Clustering-based Approach for Anomaly Detection in XACML Policies

Maryem Ait El Hadj[1], Meryeme Ayache[1], Yahya Benkaouz[2], Ahmed Khoumsi[3] and Mohammed Erradi[1]

[1]*NDSR Group, ENSIAS, Mohammed V University in Rabat, Morocco*

[2]*Conception and Systems Laboratory, FSR, Mohammed V University in Rabat, Morocco*

[3]*Dept. Electrical & Comp. Eng., University of Sherbrooke, Sherbrooke, Canada*

Keywords: ABAC, XACML Policies, Clustering, Similarity Computation, Anomaly Detection.

Abstract: The development of distributed applications arises multiple security issues such as access control. Attribute-Based Access Control has been proposed as a generic access control model, which provides more flexibility and promotes information and security sharing. eXtensible Access Control Markup Language (XACML) is the most convenient way to express ABAC policies. However, in distributed environments, XACML policies become more complex and hard to manage. In fact, an XACML policy in distributed applications may be aggregated from multiple parties and can be managed by more than one administrator. Therefore, it may contain several anomalies such as conflicts and redundancies, which may affect the performance of the policy execution. In this paper, we propose an anomaly detection method based on the decomposition of a policy into clusters before searching anomalies within each cluster. Our evaluation results demonstrate the efficiency of the suggested approach.

1 INTRODUCTION

Attribute-Based Access Control model (ABAC) (Yuan and Tong, 2005) has been suggested as a generic access control model. ABAC considers a set of attributes, based on which access decisions should be taken. The attributes are any information that can be assigned to a subject (i.e. the user or the process that takes action on a resource), a resource (i.e. the entity that is acted upon by a subject) and an environment (i.e. the operational and technical context in which the information access occurs).

eXtensible Access Control Markup Language (XACML) (Anderson et al., 2003) is the most convenient way to express ABAC model. In fact, XACML defines an XML schema that supports the ABAC model. Each XACML policy contains a set of rules, each rule being composed of attributes and a decision effect (deny/permit), that decides over a given request to access a given resource. XACML policy representation is more expressive and fine-grained. However, in large collaborative platforms, processing and analyzing XACML policies might be very hard and complicated. This is due to the massive amount of information that should be considered as attributes. Therefore, XACML policies may contain several anomalies, such as redundancies and conflicting rules, which may affect the performance of the policy execution. Detecting automatically such anomalies in large sets of complex policies is primordial.

In this paper, we propose an approach to detect anomalies within an XACML policy. Inspired by (Benkaouz et al., 2016), the suggested approach is based on decomposing the policy into clusters before searching anomalies within each cluster. More precisely, given an XACML policy, we proceed as follows: (1) extract the rules of the XACML policy, (2) compute a similarity score for each pair of rules, (3) regroup similar rules into clusters. Finally, (4) detect anomalies within each cluster. In this paper, we consider three main categories of anomalies, redundancy, conflict of modality and conflict of fraction permission. The evaluation results demonstrate the efficiency of the suggested approach.

The rest of the paper is organized as follows: Section 2 present related work. In section 3 we present how rules are expressed and the similarity measure adopted. The clustering algorithm is presented in section 4. Section 5 describes the policy anomaly detection method. Section 6 reports experimental results.

Hadj, M., Ayache, M., Benkaouz, Y., Khoumsi, A. and Erradi, M.
Clustering-based Approach for Anomaly Detection in XACML Policies.
DOI: 10.5220/0006471205480553
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT*, pages 548-553
ISBN: 978-989-758-259-2

Finally, the conclusion and expected future work are drawn in section 7.

2 RELATED WORK

Regarding anomalies classification, Khoumsi et al. (Khoumsi et al., 2016) categorize the anomalies into two categories: a *conflicting anomaly* and a *nonconflicting anomaly*. On the other hand, Jonathan et al. (Moffett and Sloman, 1994) have classified the policies conflicts into four different categories: conflict of modality (permit/deny), conflict between imperative and authority policy (obliged/deny), conflict of priorities occurs when the resources are limited to meet the demands upon them, and conflict of duties when a subject has two tasks and maintains them simultaneously.

In the context of XACML, Mourad et al. (Mourad et al., 2015) use UML to offer model-driven specification of XACML policies in order to detect conflicting and redundant rules. Hu et al (Hu et al., 2013) consider representing XACML policies as decision trees to detect conflicts and redundancies. Another representation of XACML policies was proposed by Stepien et al. (Stepien and Felty, 2016). They represent XACML policies using Prolog's built-in powerful indexing system.

Contrary to the works discussed above, our work takes into account a large set of attributes. It proposes an anomaly detection method performed in each cluster of rules, instead of the whole policy set, which implies less processing time. An advantage of our anomaly detection method is that it is performed before even enforcing the policy in the system, which offers the ability to correct the anomalies and gain a significant improvement in policy decision time.

3 RULES EXPRESSION AND SIMILARITY COMPUTATION

3.1 Rules Expression

In an XACML policy, each rule has three categories of attributes (subject, resource and environment). Each of the 3 categories is indicated by $x = s$, r or e. A rule is specified by an action decision to be taken if the attributes satisfy a given condition (or profile). The action decision is noted in the form X_{act}, where X is *Permit* or *Deny* to indicate that the action *act* is permitted or denied, respectively. $Permit_{read}$ and $Deny_{write}$ are two examples of action decisions.

The profile of a rule is specified by assigning a set of values to all attributes. We use the expression $attr\_name \in attr\_values$ to assign a set of values $attr\_values$ to an attribute identified by $attr\_name$. The assignments corresponding to the same category are separated by a comma ",", while a semicolon ";" means the passing to the next category. The semantics of the rule is that its action decision is taken if for each assignment $attr\_name \in attr\_values$, the attribute $attr\_name$ takes one of the values belonging to $attr\_values$. The formal expression of a rule profile is therefore as follows:

$$X_{act}(attr\_name_1 \in attr\_values_1, \ldots; attr\_name_i = attr\_values_i, \ldots)$$

Example : $Permit_{read}$ (position $\in$ {doc}, specialist $\in${generalist} , team $\in$ {oncology}, experience $\in$ {+10} , grade $\in$ {Registrar} , department $\in$ {oncology}; type $\in$ {PR/CAT} , formatType $\in$ {AST} , degreeOfConfidentiality $\in$ {Secret} ; Organization $\in$ {EMS} , time $\in$ {8-12}).

The attributes of the *Subject* category are: position, specialist, team, experience, grade, department. The attributes of the *Resource* category are: type, formatType, degreeOfConfidentiality. The attributes of the *Environment* category are: Organization, time.

3.2 Similarity Computation

The rule similarity measure is a function S_{rule} that assigns a similarity score $S_{rule}(r_i, r_j)$ to any two given rules r_i and r_j. Such a score reflects the degree of similarity between r_i and r_j, with respect to their subject, resource and environment attributes values.

The formal definition of the similarity score $S_{rule}(r_i, r_j)$ is given in Equation 1, which is the sum of the three similarity scores $S_s(r_i, r_j)$, $S_r(r_i, r_j)$ and $S_e(r_i, r_j)$ related to the three attribute categories (subject, resource and environment), which are weighted by values W_s, W_r and W_e respectively. W_s, W_r and W_e can be chosen to reflect the relative importance to be given to the similarity computation. The weight values must satisfy the constraint: $W_s + W_r + W_e = 1$.

$$S_{rule}(r_i, r_j) = W_s S_s(r_i, r_j) + W_r S_r(r_i, r_j) + W_e S_e(r_i, r_j) \quad (1)$$

The assignment of weights relies on user needs (i.e., the weight values can be specified depending on a specific application). For example, if a user would like to compute the similarity score of *Subject* attributes regardless of *Resource* and *Environment* attributes, he can set W_s to 1, and W_r and W_s to 0. In this paper, the same value $\frac{1}{3}$ is assigned to the three weights by default.

Before continuing, we need to define the following notation:

- $ATT_x(r_i)$ is the set of attribute names of category x in rule r_i.
- $S_{att}(r_i,r_j)$ is a score reflecting the similarity of r_i and r_j based uniquely on att which is a common attribute of r_i and r_j.
- $W_{att}(r_i,r_j)$ is a nonnegative value that reflects the relative importance of an attribute att among all the common attributes of r_i and r_j of the same category as att. The sum of all these weights is equal to 1, i.e., $\sum_{att \in ATT_x(r_i) \cap ATT_x(r_j)} W_{att} = 1$ for each category x.
- $V_{att}(r_i)$ is the set of values assigned to the attribute att in the rule r_i.
- $\mid X \mid$ denotes the number of elements belonging to a set X.

Each similarity score $S_x(r_i, rj)$ (for $x = s$, r, e) of Eq. 1 is computed based on Eq. 2. This latter consists in summing the scores $S_{att}(r_i,r_j)$ for every attribute att of category x that is common to r_i and r_j. Besides, every $S_{att}(r_i,r_j)$ is weighted by $W_{att}(r_i,r_j)$.

$$S_x(r_i,r_j) = \sum_{att \in ATT_x(r_i) \cap ATT_x(r_j)} W_{att}(r_i,r_j) S_{att}(r_i,r_j) \tag{2}$$

Equation 3 shows how each similarity $S_{att}(r_i,r_j)$ is computed is computed. This equation consists in estimating the number of elements that are common to $V_{att}(r_i)$ and $V_{att}(r_j)$ relatively to the total number of elements in $V_{att}(r_i) \cup V_{att}(r_j)$.

$$S_{att}(r_i,r_j) = \frac{\mid V_{att}(r_i) \cap V_{att}(r_j) \mid}{\mid V_{att}(r_i) \cup V_{att}(r_j) \mid} \tag{3}$$

Example:

r_1: $Permit_{read}$ (Group $\in$ IBM, Designation $\in$ {Professor, PostDoc, TechStaff}; File-Type $\in$ {Source, Documentation, Executable}; Time $\in$ [8:00, 18:00]).

r_2 : $Permit_{read}$ (Group $\in$ IBM, Designation $\in$ {Student, TechStaff}; File-Type $\in$ {Source, Documentation}; Time $\in$ [12:00, 16:00]).

For the $Subject$ category, we have two attributes, Group (g) and Designation (d). For r_1, $V_g(r_1) = \{IBM\}$ and $V_d(r_1) = \{Professor, PostDoc, TechStaff\}$. While for r_2, $V_g(r_2) = \{IBM\}$ and $V_d(r_2) = \{Student, TechStaff\}$.

For the $Resource$ category, we have the attribute File-Type (ft). Where for r_1 $V_{ft}(r_1) = \{Source, Documentation, Executable\}$ and for r_2, $V_{ft}(r_2) = \{Source, Documentation\}$.

For the $Environment$ category, we have the attribute Time (t). For r_1, $V_t(r_1) =$ [8:00, 18:00]. And for r_2, $V_t(r_2) =$ [12:00, 16:00].

The similarity between r_1 and r_2 is computed using Eqs (1, 2, 3) as follows:

Use of Eq. (1). Assuming that $W_s = W_r = W_e = \frac{1}{3}$:

$$S_{rule}(r_1,r_2) = \tfrac{1}{3}S_s(r_1,r_2) + \tfrac{1}{3}S_r(r_1,r_2) + \tfrac{1}{3}S_e(r_1,r_2)$$

Use of Eq. (2). Each of the above $S_s(r_1,r_2)$, $S_r(r_1,r_2)$ and $S_e(r_1,r_2)$ is computed as follows, assuming that $W_g(r_1,r_2) = W_d(r_1,r_2) = \frac{1}{2}$, $W_{ft}(r_1,r_2) = W_t(r_1,r_2) = 1$:

- $S_s(r_1,r_2) = \frac{1}{2}\ S_g(r_1,r_2) + \frac{1}{2}\ S_d(r_1,r_2)$
- $S_r(r_1,r_2) = S_{ft}(r_1,r_2)$
- $S_e(r_1,r_2) = S_t(r_1,r_2)$

Use of Eq. (3). Each of the above $S_g(r_1,r_2)$, $S_d(r_1,r_2)$, $S_{ft}(r_1,r_2)$ and $S_t(r_1,r_2)$ is computed as follows:

- $S_g(r_1,r_2) = \frac{|V_g(r_i) \cap V_g(r_j)|}{|V_g(r_i) \cup V_g(r_j)|} = \frac{|\{IBM\}|}{|\{IBM\}|} = 1$
- $S_d(r_1,r_2) = \frac{|V_d(r_i) \cap V_d(r_j)|}{|V_d(r_i) \cup V_d(r_j)|} = \frac{|\{TechStaff\}|}{|\{Professor, PostDoc, TechStaff, Student\}|} = \frac{1}{4}$
- $S_{ft}(r_1,r_2) = \frac{|V_{ft}(r_i) \cap V_{ft}(r_j)|}{|V_{ft}(r_i) \cup V_{ft}(r_j)|} = \frac{|\{Source, Documentation\}|}{|\{Source, Documentation, Executable\}|} = \frac{2}{3}$
- $S_t(r_1,r_2) = \frac{|V_t(r_i) \cap V_t(r_j)|}{|V_t(r_i) \cup V_t(r_j)|} = \frac{|[12{:}00, 16{:}00]|}{|[8{:}00, 18{:}00]|} = \frac{4}{10}$

By combining all these equations, we obtain:

- $S_s(r_1,r_2) = \frac{1}{2}\ S_g(r_1,r_2) + \frac{1}{2}\ S_d(r_1,r_2) = \frac{1}{2} + \frac{1}{2} \times \frac{1}{4} = 0.62$
- $S_r(r_1,r_2) = S_{ft}(r_1,r_2) = \frac{2}{3}$
- $S_e(r_1,r_2) = S_t(r_1,r_2) = \frac{4}{10}$
- $S_{rule}(r_1,r_2) = \frac{1}{3}S_s(r_1,r_2) + \frac{1}{3}S_r(r_1,r_2) + \frac{1}{3}S_e(r_1,r_2) = \frac{1}{3} \times 0.62 + \frac{1}{3} \times \frac{2}{3} + \frac{1}{3} \times \frac{4}{10} = 0.56$

4 POLICY CLUSTERING

Given a set S of objects, clustering S consisting in regrouping the objects of S into several subsets of $S : C_1, C_2, ...$, where each C_i contains objects that are similar, based on a given similarity metric. There exist many clustering algorithms, such as the K-nearest neigbors (KNN) algorithm (Bhatia et al., 2010). We propose a clustering algorithm that regroups the rules of the policy into clusters, based on the similarity score presented in Section 3. Let us say that two rules are similar if their similarity score is greater than a given threshold. Based on previous works (Lin et al., 2013; Guo, 2014), the considered threshold is 0.8. Our clustering method has been developed so that for every obtained cluster C, every rule in C is similar to at least another rule of C. That is: for every rule r_i in a cluster C, there exists another rule r_j in C such that $S_{rule}(r_i, r_j) \geq threshold$.

The inputs of our clustering algorithm are:

- a policy P which is a set of rules
- a list S_1, S_2, ... , where each S_k is the set of similarity scores depending on the rule r_k.

The algorithm proceeds iteratively as follows:

For $k = 1, 2, ...$: by analyzing S_k, we construct a new cluster consisting of r_k and all the other rules that are similar to r_k.

Then, when all S_k are treated (by the above loop), we remove every cluster that is included or equal to another cluster.

Note that the clusters resulting from our algorithm satisfy the following two properties:

- Each cluster contains at least one rule;
- Every rule is contained in one or more clusters.

Example: We consider a 4-rule policy whose similarity scores are shown in Table 1. The sets S_k are therefore:

- $S_1 = \{S_{rule}(r_1, r_2), S_{rule}(r_1, r_3), S_{rule}(r_1, r_4)\}$
- $S_2 = \{S_{rule}(r_1, r_2), S_{rule}(r_2, r_3), S_{rule}(r_2, r_4)\}$
- $S_3 = \{S_{rule}(r_1, r_3), S_{rule}(r_2, r_3), S_{rule}(r_3, r_4)\}$
- $S_4 = \{S_{rule}(r_1, r_4), S_{rule}(r_2, r_4), S_{rule}(r_3, r_4)\}$

Iteration 1: We obtain the cluster $C_1 = \{r_1, r_3\}$, because $S_{rule}(r_1, r_3)$ is the only score in S_1 which is $\geq$ 0.8. Iteration 2: We obtain the cluster $C_2 = \{r_2, r_4\}$, because $S_{rule}(r_2, r_4)$ is the only score in S_2 which is $\geq$ 0.8. Iteration 3: We obtain the cluster $C_3 = \{r_1, r_3\}$, because $S_{rule}(r1, r3)$ is the only score in S_3 which is $\geq$ 0.8. Iteration 4: We obtain the cluster $C_4 = \{r_2, r_4\}$, because $S_{rule}(r_2, r_4)$ is the only score in S_4 which is $\geq$ 0.8. Then, the clusters C_3 and C_4 are removed, because they are identical to the clusters C_1 and C_2, respectively. The constructed clusters are : $C_1 = \{r_1, r_3\}$ and $C_2 = \{r_2, r_4\}$.

Table 1: Example of computed similarity scores for 4 rules.

Pairs of rules	similarity score
(r_1, r_2)	0.041
(r_1, r_3)	0.811
(r_1, r_4)	0.111
(r_2, r_3)	0.166
(r_2, r_4)	0.866
(r_3, r_4)	0.5

5 ANOMALY DETECTION

Let an access request denotes a subject that tries to have a specific access to a resource under certain conditions (Bonatti et al., 2002; De Capitani Di Vimercati et al., 2007). Formally, an access request R is specified by an action (e.g., read, write ...) and a value for each attribute att; such value is denoted $v_{att}(R)$. We say that R matches a rule r_i (we may also say: r_i matches R), if for every attribute att of r_i , we have $v_{att}(R) \in V_{att}(r_i)$. An anomaly in a policy P is defined as the existence of access request matching several rules of P.

Statistically, the probability of anomalies between rules increases with the similarity between rules. In Section 4, the threshold of 0.8 has been used to decompose a policy into clusters, because it is estimated that most anomalies are between rules whose similarity score is ≥ 0.8 (Bhatia et al., 2010). For this reason, we propose to detect anomalies within the same cluster. We propose to classify anomalies in two categories as presented in (Khoumsi et al., 2016):

- **An Anomaly without Conflict:** occurs when there exists an access request that matches two (or more) rules that have the same action decision (i.e. X_{act}, where X is Permit or Deny to indicate that the action *act* is permitted or denied).
- **An Anomaly with Conflict:** occurs when there exists an access request that matches two (or more) rules that have different action decisions. We consider : *conflict of modalities* and *conflict of fraction permissions*. The first type occurs when two rules matched by the same access request have contradictory action decisions. The second type occurs when two rules matched by the same access request have ambiguous action decisions (e.g., $Permit_{read}$ and $Permit_{read,write}$ represent a conflict of fraction permissions).

Regarding anomalies detection, we consider the following notions:

- r_i is included in r_j (noted $r_i \subseteq r_j$), if they have the same attributes, and for every of their attributes att, we have: $V_{att}(r_i) \subseteq V_{att}(r_j)$.
- r_i is said compatible with r_j (noted $r_i \cap r_j \neq \emptyset$), if they have the same attributes, and for every of their attributes att, we have: $V_{att}(r_i) \cap V_{att}(r_j) \neq \emptyset$.

Note that if r_i and r_j are identical (which implies $S_{rule}(r_i, r_j) = 1$), then they are compatible and each one is included in the other one.

5.1 Detecting Redundancy Anomaly

Consider two rules r_i and r_j in a cluster C. We say that r_i is redundant to r_j, if removing r_i from C (while keeping r_j in C) does not change the global effect of the rules of C. Redundancies may affect the performance of a policy as well as slow down the system, because verifying if an access request respects a policy depends on the size (i.e. the number of rules) of the policy. For this reason, we consider redundancy as anomaly.

Proposition 1. *Consider a cluster C_k and two of its rules r_i and r_j whose action decisions are X_a and Y_b, respectively. r_i is redundant to r_j iff :*

1. r_i is included in r_j, and
2. $X_a = Y_b$.

Example: Consider the following rules r_1 and r_2:

- r_1: $Permit_{read}$ (Position $\in$ {Doctor, Nurse}; File_Type$\in$ {Source, Documentation}; time $\in$ [8:00, 18:00])
- r_2: $Permit_{read}$ (Position $\in$ {Nurse}; File_Type$\in$ {Documentation}; time $\in$ [8:00, 18:00])

Since r_2 is included in r_1 and the two rules have the same action decision ($Permit_{read}$), then r_2 is redundant to r_1.

5.2 Detecting Anomalies with Conflict

Consider a policy P, two rules r_i and r_j are conflicting if they can match the same profile and have different access decisions.

Proposition 2. *Given a cluster C_k and two of its rules r_i and r_j whose action decisions are X_a and Y_b , respectively. r_i and r_j are conflicting iff :*

1. r_i and r_j are compatible, and
2. $X_a \neq Y_b$.

In this paper, we consider two types of anomalies with conflict: *conflict of fraction permissions* and *conflict of modalities*.

5.2.1 Conflict of Fraction Permissions

We have a conflict of fraction permissions when in Point 2 of Proposition 2, we have $a \neq b$ and $X = Y$, i.e. the two rules permit or deny different actions. Consider the following example:

- r_1: $Permit_{read}$ (Position $\in$ {Doctor, Nurse}; File_Type$\in$ {Source, Documentation}; time $\in$ [8:00, 18:00])
- r_2: $Permit_{read/write}$ (Position $\in$ {Nurse}; File_Type$\in$ {Documentation}; time $\in$ [10:00, 18:00])

There is a conflict of fraction permissions between r_1 and r_2, because r_1 and r_2 are compatible and permit different actions (*read* for r_1, and *read/write* for r_2).

5.2.2 Conflict of Modalities

We have a conflict of modalities when in Point 2 of Proposition 2, we have $a = b$ and $X \neq Y$, i.e. an action is permitted by a rule and forbidden by the other rule. Consider the following example:

- r_1: $Deny_{read}$ (Position $\in$ {Doctor, Nurse}; File_Type$\in$ {Source, Documentation}; time $\in$ [8:00, 18:00])
- r_2: $Permit_{read}$ (Position $\in$ {Nurse}; File_Type$\in$ {Documentation}; time $\in$ [10:00, 16:00])

There is a conflict of modality between r_1 and r_2, because r_1 and r_2 are compatible while action *read* is permitted by r_2 and forbidden by r_1.

6 EVALUATION RESULTS

In order to evaluate the efficiency and effectiveness of the suggested approach, we consider synthetic datasets. The synthetic dataset is composed of the combination of eight subject attributes, four resource attributes and two environment attributes. The attribute values are inspired from real world (i.e., medical environment).

We have implemented our approach in Java and the experiments were performed on an Intel Core i5 CPU 2.7 GHz with 8 GB RAM. Figure 1 shows the running time needed to process the XACML policy entirely and output the results. The running time increases with the number of policy rules in a quadratic way. This is due to the number of the combinations being computed for policy rules during the four steps, especially in the similarity computation where we consider brute force technique to compute the similarity scores. Regarding ABAC-PC algorithm complexity, the computational time is in $O(n^2)$ where n is the number of rules.

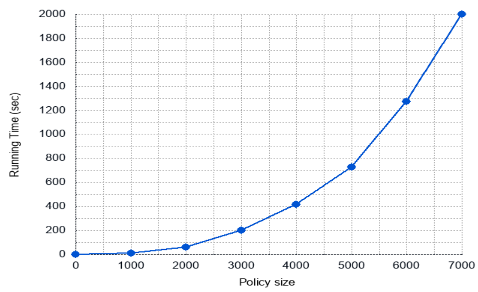

Figure 1: Running time.

Figure 2 shows the number of anomalies detected regarding each type (i.e. redundancy anomaly, conflict of fraction permissions and conflict of modalities). The number of anomalies increases with the policy size. The obtained results can be explained by the fact that with the increase of the police size, the probability of having anomalies increases.

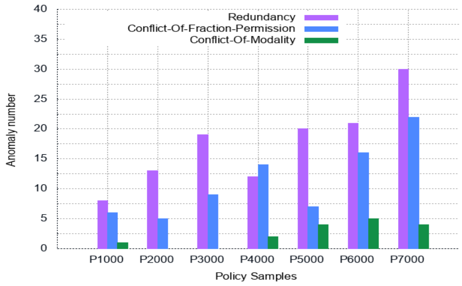

Figure 2: The number of detected anomalies.

Figure 3 shows the time gained from using clustering step as a function of policy size. To compute this metric, we run our approach without clustering. This means that the detection step is run once on the whole set of rules. Then, we compute the difference in running time between the two versions of our approach (i.e., with/without clustering). As shown in this figure, the time gained increases with the number of policy rules.

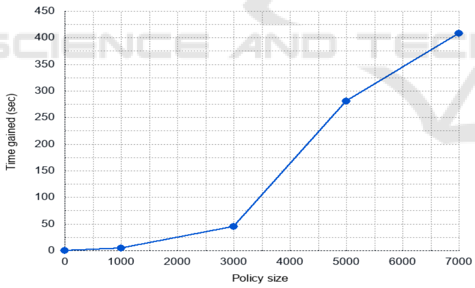

Figure 3: Time gained from clustering step.

7 CONCLUSIONS

An XACML policy for distributed applications might be aggregated from multiple stakeholders and could be managed by several administrators. Therefore, it may contain several anomalies, which may lead to high implementation complexity. In this direction, we have proposed an approach which is based on decomposing the policy into clusters before searching anomalies within each cluster. The evaluation results demonstrate the efficiency of the proposed approach to detect different types of anomalies. Directions for future work include the detection of other type of anomalies, such as inconsistency and similarity anomalies between two aggregated policies. As well as the resolution of the detected anomalies.

REFERENCES

Anderson, A., Nadalin, A., Parducci, B., Engovatov, D., Lockhart, H., Kudo, M., Humenn, P., Godik, S., Anderson, S., Crocker, S., et al. (2003). extensible access control markup language (xacml) version 1.0. *OASIS*.

Benkaouz, Y., Erradi, M., and Freisleben, B. (2016). Work in progress: K-nearest neighbors techniques for abac policies clustering. In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*, pages 72–75. ACM.

Bhatia, N. et al. (2010). Survey of nearest neighbor techniques. *arXiv preprint arXiv:1007.0085*.

Bonatti, P., De Capitani di Vimercati, S., and Samarati, P. (2002). An algebra for composing access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 5(1):1–35.

De Capitani Di Vimercati, S., Foresti, S., Samarati, P., and Jajodia, S. (2007). Access control policies and languages. *International Journal of Computational Science and Engineering*, 3(2):94–102.

Guo, S. (2014). Analysis and evaluation of similarity metrics in collaborative filtering recommender system. Master's thesis, lapland university of applied sciences.

Hu, H., Ahn, G.-J., and Kulkarni, K. (2013). Discovery and resolution of anomalies in web access control policies. *Dependable and Secure Computing, IEEE Transactions on*, 10(6):341–354.

Khoumsi, A., Erradi, M., and Krombi, W. (2016). A formal basis for the design and analysis of firewall security policies. *Journal of King Saud University-Computer and Information Sciences*.

Lin, D., Rao, P., Ferrini, R., Bertino, E., and Lobo, J. (2013). A similarity measure for comparing xacml policies. *IEEE Transactions on Knowledge and Data Engineering*, 25(9):1946–1959.

Moffett, J. D. and Sloman, M. S. (1994). Policy conflict analysis in distributed system management. *Journal of Organizational Computing and Electronic Commerce*, 4(1):1–22.

Mourad, A., Tout, H., Talhi, C., Otrok, H., and Yahyaoui, H. (2015). From model-driven specification to design-level set-based analysis of xacml policies. *Computers & Electrical Engineering*.

Stepien, B. and Felty, A. (2016). Using expert systems to statically detect" dynamic" conflicts in xacml. In *Availability, Reliability and Security (ARES), 2016 11th International Conference on*, pages 127–136. IEEE.

Yuan, E. and Tong, J. (2005). Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*, page 569.