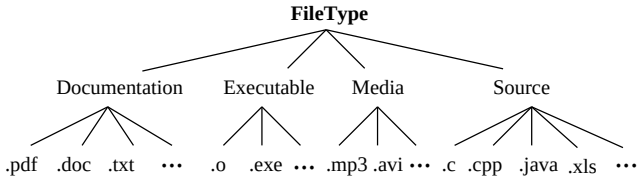

Figure 12: File Hierarchy in the University Domain

6. For each rule r_{2j} in P_2, the corresponding rule similarity score rs_{2j} is computed:

$$rs_{21} = S_{rule}(R11, R21) = 0.81$$
$$rs_{22} = 0$$
$$rs_{23} = S_{rule}(R12, R23) = 0.81$$
$$rs_{24} = S_{rule}(R12, R24) = 0.76$$

7. Then, the similarity between the permit rule sets of P_1 and P_2, given by $S^P_{rule-set}$ is computed:

$$\begin{aligned} S^P_{rule-set} &= \frac{rs_{11} + rs_{21} + rs_{22}}{3} \\ &= \frac{0.81 + 0.81 + 0}{3} \\ &= 0.54 \end{aligned}$$

8. The similarity between the deny rule sets of P_1 and P_2, given by $S^D_{rule-set}$, is computed:

$$\begin{aligned} S^D_{rule-set} &= \frac{rs_{12} + rs_{23} + rs_{24}}{3} \\ &= \frac{0.79 + 0.81 + 0.76}{3} \\ &= 0.79 \end{aligned}$$

9. Finally the permit and deny rule set similarities and policy target similarities are combined to obtain the overall policy similarity score between policies P_1 and P_2:

$$\begin{aligned} S_{policy}(P_1, P_2) &= \frac{1}{3}S_T + \frac{1}{3}S^P_{rule-set} + \frac{1}{3}S^D_{rule-set} \\ &= \frac{1}{3}\cdot 0.75 + \frac{1}{3}\cdot 0.54 + \frac{1}{3}\cdot 0.79 \\ &= \mathbf{0.71} \end{aligned}$$

We then calculate the policy similarity score for policies P_1 and P_3. The policy target similarity score $S_T = 0.5$. The rule similarity scores for policies P_1 and P_3 are:

$$S(R11, R21) = 0.7$$
$$S(R12, R23) = 0.66$$

By using the threshold 0.7, we obtain the following Φ mappings:

$$\Phi^P_1 = \{R11 \rightarrow \{R31\}\}$$
$$\Phi^D_1 = \{R12 \rightarrow \{\}\}$$

Following the same steps as described for policies P_1 and P_2, we have the following similarity score between P_1 and P_3.

$$\begin{aligned} S_{policy}(P_1, P_3) &= \frac{1}{3}S_T + \frac{1}{3}S^P_{rule-set} + \frac{1}{3}S^D_{rule-set} \\ &= \frac{1}{3}\cdot 0.5 + \frac{1}{3}\cdot 0.7 + \frac{1}{3}\cdot 0 \\ &= \mathbf{0.4} \end{aligned}$$

We observe that policy P_1 is clearly more similar to policy P_2 than to policy P_3. Hence the data owner would choose to maintain his data on the resource owner with policy P_2 rather than the resource owner with policy P_3.

4. RELATED WORK

A lot of efforts has been devoted to the analysis of access control policies with respect to policy property verification, policy conflict detection and so forth.

Most approaches to policy analysis are based on the technique of model checking. Ahmed et al. [2] propose a methodology for specifying and verifying security constraints in role based CSCW (Computer Supported Cooperative Work) systems by using finite-state based model checking. Guelev et al. present a formal language for expressing access-control policies and queries [6]. Their follow-up work [18] provides a model-checking algorithm which can be used to evaluate access control policies written in their proposed language. The evaluation includes not only assessing whether the policies give legitimate users enough permissions to reach their goals, but also checking whether the policies prevent intruders from reaching their malicious goals.

Concerning the specific problem of policy similarity, this problem has not been much investigated and only a few approaches have been proposed. Koch et al. [8] propose a uniform framework for comparing different policy models. Specifically, they use graph transformations to represent policy change and integration. Though they present examples of changes and the result as graphs, they did not give any detailed algorithm. A more practical work is by Fisler et al. [5], who developed a software tool known as Margrave for analyzing role-based access-control policies written in XACML. Margrave represents policies using the Multi-Terminal Binary Decision Diagram (MTBDD) and is able to verify policy properties and analyze differences between versions of policies. In [3], Backes et al. propose an algorithm for checking refinement of privacy policies in an enterprise. The concept of policy refinement is similar to policy similarity in some sense because policy refinement checks if one policy is a 'subset' of another. Another category of relevant related work is represented by the approaches to the problem of policy conflict detection [9, 14]. A recent work by Agrawal et al. [1] investigates interactions among polices and proposes a ratification tool by which a new policy is checked before being added to a set of policies. The main idea of such approach is to determine satisfiability of boolean expressions corresponding to different policies.

Most recently, Mazzoleni et al. [10] also considered policy similarity problem in their proposed policy integration algorithm. However, their method of computing policy similarity is limited in identifying policies specifying the same attribute.

Unlike existing approaches to policy similarity analysis which require extensive comparison between policies, our proposed similarity measure is a lightweight approach which aims at reducing the searching space, that is, at reducing the number of policies that need to be fully examined. From the view of an entire policy analysis system, our policy similarity measure can be seen as a tool which can act as a filter phase, before more expensive analysis tools are applied.

For completeness it is also important to mention that the problem of similarity for documents has been investigated in the information retrieval area. Techniques are thus available for computing similarity among two documents (e.g. [4, 7, 12]). However, these cannot be directly applied because of the special structures and properties of the XACML policies.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we defined a novel policy similarity measure which can be used as a filter approach in policy comparison. The policy similarity measure represents a lightweight approach to quickly analyze similarity of two policies. According to the obtained similarity scores, dissimilar policies can be safely pruned so that the number of policies which need to be further examined is largely reduced. Detailed algorithms of computation of similarity scores are presented. Finally, we applied our similarity measure to an example application and the result shows that the proposed similarity measure reflect the policy similarity very well.

We plan to extend our work along the following directions. The first direction is related to extend our similarity measure algorithm with techniques to solve the heterogeneity of attribute names. Also the use of ontology and ontological reasoning needs to be investigated. Our goal is to enhance our similarity measure approach with semantic reasoning techniques. The second is to integrate our algorithm with other policy analysis tools in order to develop a comprehensive environment for policy analysis. Another interesting direction is to extend current approach to general policy analysis not limited to XACML-based policies. This may be feasible since different types of policies do share a common set of entities.

Acknowledgement

The work reported in this paper has been partially supported by IBM under the OCR project "Privacy and Security Policy Management". Participants to this project are: Carnegie Mellon University, IBM T.J. Watson Research Center, Purdue University.

6. REFERENCES

[1] D. Agrawal, J. Giles, K. W. Lee, and J. Lobo. Policy ratification. In *Proceedings of the 6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 223–232, 2005.

[2] T. Ahmed and A. R. Tripathi. Static verification of security requirements in role based cscw systems. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 196–203, 2003.

[3] M. Backes, G. Karjoth, W. Bagga, and M. Schunter. Efficient comparison of enterprise privacy policies. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC)*, pages 375–382, 2004.

[4] M. Ehrig, P. Haase, M. Hefke, and N. Stojanovic. Similarity for ontologies - a comprehensive framework. In *Proceedings of the 13th European Conference on Information Systems, Information Systems in a Rapidly Changing Economy (ECIS)*, 2005.

[5] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and change-impact analysis of access-control policies. In *Proceedings of the 27th International Conference on Software Engineering (ICSE)*, pages 196–205, 2005.

[6] D. P. Guelev, M. Ryan, and P. Schobbens. Model-checking access control policies. In *Proceedings of the 7th Information Security Conference (ISC)*, pages 219–230, 2004.

[7] T. hoad and J. Zobel. Methods for identifying versioned and plagiarized documents. *Journal of the American Society for Information Science and Technology*, 54(3):203–215, 2003.

[8] M. Koch, L. V. Mancini, and F. P.-Presicce. On the specification and evolution of access control policies. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 121–130, 2001.

[9] E. Lupu and M. Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering (TSE)*, 25(6):852–869, 1999.

[10] P. Mazzoleni, E. Bertino, and B. Crispo. Xacml policy integration algorithms. In *Proceedings of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 223–232, 2006.

[11] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE)*, pages 117–128, 2002.

[12] D. Metzler, Y. Bernstein, W. B. Croft, A. Moffat, and J. Zobel. Similarity measures for tracking information flow. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM)*, pages 517–524, 2005.

[13] T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. In *Proceedings of the 24th International Conference of Very Large Data Bases (VLDB)*, pages 122–133, 24–27 1998.

[14] J. D. Moffett and M. S. Sloman. Policy conflict analysis in distributed system management. *Journal of Organizational Computing*, 1993.

[15] T. Moses. Extensible access control markup language (xacml) version 1.0. *Technical report, OASIS*, 2003.

[16] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The International Journal on Very Large Data Bases (VLDB Journal)*, 10(4):334 – 350, 2001.

[17] A. Schaad, J. D. Moffett, and J. Jacob. The role-based access control system of a european bank: a case study and discussion. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 3–9, 2001.

[18] N. Zhang, M. Ryan, and D. P. Guelev. Evaluating access control policies through model checking. In *Proceedings of the 8th Information Security Conference (ISC)*, pages 446–460, 2005.