

# ABAC Rule Reduction via Similarity Computation

Maryem Ait El Hadj<sup>1</sup>(✉), Yahya Benkaouz<sup>2</sup>, Bernd Freisleben<sup>3</sup>,  
and Mohammed Erradi<sup>1</sup>

<sup>1</sup> Networking and Distributed Systems Research Group, TIES,  
SIME Lab, ENSIAS, Mohammed V University in Rabat, Rabat, Morocco  
maryem.aitelhadj@um5.ac.ma, mohamed.erradi@gmail.com

<sup>2</sup> Conception and Systems Laboratory, FSR,  
Mohammed V University in Rabat, Rabat, Morocco  
y.benkaouz@um5s.net.ma

<sup>3</sup> Department of Mathematics and Computer Science,  
Philipps-Universität Marburg, Marburg, Germany  
freisleb@informatik.uni-marburg.de

**Abstract.** Attribute-based access control (ABAC) represents a generic model of access control that provides a high level of flexibility and promotes information and security sharing. Since ABAC considers a large set of attributes for access decisions, using it might get very complicated for large systems. Hence, it is interesting to offer techniques to reduce the number of rules in ABAC policies without affecting the final decision. In this paper, we present an approach based on K-nearest neighbors algorithms for clustering ABAC policies. To the best of our knowledge, it is the first approach that aims to reduce the number of policy rules based on similarity computations. Our evaluation results demonstrate the efficiency of the suggested approach. For instance, the reduction rate can reach up to 10% for an ABAC policy with more than 9000 rules.

**Keywords:** Clustering · K-nearest neighbors · Access control · ABAC

## 1 Introduction

Collaborative computing environments (i.e., cloud computing) bring numerous benefits, such as flexibility, scalability and reliability. While benefiting from these advantages, such systems entail multiple security risks, by exercising limited control to make information accessible to only those who are allowed to access it. In this direction, access control models represent a key component for providing security features.

Access control is concerned with determining the allowed activities of legitimate users, mediating every attempt by a user to access a resource in the system [10]. Traditionally, access control was based on user identities (Access Control Lists - ACL), or through predefined roles or groups assigned to that user (Role-based Access Control - RBAC) [18]. In the ACL model, a user is allowed to

perform an access depending on whether that user appears in the list of authorized users or not. In the RBAC model, the access decision is based on the role of the requestor (i.e., the roles associated with a hospital can include doctor, nurse, clinician, etc.). One of the advantages of the RBAC model is the fact that a given user might have multiple roles. Therefore, such a user might have different permissions according to the selected role.

Several variants of access control models have been proposed as extended version of the RBAC model. For instance, Rule-based RBAC [1] provides a mechanism to dynamically assign users to roles based on a finite set of authorization rules. The Task-Role-based Access Control (T-RBAC) model [16] is based on the concept of classification of tasks, where a task is a fundamental unit of business activity. Another variant of RBAC is the context-aware access control model [6], where the access control takes into account the context-sensitive requirements (such as time, location, or environmental state). Besides these works, the Attribute-based Access Control (ABAC) model has been suggested as a generic access control model [25]. ABAC considers a set of attributes, based on which the access decision will be taken. An attribute is assigned to a subject (e.g., user, application or process), resource (e.g., data structure, web service or system component) and environment (e.g., current time, location). These attributes may be considered as characteristics of anything that may be defined and to which a value may be assigned.

ABAC policy representation is more expressive and fine-grained, because it might consider any combination of subject, resource and environment attributes. However, in distributed environments such as cloud computing, deploying and managing an ABAC model to ensure access control might become more complex and hard to manage. This is due to the massive amount of information that should be considered as attributes. For example, considering an e-health use case, each piece of data related to the patients' medical records should be taken into account as an attribute to help deciding the types of person (determined by their attributes) having access to each individual piece of data in a given environment (location, time, etc.). In fact, an ABAC policy in distributed applications may be aggregated from multiple parties and can be managed by more than one administrator. Therefore, it may contain several redundant rules, which may lead to high implementation complexity. Hence, reducing the number of rules in ABAC policies without affecting the final decision in large sets of complex policies is primordial.

Following the idea of using K-nearest neighbors (KNN) algorithms to reduce the number of ABAC policy rules to enhance the policy analysis [4], in this paper we propose a new approach referred to as ABAC-PC (ABAC Policy Clustering). ABAC-PC works as follows: (1) First, the policy rules are grouped according to their decision effects (i.e., permit rules, deny rules), and for each group, the similarity scores of each pair of rules are computed; (2) clusters of rules are created based on the similarity scores; (3) finally, given the set of clusters, ABAC-PC produces the minimum set of rules that represent each cluster. Regarding the algorithmic complexity, the computation time is in  $O(n^2)$  where  $n$  is the number

of rules. ABAC-PC has been tested on a synthetic dataset of up to 9000 rules, and the obtained results show that ABAC-PC can successfully reduce the number of policy rules up to 10% for a policy with more than 9000 rules. In a nutshell, given an ABAC policy, ABAC-PC produces a reduced policy and guarantees the policy's conformity. Furthermore, our approach can be extended and integrated with other policy analysis tools in order to enhance managing authorization policies, such as detecting and resolving anomalies among XACML (eXtensible Access Control Markup Language) policies (since XACML is the most convenient way to express ABAC policies).

The paper is organized as follows. Section 2 presents related work. Section 3 presents the ABAC model. Section 4 gives an overview on the KNN algorithms. ABAC-PC is described in Sect. 5. Section 6 reports experimental results. Section 7 concludes the paper and outlines areas for future work.

## 2 Related Work

To the best of our knowledge, this paper presents the first approach specifically designed to reduce the number of ABAC policy rules. ABAC-PC is based on KNN, which has been often used in data mining. In the following, we present existing work that considers data mining techniques to resolve some of the access control related issues.

In the literature, several works have considered the usage of data mining for role mining to discover roles from existing system configuration data [7, 13, 19]. Role mining refers to the process of mining data about the actual user-to-resource permission assignments to extract role definitions. Molloy et al. [14] consider the problem of migrating a non-RBAC system to an RBAC system. Then, a role mining algorithm constructs an RBAC state with low cost and complexity, while maintaining the semantic meaning of roles.

Ni et al. [15] have investigated the role adjustment problem. It consists of how to automate the process that provisions existing roles with entitlements from newly deployed applications. In this direction, the authors have suggested the use of supervised machine learning algorithms to automate the process of providing users with access to data and resources.

Xu and Stoller [20] attempt to produce small RBAC policies (i.e., with low weighted structural complexity) with meaningful roles. The same authors [22] present a parameterized RBAC (PRBAC) framework, in which users and permissions have attributes, i.e., implicit parameters of the roles that can be used in role definitions.

An ABAC policy mining algorithm has been suggested by Xu and Stoller [21, 24]. This algorithm aims to reduce the cost of migration to an ABAC policy from an ACL or from an RBAC policy with accompanying attributes. Another variant of the ABAC policy mining algorithm has been presented by Xu and Stoller [23], where the authors consider the logs as attribute data. These works might be considered to detect either roles or attributes during RBAC and ABAC policies construction, whereas in our work, we consider the optimization of the ABAC policies themselves.

### 3 Attribute-Based Access Control

In this section, we briefly present the ABAC model [25]. An ABAC policy defines permissions based on predefined attributes. Attributes describe any characteristics that should be taken into account for the authorization decisions. These attributes are associated to three different entities: *Subject* (i.e., the user or the process that takes an action on a resource), *Resource* (i.e., the entity that is acted upon by a subject) and *Environment* (i.e., the operational, technical or situational context in which the information access occurs). Thus, the attributes are Subject attributes, Resource attributes and Environment attributes.

Regarding the ABAC policy formulation, we consider the following notation:

- S: Set of subjects.
- RS: Set of resources.
- E: Set of environments.
- For a given subject with  $M$  attributes:  $SA_m$  is a subject attribute with  $1 \leq m \leq M$ .
- For a given resource with  $N$  attributes:  $RSA_n$  is a resource attribute with  $1 \leq n \leq N$ .
- For a given environment with  $O$  attributes:  $EA_o$  is a environment attribute with  $1 \leq o \leq O$ .
- $ATTR(s) \subseteq SA_1 \times \dots \times SA_m$ : Attribute assignment relations for a subject  $s$ .
- $ATTR(rs) \subseteq RSA_1 \times \dots \times RSA_n$ : Attribute assignment relations for a resource  $rs$ .
- $ATTR(e) \subseteq EA_1 \times \dots \times EA_o$ : Attribute assignment relations for an environment  $e$ .

A policy  $P = \{r_1, r_2, \dots, r_n\}$  is made up of a set of rules. A rule  $r$  decides whether a subject  $s$  can access a resource  $rs$  in an environment  $e$ . To this end, a Boolean function is evaluated based on the values of all the attributes of  $s$ ,  $rs$  and  $e$ . Thus, the *Policy rule* that regulates this access is expressed as follows:

$$Rule : can\_access(s, rs, e) \leftarrow f(ATTR(s), ATTR(rs), ATTR(e))$$

The next section gives an overview of the KNN algorithm that will be used in ABAC-PC.

### 4 K-Nearest Neighbors

The K-nearest neighbor algorithm (KNN) is widely used in pattern recognition, text categorization, ranking models, and so on. The KNN algorithm classifies objects based on closest training examples in the feature space [5]. Given a system with  $N$  objects, where each object has a specific profile, a KNN algorithm provides each object with its  $K$  most similar objects, based on a given similarity metric. This results in a KNN graph where there is an edge between each object and its  $K$  most similar objects (based on the comparison of profiles).

Such a graph can be useful in the context of many applications such as similarity search [2], data mining [17] and machine learning [8]. To illustrate the use of the KNN algorithm, assume that a web-based platform is visited by multiple users to listen to different kinds of music. It would be interesting to provide a given user with items matching its interest. One approach is to look for  $K$  different users sharing similar profiles (i.e., users with the same music tastes). Then, recommend the most popular songs among the music liked by the  $K$  selected users.

The most straightforward way to construct the KNN graph is to rely on a brute-force solution computing the similarity between each pair of nodes. The similarity between nodes can be computed by several metrics, such as the cosine similarity metric or Jaccard similarity. These similarities are computed as follows:

- The cosine similarity: It is represented using the dot product and the magnitude of two vectors:

$$\text{Cosine}(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \cdot \|v_2\|}$$

- The Jaccard similarity: Given two sets of attributes, this measure is defined as the cardinality of the intersection divided by the size of the union of the two sets:

$$\text{Jaccard}(s_1, s_2) = \left| \frac{s_1 \cap s_2}{s_1 \cup s_2} \right|$$

After the presentation of the basic idea of the KNN algorithm, the next section shows how such algorithms are useful for ABAC policy optimization. Therefore, we present the suggested ABAC Policy Clustering Approach (ABAC-PC).

## 5 ABAC-PC: ABAC Policy Clustering

Let us recall that the aim of the suggested approach is to reduce the number of rules in ABAC policies. To achieve this goal, we create clusters of rules that share similar characteristics based on similarity scores. Then, for each cluster, we compute the minimum set of rules that represent each cluster. The ABAC-PC process is depicted in Fig. 1. In this section, we will first present the steps of ABAC-PC and then prove its correctness.

### 5.1 Rule Profiling

Without loss of generality, we consider that an ABAC policy consists of two categories of decision effects (permit and deny rules). Therefore, the policy base is split into two categories. Rules from each category are extracted and expressed as profiles. The general format of a profile is as follows:

$$\text{Decision\_effect}(attr\_name_1 = attr\_value_1, \dots, attr\_name_n = attr\_value_n)$$

The profile represents the combination of the whole sets of subject, resource and environment attributes (ATTR(s), ATTR(rs) and ATTR(e)). For instance,

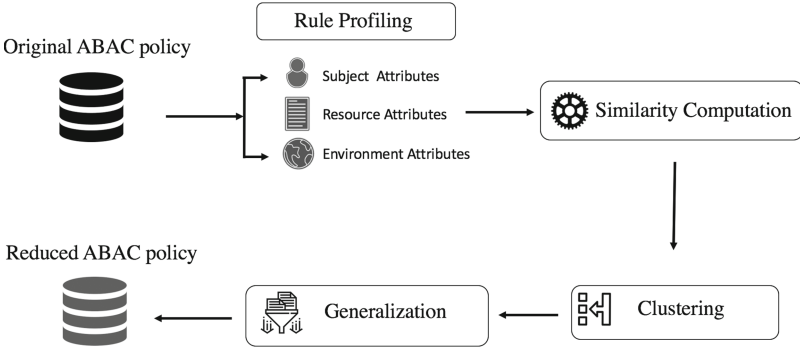


Fig. 1. ABAC-PC steps

the profile of a given rule with permit access between 12:00 and 16:00 to an MRI scan for a female nurse belonging to the oncology team, the nursing department and the Steatl organization might be expressed as:

*permit\_access* (position = {nurse}, team = {oncology}, gender = {female}, department = {Nursing}; type = {MRI}, formatType = {TXT}; Organization = {Steatl}, time in {{12-16}}).

## 5.2 Similarity Computation

The similarity measures adopted in ABAC-PC are inspired by Lin et al. [11]. The rule similarity measure assigns a similarity score  $S_{rule}$  for any two given rules, which reflects how similar these rules are with respect to subject, resource and environment attributes values. The formal definition of the rule similarity measure is given in Eq. (1), the score for each rule pair is the sum of the similarity scores of all the subject, resource and environment attributes of these rules.

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

where  $S_s$ ,  $S_{rs}$  and  $S_e$  are functions to compute similarity scores based on the Subject, Resource, and Environment attributes respectively.  $W_s$ ,  $W_{rs}$  and  $W_e$  are weights that can be chosen to reflect the relative importance to be given to the similarity computation. The weight values should satisfy the constraint:  $W_s + W_{rs} + W_e = 1$ .

The similarity score is a value between 0 and 1. Two equivalent rules are expected to obtain a similarity score that equals 1. The rule similarity measure algorithm is shown in Algorithm 1. Given two rules, the algorithm, first computes the similarity score regarding the same rule elements (subject, resource and environment). Then, the obtained scores for the different rule elements are combined according to the weights chosen in order to produce an overall similarity score.

**Algorithm 1.** Rule similarity measure algorithm**Require:**  $Elm$ : Rule Element (Subject, Resource, Environment)

- 
- 1: **ForEach** attribute  $a_i \in Elm_{r_i}$
  - 2:     **ForEach** attribute  $a_j \in Elm_{r_j}$   
        //Compute similarity of Rule Element  $iElm_i$
  - 3:  $S_{<Elm>(r_i, r_j)} = w_{Elm} S_{<Elm>(a_i, a_j)}$   
        //Compute similarity for each rule elements
  - 4:  $S_s(r_i, r_j) \leftarrow$  similarity score of Subject attributes
  - 5:  $S_{rs}(r_i, r_j) \leftarrow$  similarity score of Resource attributes
  - 6:  $S_e(r_i, r_j) \leftarrow$  similarity score of Environment attributes  
        //Compute the overall similarity score
  - 7:  $S_{rule}(r_i, r_j) = W_s S_s(r_i, r_j) + W_{rs} S_{rs}(r_i, r_j) + W_e S_e(r_i, r_j)$
- 

**Similarity Score of Rule Elements:** Each rule element *Subject*, *Resource*, and *Environment* are represented as a set of predicates in the form of:

$$(attr\_name_1 = attr\_value_1, attr\_name_2 = attr\_value_2, \dots)$$

The similarity score between two rules and regarding the same element is denoted as  $S_{<Elm>(r_i, r_j)}$ , where  $<Elm>$  refers to *Subject*, *Resource*, or *Environment*. The  $S_{<Elm>}$  is computed as the sum of the similarity scores of attribute elements (Eq. (2)).

$$S_{<Elm>(r_i, r_j)} = \sum_{k=1}^{|ATTR(Elm)|} w_{k,Elm} S_{Elm}(a_i, a_j) \quad (2)$$

where  $w_{k,Elm}$  is the weight assigned to each attribute element.  $a_i, a_j$  are attributes for  $r_i, r_j$ , respectively, regarding the same element, and  $|ATTR(Elm)|$  is the number of attribute elements being computed.

**Similarity Score of Attribute Elements:** The similarity score of attribute elements for two rules is computed only for sets of attribute elements having the same attribute names (Eq. (3)).

$$S_{<Elm>(a_i, a_j)} = \frac{|AN_i \cap AN_j|}{|AN_i \cup AN_j|} \sum_{AN_i=AN_j} S_{<att.typ>(v_i, v_j)} \quad (3)$$

where  $AN_i$  and  $AN_j$  denote the attribute names for  $a_i, a_j$  respectively.  $|AN_i \cap AN_j|$  denote the common number of attribute names and  $|AN_i \cup AN_j|$  the total number of attribute names.  $S_{<att.typ>(v_i, v_j)}$  denotes the similarity score of attribute values, where  $v_i, v_j$  are the attribute values for  $a_i, a_j$  respectively. The condition  $AN_i = AN_j$  ensures that the similarity score is only computed for sets of attribute elements having the same attribute names for both rules (i.e.  $\frac{|AN_i \cap AN_j|}{|AN_i \cup AN_j|} \neq 0$ ).

The similarity score of attribute elements algorithm is presented in Algorithm 2. Given two attribute elements, we compute the similarity score of attribute values for attribute elements having the same attribute names.

---

**Algorithm 2.** Attribute Elements Similarity Algorithm
 

---

**Require:**  $AN_i, AN_j$ : set of attribute names for  $r_i, r_j$  respectively,  $AttrElm$ : Attribute element

- 1: **ForEach** attribute value  $v_i \in AttrElm_{a_i}$
- 2:     **ForEach** attribute value  $v_j \in AttrElm_{a_j}$
- 3:     **if**  $AN_i = AN_j$  **then**  
        //Compute similarity of Attribute Element
- 4:          $S_{<Elm>}(a_i, a_j) += S_{att\_typ}(v_i, v_j)$
- 5:     **end if**
- 6:     return  $\frac{|AN_i \cap AN_j|}{|AN_i \cup AN_j|} \times S_{<Elm>}(a_i, a_j)$

---



---

**Algorithm 3.** Attribute Values Similarity Algorithm
 

---

- 1: **if**  $v_1$  and  $v_2$  are single values **then**
- 2:     **if**  $v_1 = v_2$  **then**
- 3:         return 1
- 4:     **else**
- 5:         return 0
- 6:     **end if**
- 7: **else if**  $v_1$  and  $v_2$  are bounded intervals **then**
- 8:     return  $\frac{len|v_i \cap v_j|}{len|v_i \cup v_j|}$
- 9: **else**  $v_1$  and  $v_2$  are Categorical values
- 10:     return  $\frac{|v_i \cap v_j|}{|v_i \cup v_j|}$
- 11: **end if**

---

The similarity score between attribute values differs, depending on whether their type is categorical (i.e., the string data type) or numerical (i.e., integer, real, or date/time data types). For the categorical values, we only consider the exact match of two values. The similarity is computed based on Jaccard measure (Algorithm 3).

### 5.3 Clustering

The results obtained in the similarity measures are used to classify rules into clusters, where similar rules are grouped together, while different rules belong to different groups.

Two rules  $r_i$  and  $r_j$  are similar if their similarity score is no less than a predefined threshold:  $S_{rule}(r_i, r_j) \geq threshold$ . The value of the threshold is set to 0.8, based on previous works regarding recommender systems and a similarity measure for security policies [9, 11]. Lowe [12] has proposed to use the value 0.8 reporting that this threshold allows to eliminate 90% of the false matches while discarding less than 5% of the correct matches.

Given a policy with  $n$  rules, the clustering method constructs  $k$  sets of rules with  $k \leq n$ . Rules are classified into  $k$  clusters, which satisfy the following requirements:



- Each cluster contains at least one rule.
- Each rule must belong to exactly one cluster.

Formally, a policy  $P$  is represented as a set of clusters, where  $P = \bigcup_{i=1}^k C_i$ , such that  $C_i \cap C_j = \emptyset$  for  $i \neq j$ .

#### 5.4 Generalization

After constructing clusters of rules, the purpose of this step is to compute the minimum set of rules that represent each cluster. To this end, a generalization function is applied in each cluster. This function attempts to reduce the number of policy rules, by **removing redundant rules** and **merging pairs of rules**. Let  $Rules_c$  denotes the rules in a cluster  $c$ .

**Redundancy Removal:** A rule  $r_i \in Rules_c$  is redundant if  $Rules_c$  contains another rule  $r_j$  such that:

- $r_i = r_j$  (i.e.  $S_{rule}(r_i, r_j) = 1$ ): for all the shared attribute elements of subject, resource and environment are identical. In this case, the generalization function will remove one of these two rules, or
- $r_i \subseteq r_j$ : Some of the shared attribute elements of subject, resource and environment are identical, i.e.  $r_i \cup r_j = r_j$ . In this case, the subset rule is removed (i.e.  $r_i$  is removed).

For instance, let us consider the following rules  $r_1$  and  $r_2$ :

- $r_1$ : permit\_access (Designation = {Professor, Student}; FileType = {Source, Documentation}; time = [8:00, 18:00])
- $r_2$ : permit\_access (Designation = {Professor}; FileType = {Source, Documentation}; time = [8:00, 18:00])

In this case,  $r_2$  will be removed because  $r_2$  represents the subset rule of  $r_1$  (i.e.  $r_2 \subseteq r_1$ ).

**Rule Merging:** For the merging process, two policy rules  $r_i$  and  $r_j$  are mergeable if one of their shared attribute elements of subject, resource and environment do not intersect, while the rest of the attribute elements are identical:

$$ATTR(s_{r_i}) \cap ATTR(s_{r_j}) = \emptyset \text{ or } ATTR(rs_{r_i}) \cap ATTR(rs_{r_j}) = \emptyset \text{ or } \\ ATTR(e_{r_i}) \cap ATTR(e_{r_j}) = \emptyset$$

Given two rules  $r_i$  and  $r_j$ , the merging process consists of the union of the subject, resource and environment attributes.

$$ATTR(s_{r_{merge}}) = ATTR(s_{r_i}) \cup ATTR(s_{r_j}) \\ ATTR(rs_{r_{merge}}) = ATTR(rs_{r_i}) \cup ATTR(rs_{r_j}) \\ ATTR(e_{r_{merge}}) = ATTR(e_{r_i}) \cup ATTR(e_{r_j})$$

---

**Algorithm 4.** Generalization Algorithm
 

---

```

1: ForEach rule  $r_i \in cluster_i$ 
2: // Remove subset of rules
3: subRules=  $\{r_i \in cluster_i | \exists r_j \in cluster_i \setminus r_i. r_i \subseteq r_j\}$ 
4:  $cluster_i$ .removeAll(subRules)
5: // merge of rules
6: subRules=  $\{(r_i, r_j) | r_i \in cluster_i \wedge r_j \in cluster_i\}$ 
7: if  $(S_s(r_i, r_j) = 1 \wedge S_{rs}(r_i, r_j) = 1) \vee (S_s(r_i, r_j) = 1 \wedge S_e(r_i, r_j) = 1) \vee (S_{rs}(r_i, r_j) = 1 \wedge S_e(r_i, r_j) = 1)$  then
8:    $r_{merge} = (ATTR(s_{r_1}) \cup ATTR(s_{r_2}), ATTR(rs_{r_1}) \cup ATTR(rs_{r_2}), ATTR(e_{r_1}) \cup ATTR(e_{r_2}))$ 
   // Add the merged rule to  $cluster_i$ 
9:    $cluster_i$ .add( $r_{merge}$ )
   // remove rules that become subset of  $r_{merge}$ 
10:  subRules=  $\{r_k \in cluster_i | r_k \subseteq r_{merge}\}$ 
11:   $cluster_i$ .removeAll(subRules)
12: end if
13: return  $cluster_i$ 
    
```

---

$r_{merge}$ , which is the merging rule, is added to  $Rules_c$ , while rules being merged (i.e.,  $r_i$  and  $r_j$ ) are removed from  $Rules_c$ .

For instance, let us consider the following rules  $r_1$  and  $r_2$ :

- $r_1$ : permit\_access (Designation = {Professor, Student}; FileType = {Source, Documentation}; time = [8:00, 12:00])
- $r_2$ : permit\_access (Designation = {Professor, Student}; FileType = {Source, Documentation}; time = [12:00, 18:00])
- $r_{merge}$ : permit\_access (Designation = {Professor, Student}; FileType = {Source, Documentation}; time = [8:00, 12:00]  $\cup$  [12:00, 18:00])

Given a cluster of rules  $cluster_i$ , the generalization function in Algorithm 4 attempts to update rules in  $cluster_i$ . It eliminates redundant rules and merges pairs of rules. If there is a valid generalization, the algorithm outputs an updated cluster with minimum rules. Otherwise, the algorithm outputs the same cluster.

## 5.5 Correctness

The aim of this subsection is to prove the correctness of the suggested approach.

Given an original policy base  $OP$  where  $OP = \bigcup_{i=1}^n r_i$ , we want to find a reduced

policy base  $RP = \bigcup_{j=1}^m r_j$  which is conform to  $OP$  with  $n \geq m$ . In order to prove the correctness of the suggested approach, first, we define the concepts of *Conformity* and *Access Domain*.

**Definition 1 (Conformity).**

An original policy  $OP$  conforms to a reduced policy  $RP$  if  $OP$ 's access domain is equivalent to  $RP$ 's access domain:  $AD_{OP} \equiv AD_{RP}$ .

**Definition 2 (Access Domain).**

Given a rule  $r(s, rs, e) \leftarrow f(ATTR(s), ATTR(rs), ATTR(e))$ , the access domain of  $r$  denoted  $AD_r$  is defined as the set of all possible combinations of  $ATTR(s)$ ,  $ATTR(rs)$  and  $ATTR(e)$ . Therefore, the access domain of the global policy  $P$  is  $AD_P$ , defined by the union of the access domains of the individual rules of  $P$ .

**Theorem 1.** Consider an original policy base  $OP$  with  $n$  rules. ABAC-PC produces a reduced policy base  $RP$  with  $m$  rules conforming to  $OP$  with  $n \geq m$ .

*Proof.* ABAC-PC is composed of four steps (i.e., rules profiling, similarity computation, clustering and generalization). During the first three steps, the policy conformity holds, since no change is performed on attribute values. In the fourth step, two actions are performed: (1) redundancy removal and (2) rule merging.

Let  $r_i$  and  $r_j$  be two rules in  $OP$ . Thus, the union of their access domains belong to  $OP$ 's access domain.

$$(r_i, r_j) \subseteq OP \implies AD_{r_i} \cup AD_{r_j} \subseteq AD_{OP}$$

*(1) Redundancy removal*

In case  $r_i = r_j$  (i.e.  $AD_{r_i} = AD_{r_j}$ ), either  $r_i$  or  $r_j$  is removed. Let  $r_j$  be the removed one.

$$r_i \subseteq RP \implies AD_{r_i} \subseteq AD_{RP}$$

While  $AD_{r_i} = AD_{r_j}$ , the access domain represented by the reduced policy  $RP$  is the same as  $OP$ 's access domain.

In case  $r_i \subseteq r_j$  (i.e.  $AD_{r_i} \subseteq AD_{r_j}$ ),  $r_i$  represents the subset of  $r_j$ . Thus,  $r_i$  is the removed rule.

$$r_j \subseteq RP \implies AD_{r_j} \subseteq AD_{RP}$$

Since our function keeps the superset of rules (i.e.,  $r_j$ ), the access domain selected is the general one. Therefore, the access domain represented by the reduced policy  $RP$  is the same as  $OP$ 's access domain.

*(2) Rule merging*

Let  $r_i$  and  $r_j$  are good candidates for the merging process (i.e., two of their attribute elements are matched) and  $r_{merge}$  be the merging rule.  $r_{merge}$  represents the union of the subject, resource and environment attributes of  $r_i$  and  $r_j$ :

$$\begin{aligned} ATTR(s_{r_{merge}}) &= ATTR(s_{r_i}) \cup ATTR(s_{r_j}) \\ ATTR(rs_{r_{merge}}) &= ATTR(rs_{r_i}) \cup ATTR(rs_{r_j}) \\ ATTR(e_{r_{merge}}) &= ATTR(e_{r_i}) \cup ATTR(e_{r_j}) \end{aligned}$$

While  $r_{merge} = r_i \cup r_j$ , the access domain specified by  $r_{merge}$  is the union of the access domains of  $r_i$  and  $r_j$  (i.e.,  $AD_{r_{merge}} = AD_{r_i} \cup AD_{r_j}$ ).  $r_i$  and  $r_j$  are removed from the resulting policy. Thus, the access domain represented by the reduced policy  $RP$  is the same as  $OP$ 's access domain.

$$r_{merge} \subseteq RP \implies AD_{r_{merge}} \subseteq AD_{RP}$$

Therefore, for all performed actions we guarantee that the access domain specified by the  $OP$  is the same specified by  $RP$ . Thus, the policy  $RP$  conforms to  $OP$ .

*Complexity analysis:* Let  $n$  be the number of rules of a policy,  $K$  the number of clusters and  $n_k$  the number of rules in a cluster  $k$ . The running time of the rule profiling step is in  $O(n)$ . During the similarity computation step, we use brute force approach to compute the rules similarities. Thus, the complexity is  $O(n^2)$ . The running time of the clustering step is in  $O(C_n^2)$  since every pair of  $n$  rules is explored to construct clusters. The running time of the last step (generalization) in the worst is  $O(K \times n_k^2)$ . Therefore, the overall time complexity for constructing a reduced policy is in  $O(n^2)$ .

## 6 Experimental Results

To evaluate ABAC-PC, we consider a synthetic dataset composed of the combination of eight subject attributes, four resource attributes and two environment attributes. Evaluation on policies from real organization would be perfect. Unfortunately, no benchmarks have been published in this area and real medical data are hard to obtain because of confidentiality constraints.

The number of rules varies between 1000 and 10000. ABAC-PC has been implemented in Java and the experiments were made using a laptop with a 2.7 GHz Intel Core i5 CPU, 8 Gb RAM.

As comparison metrics, we mainly consider the running time, the size of the resulted policy and the reduction rate. Figure 2 shows the running time as a function of policy size. The graph explicitly shows that the running time increases with the number of policy rules in a quadratic way. This is related to the number of the combinations generated and being treated for policy rules during the ABAC-PC steps, especially in the similarity computation.

Figure 3 shows the size of reduced policy based on the similarity threshold. The obtained results represent the evaluation of the ABAC-PC on three bases with different number of policy rules (1000, 3000 and 5000 rules). As depicted in this figure, the lowest threshold (i.e., 0.6) returns a very reduced policy. The threshold effect is negligible from the value 0.8, where the reduced policy size is close to the original policy size. Therefore, the default value of the threshold selected for our experiments is 0.8.

Figures 4 and 5 show the reduction rate for five policy bases with different number of rules. The reduction rate increases significantly for large policies. The obtained results can be explained by the fact that the probability of having redundant and mergeable rules increases.

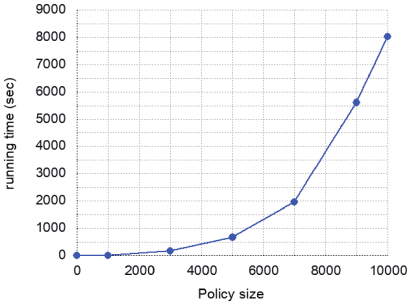


Fig. 2. Running time

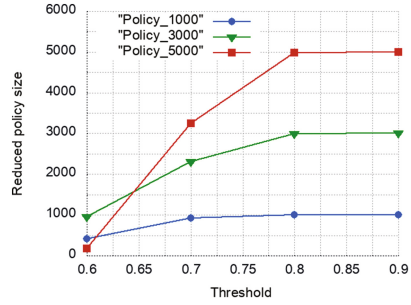


Fig. 3. Reduced policy size vs. threshold

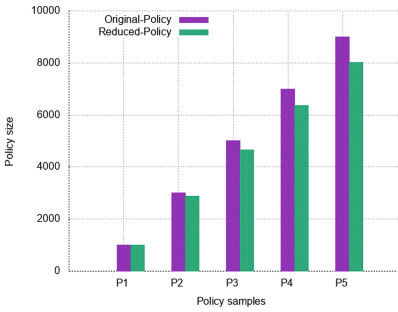


Fig. 4. Original and reduced policy sizes

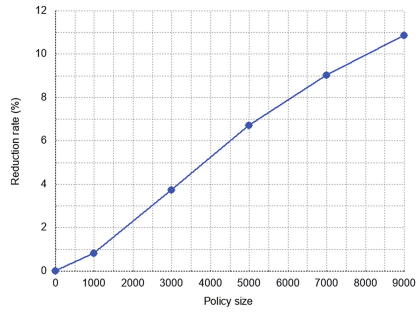


Fig. 5. Reduction rate

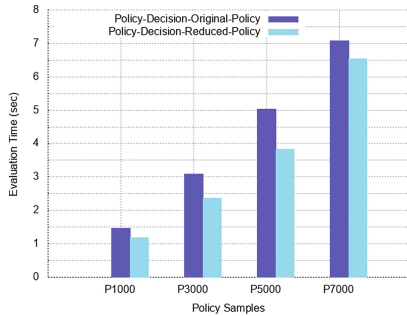


Fig. 6. Policy decision evaluation time

In order to evaluate the impact of the reduced policy on the functionality of ABAC, we consider the policy decision time (i.e., the time required for sending the final decision regarding a given request). The policy decision time is directly related to the size of the overall policy. Hence, reducing the number of rules will be beneficial for the policy decision performance.

The policy decision evaluation is depicted in Fig. 6. The obtained results represent the time evaluation for a request on four policies (original and reduced policies). The experiments were made on *Balana* [3] as the Open source XACML 3.0 implementation. As depicted in this figure, the time required for the decision on original policies is higher than the one on reduced policies. The difference can be explained by the fact that our approach reduces the policy size by eliminating redundant rules and merging pairs of rules. Therefore, the PDP takes more time to evaluate the request in a policy that contains more rules.

## 7 Conclusion

Since an ABAC policy is based on combination of subject, resource and environment attributes for access decisions, its policy representation is rich and expressive. However, using the ABAC model for large systems might generate a large number of security policy rules. In this direction, we have presented an approach that reduces the number of the ABAC policy rules. The suggested approach is mainly based on the K-nearest neighbors algorithm. The evaluation results demonstrate the efficiency of the proposed approach. Directions for future work include the integration of the suggested ABAC-PC in a real world policy analysis project.

## References

1. Al-Kahtani, M.A., Sandhu, R.: Induced role hierarchies with attribute-based RBAC. In: Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies, pp. 142–148. ACM (2003)
2. Amato, G., Falchi, F.: kNN based image classification relying on local feature similarity. In: Proceedings of the Third International Conference on Similarity Search and Applications, pp. 101–108. ACM (2010)
3. Balana: Open source xacml 3.0 implementation (2012). <http://xacmlinfo.org/2012/08/16/balana-the-open-source-xacml-3-0-implementation/>
4. Benkaouz, Y., Erradi, M., Freisleben, B.: Work in progress: K-nearest neighbors techniques for ABAC policies clustering. In: Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control, pp. 72–75. ACM (2016)
5. Bhatia, N., et al.: Survey of nearest neighbor techniques. arXiv preprint [arXiv:1007.0085](https://arxiv.org/abs/1007.0085) (2010)
6. Bhatti, R., Bertino, E., Ghafoor, A.: A trust-based context-aware access control model for web-services. *Distrib. Parallel Databases* **18**(1), 83–105 (2005)
7. Ene, A., Horne, W., Milosavljevic, N., Rao, P., Schreiber, R., Tarjan, R.E.: Fast exact and heuristic methods for role minimization problems. In: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, pp. 1–10. ACM (2008)
8. Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K.: KNN model-based approach in classification. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) OTM 2003. LNCS, vol. 2888, pp. 986–996. Springer, Heidelberg (2003). doi:[10.1007/978-3-540-39964-3\\_62](https://doi.org/10.1007/978-3-540-39964-3_62)
9. Guo, S.: Analysis and evaluation of similarity metrics in collaborative filtering recommender system. Master's thesis. Lapland University of Applied Sciences (2014)

10. Hu, V.C., Ferraiolo, D., Kuhn, D.R.: Assessment of access control systems. US Department of Commerce, National Institute of Standards and Technology (2006)
11. Lin, D., Rao, P., Ferrini, R., Bertino, E., Lobo, J.: A similarity measure for comparing XACML policies. *IEEE Trans. Knowl. Data Eng.* **25**(9), 1946–1959 (2013)
12. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
13. Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S., Lobo, J.: Mining roles with semantic meanings. In: *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, pp. 21–30. ACM (2008)
14. Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S., Lobo, J.: Mining roles with multiple objectives. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **13**(4), 36 (2010)
15. Ni, Q., Lobo, J., Calo, S., Rohatgi, P., Bertino, E.: Automating role-based provisioning by learning from examples. In: *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, pp. 75–84. ACM (2009)
16. Oh, S., Park, S.: Task-role-based access control model. *Inf. Syst.* **28**(6), 533–562 (2003)
17. Pan, R., Dolog, P., Xu, G.: KNN-based clustering for improving social recommender systems. In: Cao, L., Zeng, Y., Symeonidis, A.L., Gorodetsky, V.I., Yu, P.S., Singh, M.P. (eds.) *ADMI 2012*. LNCS, vol. 7607, pp. 115–125. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-36288-0\\_11](https://doi.org/10.1007/978-3-642-36288-0_11)
18. Sandhu, R.S., Coynek, E.J., Feinstein, H.L., Youmank, C.E.: Role-based access control models yz. *IEEE Comput.* **29**(2), 38–47 (1996)
19. Vaidya, J., Atluri, V., Guo, Q., Adam, N.: Migrating to optimal RBAC with minimal perturbation. In: *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, pp. 11–20. ACM (2008)
20. Xu, Z., Stoller, S.D.: Algorithms for mining meaningful roles. In: *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, pp. 57–66. ACM (2012)
21. Xu, Z., Stoller, S.D.: Mining attribute-based access control policies from RBAC policies. In: *2013 10th International Conference and Expo on Emerging Technologies for a Smarter World (CEWIT 2013)*, pp. 1–6. IEEE (2013)
22. Xu, Z., Stoller, S.D.: Mining parameterized role-based policies. In: *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, pp. 255–266. ACM (2013)
23. Xu, Z., Stoller, S.D.: Mining attribute-based access control policies from logs. In: Atluri, V., Pernul, G. (eds.) *DBSec 2014*. LNCS, vol. 8566, pp. 276–291. Springer, Heidelberg (2014). doi:[10.1007/978-3-662-43936-4\\_18](https://doi.org/10.1007/978-3-662-43936-4_18)
24. Xu, Z., Stoller, S.D.: Mining attribute-based access control policies. *IEEE Trans. Dependable Secure Comput.* **12**(5), 533–545 (2015)
25. Yuan, E., Tong, J.: Attributed based access control (ABAC) for web services. In: *IEEE International Conference on Web Services (ICWS 2005)*. IEEE (2005)