Similarity Analysis on Heterogeneous Security Policy

Chen Chen, Yuqing Sun†, and Peng Pan
School of Computer Science & Technology, Shandong University, P. R. China
chenmon_1983@sina.com;{sun_yuqing,ppan}@sdu.edu.cn

Abstract

To support efficient cooperation among different organizations in pervasive computing environment, it is an important issue to compare the similarity among different security policies so as to authorize appropriate rights to each other. Although there is much work on security policy similarity comparison, most of them are based on the same concept structure, which is not suitable for pervasive environment where security policies are always based on heterogeneous concept structures. In this paper, we propose a novel approach to compute the similarity between security policies, which is in two processes. Firstly, analysis and integration is on different concept structures by semantic merging. Then each rule of the security policies is mapped to the integrated structure and the similarity between them is calculated. We propose the relevant algorithms and calculation methods. The experiments are made on both efficiency and effectiveness and results show that our approach is well solving the similarity analysis of heterogeneous security policy.

Keywords: Security Policy, Similarity, Semantics

1. Introduction

To support efficient cooperation among different organizations in pervasive computing environment, it is an important issue to compare the similarity among different security policies so as to authorize appropriate rights to each other [1][2]. For example, there is an emergency that a heart disease patient is carried to the nearest hospital by an ambulance, however, the EMR(Electronic Medical Records) of this patient does not store in the medical database of this hospital. In order to save diagnostic time in rescue, this hospital should interact with other hospitals having the EMR of the patient. For ensuring the access request can be accepted, this hospital will choose the one which the security policy is the most similar with it to interact. It is important to analyze the similarity among heterogeneous security policies so as to efficient cooperate in this scenario.

Most existing approaches to policy comparison and analysis are based on logical reasoning [3][4] and model checking[5][6]. [3] investigates interactions among policies and proposes a ratification algorithm by which a new policy is checked before being added to a set of policies for judging whether they are consistent. [4] explores the use of deductive spreadsheets(DSS) for development and analysis of security policies for role-based access control models. DSS combine the power of deductive rules with the usability of spreadsheets for specifying policies and analysis. And it can be developed and customized rapidly. The model check verifies whether a given state in the propositional constraint is satisfied with the given security policies.[5]developed a software tool known as Margrave for analyzing policies written in XACML[7](eXtensible Access Control Markup Language). The Margrave expresses policy with MTBDD(Multi-Terminal Binary Decision Diagram). The MTBDD model is a structure to express boolean functions evolved from BDD (Binary Decision Diagram) [8]. This model can analyze policy with predefined rules to decide whether the policy violates the constraints. [6] establishes the analysis for general administrative role-based access control policies and gives algorithms for cases where the analysis problem is solvable in polynomial time.

In [9], the policy similarity comparison is calculated based on one uniform *concepts structure*, which is a tree hierarchy showing the concepts semantic relationships. The policies calculated in [9] is written in XACML, the current OASIS (Organization for the Advancement of Structured Information Standards) standard for specifying security policies and widely adopted to express heterogeneous security policies in web. We also use XACML to present policies in our paper.

The method in [9] assumes that all organizations use the same concepts to express security policies, which is not suitable for heterogeneous policies in pervasive computing. In this paper, we propose a security policy similarity computation algorithm having two steps to solve this problem. The first one can extract concepts of different attributions in policies to form the concept structures of attributions and then merge these semantic heterogeneous concept structures into one uniform structure. In the second process, we calculate the change of concept structures after merging which includes two parts to obtain the security policies similarity: the change of relative position on each node and the change of semantic distance between two

† Corresponding author: Jinan, Shandong, China. 250101

nodes. We would show the complexity analysis and experiment results to prove our algorithm is effective.

The rest of paper is organized as follows. Section 2 introduces the preliminary knowledge. Section 3 presents the detail of our algorithm. Section 4 gives our experiments. Finally, we conclude the paper and outline future work in section 5.

2. Preliminary knowledge

The ontology mapping or merging is widely used in semantic comparison. CTXMATCH[10] is an algorithm for discovering semantic mappings across hierarchical classifications(HCs) by logical deduction and WordNet[11]. It takes two inputs H and H_1 in HCs, and for each pair of concepts $k \in H$, $k1 \in H_1$, returns their semantic relationship: more general($\supseteq$), less general($\subseteq$), equivalent($\equiv$), compatible(*) or disjoint($\perp$). The typical approaches for ontology mapping or merging include [12,13].

The computing principle of [9] is grouping the rules in two policies p_1 and p_2 according to their effects, which results in a set of Permit Rules(denoted as PR)and Deny Rules(denoted as DR), and then each single rule r_i in p_1 is compared with a rule r_j in p_2 that has the same effect by the sum of attribution similarity of 4 elements (subject, resource, action, condition) according to XACML policy structure(Fig. 1). Then a similarity score of two rules is obtained (denoted as $S_{rule(ri,rj)}$). The policy similarity of p_1 and p_2 can be finally obtained by mapping, summing and averaging all similarity scores of rule pairs.

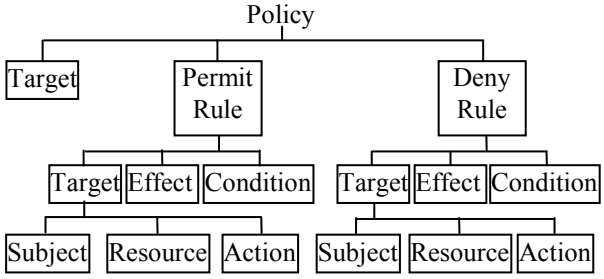


Figure 1. XACML policy structure

When computing the attribution similarity, the algorithm in [9] classifies attributions as two categories based on the type of their values, namely categorical attribution and numerical attribution.

- Categorical attribution: The attribute values are concepts belonging to some domain-specific ontology. The attribution "assignation" in "Assignation = professor" belongs to the categorical type for example.
- Numerical attribution: The attribute values are numerical values. The attribution "time" in "Time=12:00" is of numerical type for example.

The computation for *categorical attributions* relies on an assumption that a semantic structure exists for the categorical values and is based on the structure level distance of structure. So it has two shortcomings as follows. One is it is based on the same concept structure. This algorithm assumes that *categorical attributions* of different policies use the same expressions in each element. So the computation for *categorical attributions* of different policies is based on the same concept structure. This is not suitable for pervasive computing. Another one is it is based on the structure level distance, namely the number of edges in the computation for *categorical attributions*. But actually, the semantic distance of any two nodes connected by one edge is different. The analysis based on the structure level distance is not very scientific (The example will be shown in section 4).

The computation for *numerical attribution* of [9] also has shortcomings. It only works on positive integers.

3. Similarity computation algorithm

The proposed security policy similarity computation algorithm is based on XACML policy language. We assume that semantic structures exist for the corresponding *categorical attribution* values. Based on our assumption, we merge heterogeneous concept structures of attributions belonging to different policies into one uniform structure according to the five semantic relationships defined by CTXMATCH. So we can analyze different policies based on one uniform structure. And then we use the semantic structure defined by WordNet to compute the similarity of *categorical attributions* by the semantic level distance not the structure level distance.

So, our algorithm involves two processes. The first one merges the heterogeneous concept structures by CTXMATCH. In the second process, we focus on the shortcomings of [9] mentioned in section 2 to propose a new computation for its core process. We compute the policy similarity by analyzing the change value of concept structures after merging based on the results of the first process.

We will elaborate our algorithm according to its processes below. The main functions and predicates used in the algorithm are listed in table 1.

3.1. Merging concepts structure

We give the specific definition about concepts structure before the elaboration of our algorithm.

Table 1. Functions/Predicates used in the algorithm

Function/predicate	Meaning
CTX(n,n')	Judges the semantic relationship between nodes n and n' by CTXMATCH and returns one of five semantic relationships it defined.
Merge(n,n')	Merges nodes *n* and *n*' in different ways according to CTX returns.
JudgeLevel(C)	Selects a comparing level for node C belonging to another structure according to the merging way of the parent of C(if C is root, the comparing level is the level of another root)
value(n)	Change value of node *n* in concepts structure
change(n)	Change value of relative position on node *n*
n_{ci}	The i^{th} child of node *n*
semanValue(En_{ci}-root)	The value of semantic level distance between the i^{th} child of node *n* and root in the merged structure

Definition 1: a concepts structure is a finite set having n(n≥0) nodes which satisfy below features:

(1) There is only one node name root and this node is in the top row of concepts structure.

(2) If n>1, the nodes except root can be divided into m(m>0)mutually disjoint finite sets S_1, S_2,…, S_m, and each set itself is another concepts structure named sub-structure. We call S_i is the **child** of its own definite set *S* and *S* is the **parent** of all its sub-structure. The two sub-structures S_i and S_j of concepts structure *S* named **brother**.

(3) Each node in concepts structure present one concept in the same domain-specific ontology.

This process of algorithm inputs two heterogeneous concept structures H_A and H_B rooted at A and B respectively, and outputs one merged concept structure. The merging will traverse H_A in level order with one queue and select one comparing level in H_B for each node traversed to judge the semantic relationship between this node and each node of the comparing level.

Then we process the merging in different ways according to different semantic relationship results. In our paper, we assume that the semantic relationship among nodes in same level is only compatible (*) and among one node and its children is only more general ($\supseteq$) in structures. And we don't consider the intersectional semantic relationship e.g. *boy* and *freshman* in this paper. The detail of algorithm is shown as Fig. 2.

```
Procedure StructureMerge(A,B)
Input: Two heterogeneous structures H_A, H_B
Output: One uniform structure H_B'
1 if CTX(A,B)=* then
2  build a new root as the parent of A and B
3 else
/* Traverse H_A in level order with one queue, k is the
number of max level */
4  for i=1 → k
5   for each node  C in i^th level of H_A
6    JudgeLevel(C)  //select a comparing level in H_B
7    for each node C' in comparing level in H_B
8      Merge(C,C')
9 return H_B'
end StructureMerge(A,B)

Merge(n,n')
1 if CTX(n,n')=⊆
2  if n' has children then Merge(n, children of n')
3  else the subtree rooted at n is merged as a child of
        n', the children of n will not be inserted into
        the queue
4 elseif CTX(n,n')=⊇ then
5  n' is merged as the child of n
/* Select a comparing level for the children of n */
6  JudgeLevel(n) is the level of n'
7 elseif CTX(n,n')=*
8  if n' has brothers then Merge(n, brothers of n')
9  else the subtree rooted at n is merged as a brother of
        n', the children of n will not be inserted into
        the queue
10 elseif CTX(n,n')=≡ then
11  n is merged into n'
/* Select a comparing level for the children of n */
12  JudgeLevel(n) is the level of the children of n'
```

Figure 2. Concept structures merge algorithm

In Fig. 2, if the semantic relationship of two roots is *, the new root of A and B is their common parent in the semantic structure defined by WordNet. If CTX(n,n') is $\subseteq$ or * , the algorithm will be in the recursion or the children of n will not be inserted into the queue, so we don't need the JudgeLevel function to select a comparing level for the children of n. When CTX(n,n') is $\supseteq$, the children of n only compare with the nodes whose semantic relationship with n is $\supseteq$ in the level of n'. The algorithm is over when the traversal is over, and H_A is merged into H_B to form one uniform structure for the next process of our algorithm.

3.2. Calculating policy similarity

As the analysis in section 2, the core process of [9] is the computation for $S_{rule(ri,rj)}$, however, it has shortcomings not only for the *categorical attributions*, but also for the *numerical attributions*. In this process of our algorithm, we will propose our new computation for $S_{rule(ri,rj)}$ for the *categorical attributions* and *numerical attributions* separately below.

3.2.1. Computation for categorical attributions. In XACML policy, the difference between rules r_1 and r_2 consists of the difference of attribution values in four elements: subject, resource, action and condition. If the attribution of one element is *categorical attribution*, as our assumption, there exists corresponding concept structure for its values. So the difference of attribution values in rules r_1 and r_2 can be showed as the change value of their structures after merging.

For instance, we can compare the structures of r_1 and r_2 after merging, and do the same comparison for rules r_1 and r_3 to compute the similarity of them belonging to different policies in subject element. If the change value of the structures of r_1 and r_2 is less than r_1 and r_3, we will say the r_1 and r_2 is more similar than r_1 and r_3 in subject element.

For analyzing the change value of structures, we traverse each node belonging to the merged structure by level order, and compute recursively the relative position change of node traversed by comparing with its own pre-merged structure. And we use the semantic level distance of edge as the weight of each node except root in the traversal. The change value computation is defined as follows:

$$value(n)=\begin{cases} change(n)+\sum_{i=1}^{k} value(n_{Ci}) * SemanValue(E_{n_{Ci}\text{-root}}) & n \neq \text{leaf node} \\ change(n) & n = \text{leaf node} \end{cases} \quad (1)$$

In equation 1, k is the number of children of one node. The value(n) is different according to whether the concept of node n belongs to attribute values or not. We will elaborate change(n) and semanValue($E_{nci\text{-}root}$) below.

- change(n) For the *categorical attributions*, the concept structure after merging is formed or merged by semantic knowledge. The position of one node in structure can reflect the semantic relationship among the concept it represents and other concepts. Therefore, the change of relative position means the semantic relationship among concepts is changed. So, we can infer the difference between two rules from the change value of attribution values after merging to compute the similarity of rules based on semantics.

The change of relative position can be measured by the change of parent, children and brothers of one node between its own structure and the merged one. The formulation of it is given by the following equation:

$$Par+Chi*Change_{chinum}+Bro*Change_{bronum} \quad (2)$$

In equation 2, *Par*, *Chi*, *Bro* represent the parent, children and brother respectively. We assume that the change of parent, children and brothers has the same effect on the relative position change of one node. So we set *Par*=*Chi*=*Bro*=1. If there is no change in parent or children or brothers, the *Par* or *Chi* or *Bro* is 0. $Change_{chinum}$ and $Change_{bronum}$ denote the number of changed children and brothers. We don't consider the number of changed parent because each node only has one parent in structure. The word "change" we discuss means increase, reducing and varying. The more different between two rules in some element, the more change in the concept structures after merging, the more value of equation 2 and 1 are, the less similar of two sets of attribute values are, so the less similar of two rules are, and vice versa.

- semanValue($E_{nci\text{-}root}$) If there is an edge connecting two nodes, it means there exists hyponym, hypernym, meronym or holonym[11] semantic relationship between these two nodes as our assumption in section 3.1. We use the semantic level distance between two concepts as the value of edge. This kind of value shows the closing degree of two concepts in semantics. So, we use the semantic level distance between the concept of node n_{ci} and root in the merged structure as the weight of n_{ci} to reflect the semantic position of each concept directly. The semantic level distance can be obtained by definition as follows:

Definition 2: The semantic level distance between two concepts is obtained by the difference value of their own level number defined by the hyponym, hypernym, meronym or holonym semantic structure in WordNet 3.0.

What should be noticed in the computation is: 1. If one node in the merged structure is formed by merging two nodes, and there are two different change(n) values on this node, we should chose the less one. Because if two nodes can be merged, the concepts they represent is equivalent in semantics. The change of relative position on this kind of node in the merged structure results from the semantic heterogeneity. If the concept is expressed as its equivalent one, the change value can be reduced. So we choose the less value. 2. If one node both exists in two pre-merged structures, and there are two different change(n) values on this node in the merged structure, we will choose the more value, or else we think the change value will be underestimated.

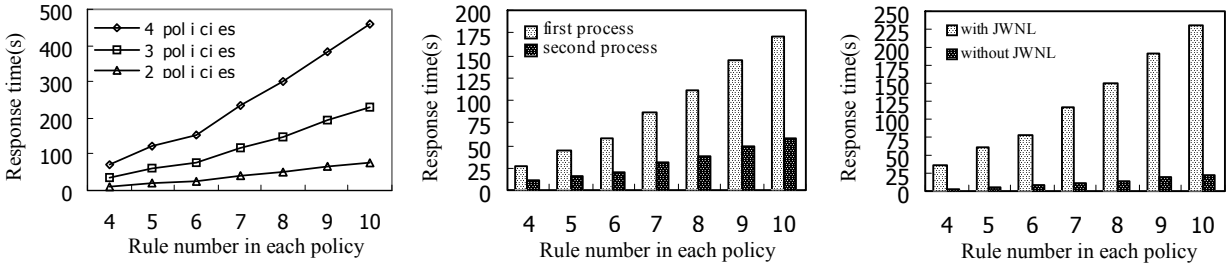
3.2.2. Computation for numerical attributions. The computation for *numerical attributions* is easier than categorical ones because we don't need to form concept structure for them. If two *numerical attribution* values v_1 and v_2 are not positive integers, we use equation 3 to compute the similarity.

$$\frac{1}{2}*\left(1-\frac{|v_{l1}-v_{l2}|}{\max(v_{l1},v_{l2})}\right)+\frac{1}{2}*\left(1-\frac{|v_{r1}-v_{r2}|}{\max(v_{r1},v_{r2})}\right) \quad (3)$$

In equation 3, v_1 is within a bound of v_{l1} and v_{r1}, so does v_2 to v_{l2} and v_{r2}.

And if two *numerical attribution* values v_1 and v_2 are positive integers, we use the following equation to compute the similarity:

$$1-\frac{|v1-v2|}{\max(v1,v2)} \quad (4)$$



(a) Response time of policy similarity computation

(b) Proportion between two processes in time cost

(c) The effects of JWNL on response time

Figure 3. Experiment results of algorithm

The more similar v1 and v2 are, the larger the equation 3 and 4 are.

We analyze the complexity of our algorithm as follows. We assume the level number of H_A and H_B is l and l', and the max number of nodes in each level of H_A and H_B is m and m' respectively. So the upper limit of complexity in the first process of our algorithm is O(l·m·max(l',m')). In the second one, we will traverse each node in the merged structure and compute the change of its relative position. The upper limit of complexity is O(max(m,m')·(l+l')) in traversal and O(1) in computing, so the upper limit of total complexity is O(l·m·max(l',m') + max (m,m') · (l+l')). If we set m' more than l' and m, then the upper limit of total complexity is O(l·m·m'). This complexity is not exponential. It means our algorithm is feasible.

4. Experiments

The experiments environment is: Intel Core 2 1.5G, RAM 1G, OS Window XP, MyEclipse 4.0, JDK 1.5, JWNL 1.4 and WordNet 3.0. The data of experiments are selected from three websites below: ontologymatching.org, www.dmoz.org, dir.yahoo.com, and we adjust them appropriately according to WordNet to form 30 structures including 300 concepts, and then build 4 policies according to these structures. Each policy is composed of 4 to 10 rules, and each rule includes all 4 elements: subject, resource, action and condition. The number of concepts belonging to attributions in subject and resource element is 2 to 6, and the attribution values in action element are the random combination of "read" and "write". We set one attribution value in condition element. The experiment results are shown in Fig. 3.

Fig. 3(a) shows the response time of similarity computation for 2 to 4 policies in seconds. We can see from the figure that the response time increases as the number of comparing rules in policies increases, and the increasing rate is multiplying. The multiplying trend is more obvious as the number of comparing policies increases because it results in the number of comparing rules multiplies. Fig. 3(b) shows the proportion between two processes of our algorithm in time cost. It is clear that the first process costs more time than the second one. And the difference value of them increases as the number of rules in policy. The experiment show that the first process costs multiple time than the second one. Fig. 3(c) shows the effects of JWNL, the WordNet API, on response time of our algorithm. We find out that most of response time is consumed by calling to JWNL by experiment data. We need to call JWNL to judge the semantic relationship of concepts in the first process and obtain the semantic level distance of edges in the second one. The calling times in the first process is much more than the second one, that's why it costs much more time as shown in Fig. 3(b). On the assumption that we are given the semantic relationship of concepts and the semantic level distance of edges, the response time of our algorithm would be greatly reduced. The data of Fig 3(b) and 3(c) is obtained in similarity computation for 3 policies.

We now compare our method with [9]. The experiments are based on the concepts structures of rules r_{1i}, r_{2j}, r_{3k} that belong to three different policies in subject element as shown in Fig. 4. We would check which rule is more similar with r_{2j} by computing the similarity between r_{1i}, r_{2j} and r_{2j}, r_{3k}. The value of edge is the semantic level distance and the concepts included in rules are denoted in boldface in Fig. 4. Each concept is denoted as the abbreviations. The computation matrix of r_{2j}, r_{1i} and r_{2j}, r_{3k} according to the algorithm of [9] is shown in Fig. 5(a) and 5(b). The rows in Fig. 5 represent the concepts of r_{2j}, while the columns in Fig. 5(a) and (b) represent the concepts of r_{1i} and r_{3k} respectively.

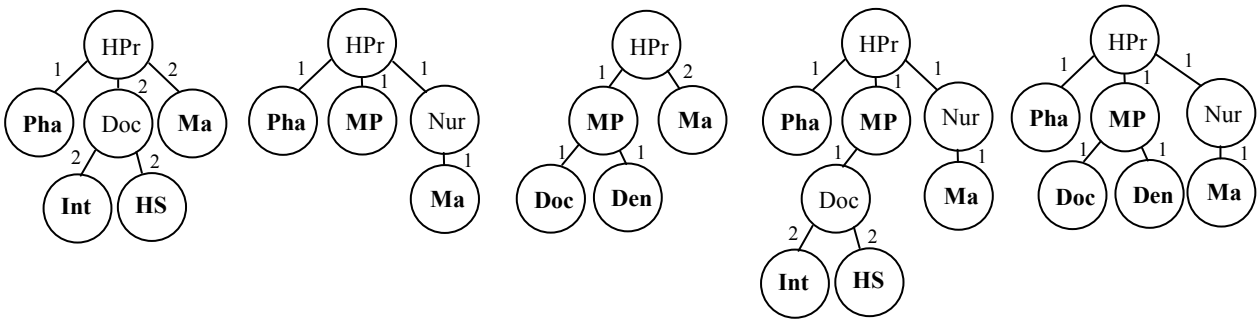
The result is $S_{rule(r1i,r2j)}=S_{rule(r3k,r2j)}=0.88$ (the rounding may cause error in Fig. 5) in subject element. We can't tell which rule is more similar with r_{2j}. Now we use our algorithm to compute the similarity between them. The structures after merging for r_{1i}, r_{2j} are corresponding to Fig. 4(a), 4(b) and 4(d). As our elaboration in section 3.2.1, The change(MP) = 0 + 1 + 0 = 1. The

$$\begin{array}{c} \\ \text{MP} \\ \text{Pha} \\ \text{Ma} \end{array}\begin{array}{c} \begin{array}{cccc} \text{Int} & \text{HS} & \text{Pha} & \text{Ma} \end{array} \\ \begin{pmatrix} \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & 1 & \frac{1}{2} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{2} & 1 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ \text{MP} \\ \text{Pha} \\ \text{Ma} \end{array}\begin{array}{c} \begin{array}{cccc} \text{Doc} & \text{Den} & \text{MP} & \text{Ma} \end{array} \\ \begin{pmatrix} \frac{5}{6} & \frac{5}{6} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{2}{3} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{2} & 1 \end{pmatrix} \end{array}$$

(a) (b)

Figure 5. Similarity computation matrix

change(Pha) = change(Ma) = change(Int) =



(a) Concepts structure of rule r_{1i} (b) Concepts structure of rule r_{2j} (c) Concepts structure of rule r_{3k} (d) Concepts structure merged from r_{1i}, r_{2j} (e) Concepts structure merged from r_{3k}, r_{2j}

Figure 4. Examples of merging concepts structures

change(HS)=0. The structures for r_{2j}, r_{3k} are corresponding to Fig. 4(b), 4(c) and 4(e). The node MP both exists in Fig. 4(b) and 4(c), so we choose the more value. The change(MP) = 0 + 2 + 0 = 2 . The other computations are similar with r_{1i} and r_{2j}.So $S_{rule(r1i,r2j)}$= value(Pha) *1 + value(MP) * 1+value(Nur)*1=0*1+(1+value(Doc)*2)*1+(value(Ma)*2)*1=0*1+(1+(value(Int)*4+value(HS)*4)*2)*1+(value(Ma)*2)*1=0*1 + (1 + (0 * 4+0* 4) * 2) * 1+ (0 * 2)*1=1, similarly, $S_{rule(r3k,r2j)}$ = value(Pha) * 1 + value(MP)*1+value(Nur)*1=2, 1<2. It means r_{1i} and r_{2j} is more similar than r_{3k} and r_{2j} in subject element.

5. Conclusion and future work

In this paper, we propose a novel method to calculate the similarity between heterogeneous policies. Our method includes two processes. First we merge two concept structures of heterogeneous policies into one uniform structure. Then we calculate the policy similarity by the differences of their structures and attributes based on such structure. The experiments show that our algorithm is efficient and effective. For the future work, we are planning to do policy mining by using the similarity analyses so as to support automatic policy update and web based cooperation.

Acknowledgement

Research supported by the National High Technology Research and Development Program (863 Program) of China (2006AA01A113).

References

[1] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, 2001, 8(4), 10–17.

[2] R. Campbell, J. Al-Muhtadi, P. Naldurg, G. Sampermane, and M. D. Mickunas, "Towards Security and Privacy for Pervasive Computing", In Proceedings of the International Symposium on Software Security, Keio University, Tokyo, Japan, Nov. 8-10, 2002, pp. 1-15.

[3] D. Agrawal, J. Giles, K. W. Lee, and J. Lobo, "Policy ratification", Proceedings of the 6th IEEE International Workshop on Policies for Distributed Systems and Networks, Stockholm, Sweden, June. 6-8,2005, pp. 223–232.

[4] A. Singh, C. R. Ramakrishnan, I. V. Ramakrishnan, S. D. Stoller and D. S. Warren, "Security policy analysis using deductive spreadsheets", Proceedings of the 2007 ACM workshop on Formal methods in security engineering, Virginia, USA, Nov. 2, 2007, pp. 42-50.

[5] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz, "Verification and change-impact analysis of access-control policies", Proceedings of the 27th International Conference on Software Engineering, St. Louis, Missouri, USA, May. 15-21, 2005, pp. 196–205.

[6] A. Sasturkar, P. Yang, S. D. Stoller, C. R. Ramakrishnan, "Policy Analysis for Administrative Role Based Access Control", Proceedings of the 19th IEEE workshop on Computer Security Foundations, Venice, Italy, July. 5-7, 2006, pp. 124-138.

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

[8] R. E. Bryant. "Graph based algorithms for Boolean function manipulation", IEEE Transactions on Computers, 1986, 35(8), 677–691.

[9] D. Lin, P. Rao, E. Bertino and J. Lobo, "An Approach to Evaluate Policy Similarity", Proceedings of the ACM Symposium on Access Control Models and Technologies, Sophia Antipolis, France, June 20-22, 2007,pp 1-10.

[10] P. Bouquet, L. Serafini and S. Zanobini, "Semantic Coordination: A New Approach and an Application", Proceedings of 7th IEEE International Symposium on Wearable Computers, New York, USA, Oct. 21-23, 2003, pp. 130-145.

[11] C. Fellbaum, WordNet: An Electronic Lexical Database, the MIT Press Cambridge, Massachusetts London, England, 1998.

[12] A. Doan, J. Madhavan, P. Domingos and A. Halevy, "Learning to Map between Ontologies on the Semantic Web", Proceedings of the 11th international conference on World Wide Web, Honolulu, Hawaii, USA, May 7-11,2002, pp. 662-673.

[13] N. F. Noy and M. A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment", Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, Austin, Texas, USA, July 30-August 03, 2000, pp. 450-455.