Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105
Volume **8**, Number **3**, August **2023**

Policy Extraction and Optimization with Access Logs for Attribute-based Access Control

Wei Sun*
School of Computer and Information Technology
Xinyang Normal University
Xinyang 464000, P. R. China
sunny810715@xynu.edu.cn

Long Li
Guangxi Key Laboratory of Trusted Software
Guilin University of Electronic Technology
Guilin 541004, P. R. China
lilong@guet.edu.cn

Ying Hu
Department of Electric Engineering
Pacific Gas and Electric Company
San Ramon 94583, USA
huying0902@gmail.com

*Corresponding author: Wei Sun
Received March 19, 2023, revised May 13, 2023, accepted June 16, 2023.

Abstract. *The policy engineering technology based on attribute-based access control is commonly used to derive rules from the existing authorization information such as access logs. However, most conventional methods extract rules that do not support negative attribute expressions, and there are numerous redundant and inaccurate rules. This study proposes a novel policy-engineering method to extract and optimize the ABAC policy based on access logs. First, to enhance the policy interpretability while reducing the engineering scale, the clustering technology is utilized to partition access logs into different rule clusters, and an algorithm is presented to extract effective rules supporting both the positive and negative attribute conditions. Second, to guarantee the policy quality, an optimization algorithm is proposed to repeatedly correct the extracted rules according to different policy categories, which improves the policy accuracy and reduces the complexity of the policy engineering. Last, the evaluation criteria of the policy quality are presented based on principles of the correctness and conciseness, and the efficiency and effectiveness of the proposal are demonstrated through experiments. The experimental results show that compared to the existing studies, the proposed method not only enhances the policy interpretability but also guarantees the policy quality.*
Keywords: Attribute-based access control, Policy extraction, Policy optimization, Policy quality, Access logs

1. **Introduction.** With the rapid advances and wide applications of the emerging network-information and computing technologies such as the Internet of Things (IoT) [1–3], blockchain and edge computing, traditional access control mechanisms cannot realize the authorization requirements in the fine-grained application scenarios. The attribute-based access control (ABAC) overcomes the limitations of the existing models and provides a flexible way to capture the access requests of large-scale complex and dynamic systems [4]. To successfully implement the ABAC mechanism, it is necessary to determine an

appropriate authorization policy and build a good access control system. For this purpose, the policy engineering technology based on ABAC was proposed [5, 6], which was fallen into two construction approaches: The top-down and bottom-up. Compared to the time-consuming, error-prone and labor-intensive processing way for the former, the latter adopted an automatic or semi-automatic way to derive rules, which could reduce the cost, time and errors during process of the policy construction and system development. The bottom-up approach for ABAC policy engineering, often referred to as policy extraction, has attracted much attention and interest for both academia and industry in recent few years [7, 8].

Usually, the policy-extraction method attempts to derive rules from the existing authorization information such as access logs or access lists. Different extraction methods have been proposed. Xu and Stoller [9] were first to study the ABAC policy extraction problem from a given access control lists or log records and proposed a bottom-up method that is simply represented as the Xu-Stoller. Das et al. [10] showed that there existed the similarity for developing different systems between the policy engineering based on ABAC and the role engineering based on role-based access control (RBAC), and they also presented a survey that discussed how to resolve such two engineering problems in detail. To make the results of the policy extraction more interpretable, it is necessary to aggregate subjects, resources and other entities that have the same or similar characteristics [11]. However, the engineering scale is very large, and the extraction process becomes more complex as the number of attribute properties increases using the traditional methods. To reduce the scale of the policy engineering, Das et al. [12] presented a policy mining scheme using the Gini impurity method. Subsequently, to further minimize the number of the mined rules, Das et al. [13] visually represented a given authorization matrix, and then proposed a novel rule-extraction method to mine ABAC rules from the rearranged matrix, which is simply denoted as VisMAP. To visually specify ABAC rules while detecting conflicts, Zheng et al. [14] converted the rules into a set of binary sequences and proposed a new policy engineering method. To further reduce the policy-engineering scale, Sun et al. [15] adopted the partitioning and compressing technologies and pro-posed a novel optimization method.

The ABAC mechanism is very flexible due to its powerful policy expressiveness. Essentially, a policy can contain several authorization rules, and each rule is comprised of different combinations of the attribute–value pairs of entities, as well as operations. Actually, ABAC rules with various kinds of attribute conditions can enhance the policy interpretability and further enrich the policy expressiveness. Note that the variant and number of attribute-condition expressions in any extracted rule are very important for realizing various requirements such as the permitted or denied authorization, and they also affect the time and result of policy decisions. Das et al. [16] combined the top-down and bottom-up approaches and proposed a novel method to construct ABAC rules, in order to improve the efficiency of the policy engineering. As a matter of fact, the authorization rules for ABAC with both the positive and negative action conditions may be more flexible and convenient. For this purpose, Iyer et al. [17] presented a novel policy-extraction method based on ABAC. To reduce the number of the policy rules, John et al. [18] attempted to derive rules that satisfied access requests in the multi-cloud environment, and they presented a new solution. However, most existing methods extract the authorization rules that only support positive condition expression, and they do not discuss the attribute expressions with the negative operation type.

The most critical step for the construction of an ABAC system is to extract an appropriate authorization policy, such that the authorization result produced by the extracted rules is consistent with the original access mode, while the extracted policy should be

as concise and correct as possible. Talukdar et al. [19] converted the policy extraction into to the problem of the functional dependencies for a given relational database and then presented an algorithm named as ABAC-FDM in order to extract ABAC policies. However, the computational complexity of the algorithm was very high. Subsequently, to minimize the set of the extracted rules, the authors proposed an alternative algorithm named as ABAC-SRM for extracting suitable rule set from the candidates. To meet the collaborative access requests in multi-cloud environments, John et al. [20] presented the definition of the rule-extraction problem called CDRMP and then proposed the resolution, in order to discover a minimal policy set. To further suit complex requirements under the dynamic collaboration environment with cross domains, they also defined a policy extraction problem and presented a heuristic solution for extracting the minimal set of authorization rules [21]. To improve the effectiveness of the policy extraction and reduce the complexity of policy engineering, Gautam et al. [22] associated the property of each attribute within any rule to a specific weight and presented a constrained rule-extraction algorithm to construct policy rules from the given authorization matrix, while minimizing the sum of the total weights of the rules. To guarantee the succinctness and correctness of the policy engineering, Cotrini et al. [23] presented a new policy engineering method to derive rules from the sparse logs and considered the reliability as a new evaluation criterion of the policy quality. However, there are numerous redundant and incorrect ABAC rules extracted by the existing methods. Inconsistent or incorrect policy decisions may result in previously authorized access requests being denied, or previously unauthorized requests being allowed, which reduce the policy quality.

To resolve the above-mentioned problems, this study proposes a novel policy-engineering method, which is called ABAC policy extraction and optimization based on access logs. The main contribution points of this study are generalized as follows:

(1) To enhance the policy interpretability, while reducing the scale of the policy engineering, we utilize the clustering partition technique to construct an initial rule set, and then we present an algorithm to extract effective attribute conditions including the positive and negative from the rule clusters during the policy-extraction process. We also demonstrate the efficiency of the proposal through experiments.

(2) To improve the policy accuracy, while reducing the complexity of the policy engineering, we further pro-pose a policy optimization algorithm to repeatedly correct the extracted rules and improve the policy quality according to different policy categories. We also present the criteria of evaluating the policy quality based on principles of the correctness and conciseness and demonstrate the effectiveness of the proposal through experiments.

The rest of the article is structured as follows. The preliminaries used for our work are discussed in Section 2. Section 3 proposes a novel policy-engineering method with access logs, including three phases: Preprocessing, policy extraction, and policy optimization. We implement experiments and comprehensively present the eval-uation analysis in Section 4, and we conclude the article and discuss future work in Section 5.

2. **Preliminaries.**

Before proposing our methodology, some necessary preliminaries are discussed, which include the basic elements of ABAC, the ABAC policy specification with access logs, and the clustering method.

2.1. **Basic Elements of the ABAC Model.**

The basic ABAC model [4] mainly consists of the following sets, relationships, and functions:

(1) *U, O, S,* and *OP* represent the finite sets of requesting subjects (or users), requested objects, environments, and operations, respectively;

(2) A_u, A_o, A_s represent the attribute sets of user u, object o, and session s, respectively;

(3) E and A represent the finite sets of all the entities and entity attributes in the system, where $E = U \cup O \cup S$, $A = A_U \cup A_O \cup A_S$;

(4) V_a represents a finite set of all the possible values that attribute a can take, and *fa_e(e,a)* represents a function that returns the values of attribute a that entity e takes.

Besides, the expression of the attribute value pair is represented as the two-tuple form $< a, \nabla v >$ in this work, where a is the attribute name, v is the corresponding attribute value, and the relation-operator set $\nabla = \{" = ","!"," > "," < "\}$ is used to denote the relationship between attribute a and value v. For example, *¡a,=v¿* indicates that a can take v, which is regarded as the positive attribute expression and is simply denoted as *¡a,v¿*. Similarly, *¡a,!v¿* indicates that a can take a value except v, which is regarded as the negative expression. For convenience of the specifications, we only discuss the first two relation operators in this work. Further, AC is used to represent the set of all the attribute conditions, and EAV is used to represent the assignment relationship between all the entities and attribute conditions.

2.2. **ABAC Policy Specification with Access Logs.** The basic ABAC policy specification with the access logs [17], which is represented as π, mainly involves the following components:

(1) Access request rq: It is a quadruple form $< u, o, s, op >$, which indicates that user u requests to access and perform operation op on object o in session s.

(2) Authorization decision function $d_\pi(rq)$: It represents the decision of the authorization policy π for a given request rq, where $d_\pi(rq) = permitted$ indicates that the subject is allowed to access the object, and $d_\pi(rq) = denied$ indicates that the access is not allowed.

(3) Access logs AL: An access record is a two-tuple form $< rq, d >$, where rq indicates the access request, d indicates the authorization decision for rq, and the value of d can take permitted or denied. The record with the permitted decision indicates that the subject is allowed to perform an operation on the object, which is regarded as the positive log; otherwise, the user is not allowed to access the resources, such record is regarded as the negative log. Thus, set AL of all the access logs contains both the positive AL^+ and negative AL^-, which are represented as:

$AL^+ = \{< rq, d > | < rq, d > \in AL \, and \, d = permitted\}$, where AL^+ represents the set of all the positive log records;

$AL^- = \{< rq, d > | < rq, d > \in AL \, and \, d = denied\}$, where AL^- represents the set of all the negative log records; $AL = AL^+ \cup AL^-$.

(4) Authorization rule ρ: It is a two-tuple form $< AC, op >$, where AC indicates the attribute conditions, and op indicates the operating action. The set of all the authorization rules is represented as $P = \rho_1, \rho_2, \ldots$.

(5) Policy categories: According to whether or not the extracted policy is consistent with the original policy, the policy can be divided into different categories. For the same access request, if the authorization decision of the original policy is permitted, then that of the extracted also should be permitted. It is regarded as the false negative policy once the permitted access request is denied by the extracted policy, which is simply denoted as π_{FN}. Similarly, π_{FP} indicates that the original denied access request is permitted by the extracted policy, π_{TP} indicates that the original and extracted policies are both permitted, while π_{TN} indicates that the original and extracted ones are both denied for the same access request.

2.3. **Clustering Method.** The clustering technique using the unsupervised learning method can specify the structure of sample data, particularly for the categorical data with no labels. In order to mine and extract a high-quality ABAC policy from the given

access logs, the process of the policy extraction can be regarded as the mapping from the log set to a set of clusters that represent ideal ABAC rules [24]. Such mapping relationship can be represented as a function $f : X \to Y$, where X is a set of authorization-tuple records, and Y is a set of cluster labels.

The authorization records in any cluster should have similar characteristics, since each cluster label of Y corresponds to a single ABAC rule. To obtain the similarity or distance among sample clusters, the Jaccard coefficient is widely used to identify the clusters with the same or similar attribute features [25]. Thus, given set $S = S_a, S_b, \ldots, S_i, \ldots$, where $S_a = a_1, a_2, \ldots, S_b = b_1, b_2, \ldots, S_i = i_1, i_2, \ldots$, the similarity and distance among samples are calculated as follows.

(1) $\forall(S_i, S_j) \in S$ the similarity and distance between samples S_i and S_j are expressed as:

$$sim(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \tag{1}$$

$$dis(S_i, S_j) = 1 - sim(S_i, S_j) \tag{2}$$

(2) $\forall(S_{j1}, S_{j2}, \ldots \in S)$ the similarity and distance between a single sample S_i and a sample set $\{S_{j1}, S_{j2}, \ldots\}$ are expressed as:

$$sim(S_i, \{S_{j1}, S_{j2}, ...\}) = \frac{1}{|\{S_{j1}, S_{j2}, ...\}|} \sum_{S_j \in \{S_{j1}, S_{j2}, ...\}} sim(S_i, S_j) \tag{3}$$

$$dis(S_i, \{S_{j1}, S_{j2}, ...\}) = 1 - sim(S_i, \{S_{j1}, S_{j2}, ...\}) \tag{4}$$

3. **Methodology.** In this section, we propose a novel policy-engineering method, and it involves three phases: (1) Preprocessing, including the representation and conversation for access information, as well as the cluster partitioning for the access logs, (2) ABAC rule extraction, and (3) policy optimization. Specifically, entities as well as the relationships among them in the logs are first formally represented with the basic sets and functions, and all the numerical variables are converted into the corresponding categorical variants. Subsequently, the access logs associated with the authorized rules are determined, and the appropriate clustering technique is used to divide the logs into different clusters in the preprocessing phase. Next, the similar patterns are searched based on the authorization features of the log records, from which the effective attribute conditions that are combinations of different attribute–value pairs are extracted, in order to construct the initial rules in the extraction phase. Compared to the original ABAC policy, rules extracted from the original access logs may be too restricted or too relaxed. Thus, these restricted and relaxed rules need to be corrected, in order to further optimize the extracted rules. Last, we evaluate the policy quality through experiments according to the accuracy and complexity criteria. The flow chart of the proposal is presented in Figure 1.

3.1. **Preprocessing.** As an important method to analyze the clustering problem, the partition is comprehensively used in scientific research and production practice due to its convenience and accuracy characteristics. Compared with other initialization methods, this method initializes the clustering based on the density and distance, which is more robust and accurate.

Since the partition technique is an unsupervised learning algorithm for clustering the categorical type, all the numerical variables need to be converted into the corresponding variants. In the ABAC system, the session attribute is related to dynamic factors such as time, location, and the access-control scenario. Thus, in the pre-processing, these continue attribute variables are divided into the categorical ones, in order to extract the standard ABAC policy. Moreover, the default-value problem of the attribute condition

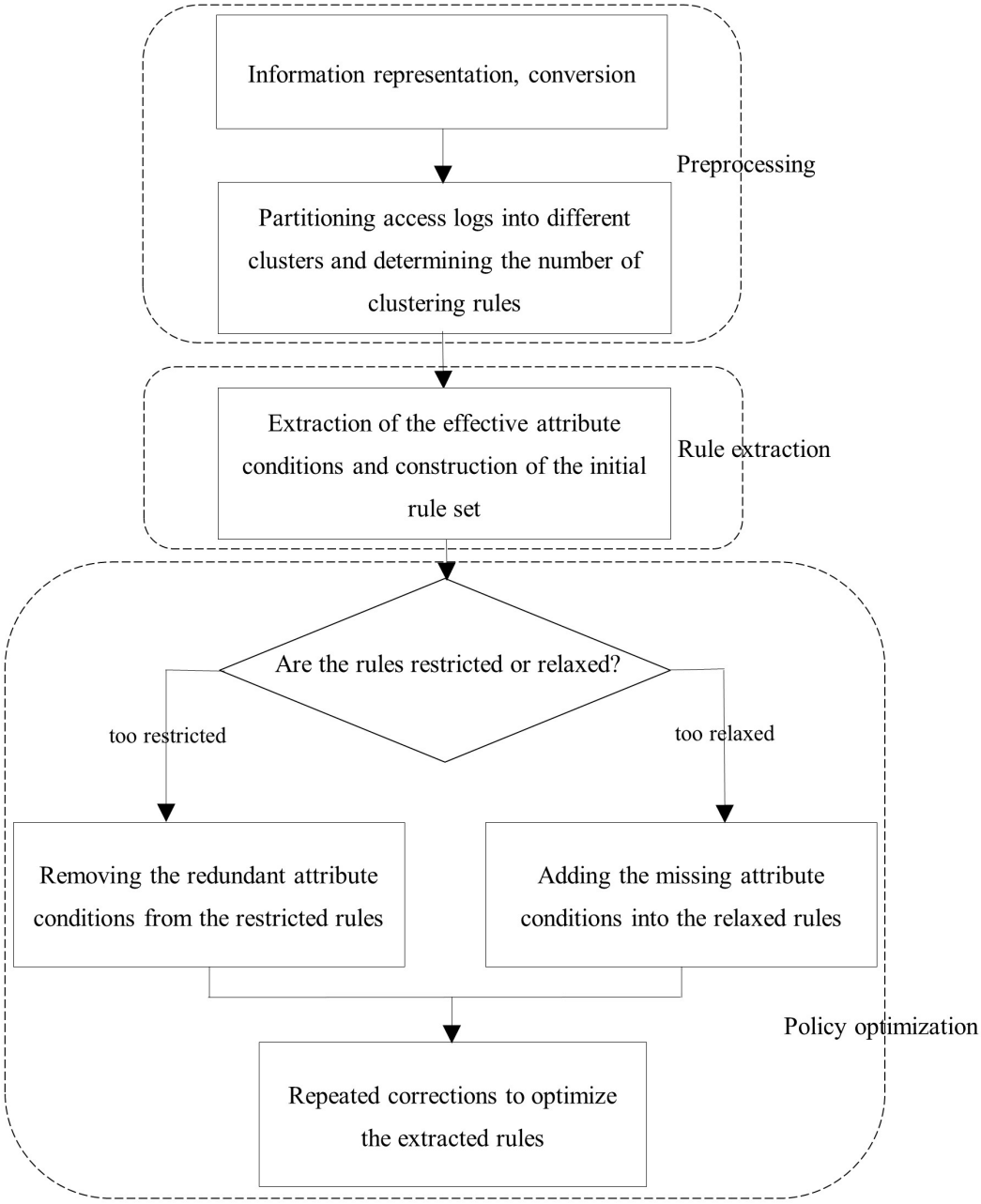

Figure 1. Flow chart of the proposed method

also needs to be addressed, as the frequency of each attribute condition is an important factor in the rule extraction algorithm, which is used to determine whether the attribute is valid. Thus, we assume that each missing value is replaced with the unknown value simply denoted as "UNK".

To make each cluster that contains several log records correspond to a single ABAC rule, the process of partitioning the logs AL is presented as follows:

Step 1. Similar to the k-means algorithm, the clustering method of partitioning around medoids (PAM) is used to divide the access logs into k different partitions such as $c_1, c_2, \ldots, c_k$, from which k initial medoids are randomly selected.

Step 2. The distance of the cluster medoid ali to the non-medoid log records is computed as follows:

$$dis(al_i, associate(al_i)) = 1 - \frac{1}{|associate(al_i)|} \sum_{al_j \in associate(u_i)} sim(al_i, al_j) \tag{5}$$

Similarly,

$$dis(al_j, associate(al_i)\backslash\{al_j\} \cup \{al_i\}) = 1 - \frac{1}{|associate(al_i)\backslash\{al_j\} \cup \{al_i\}|} \sum_{al_k \in (associate(al_i)\backslash\{al_j\} \cup \{al_i\})} sim(al_j, al_k) \tag{6}$$

where $sim(al_i, al_j) = \frac{\sum_{AC_e \in \{AC_U, AC_O, AC_S\}} |AC_e{}^{al_i} \cap AC_e{}^{al_j}|}{\sum_{AC_e \in \{AC_U, AC_O, AC_S\}} |AC_e{}^{al_i} \cup AC_e{}^{al_j}|}$ and function *associate(al_i)* represents all the non-medoid records associated with the cluster medoid al_i.

Step 3. Compare $dis(al_i, associate(al_i))$ with $dis(al_j, associate(al_i)\backslash\{al_j\}\cup\{al_i\})$, and then judge whether to exchange al_i with al_j and determine the new medoid, such that $\forall al_j \in c\backslash\{al_i\} : dis(al_i, c\backslash\{al_i\}) < dis(al_j, c\backslash\{al_j\})$.

Step 4. For different values of k, the PAM method is run repeatedly, and the accuracy and error rate of the model are computed. The number k of the partitions that can better balance the relationship of the policy accuracy and complexity is selected as the number of the initial ABAC rules.

3.2. **ABAC Rule Extraction.** To extract the attribute conditions from the similar recording characteristics, the effective attribute–value pairs including the positive and negative are defined first.

Definition 1. Effective positive (or negative) attribute–value pair: Let all the possible attribute–value pairs be $S = < a, v|!v >$. $< a, v >$ is regarded as the effective positive attribute–value pair of rule $\rho =< AC, op >$ with respect to cluster c_i, if and only if the difference between the proportion of the value v in the cluster log c_i and that of v in the original log set is greater than threshold T_p. Similarly, $< a, !v >$ is regarded as the effective negative attribute–value pair, if and only if the difference of the two proportions is less than a given threshold T_n. Then, $< a, v >$ (or $< a, !v >$) is appended to the attribute-condition set of ρ, which is denoted as EAC^{ρ}. The set of all such rules is represented as P.

According to the above definition, the process of extracting effective attribute conditions for a given cluster ci is presented in Algorithm 1.

3.3. **Policy Optimization.** Compared to the original rules, rules extracted from the original access logs may be too restricted or too relaxed. Specifically, the extracted rule is considered to be restricted if it contains more complex attribute conditions than the original rule; otherwise, it is considered to be relaxed if the rule only contains some simple attribute conditions. Based on the logs AL^+ and AL^- as well as the initial extracted policy set $\Pi = \{\pi_1, \pi_2, \ldots\}$, the extracted log records with respect to different policy categories Π_{TP}, Π_{FP}, Π_{TN} and Π_{FN}, which are respectively represented as $TP_{\Pi|AL}$, $FP_{\Pi|AL}$, $TN_{\Pi|AL}$, and $FN_{\Pi|AL}$, are defined first.

Definition 2. $TP_{\Pi|AL} : TP_{\Pi|AL} = \{< rq, d > |\exists < rq, d >\in AL^+ : d_{\Pi}(rq) = permitted\}$, each record of which indicates that the authorization decision made by Π is still permitted for the access request rq of the positive logs AL^+;

ALGORITHM 1: Extraction of effective attribute conditions.

Algorithm 1. Extraction of effective attribute conditions

Input: Cluster c_i, access logs AL, attribute set A, set V of attribute–value pairs, and thresholds T_p and T_n

Output: ρ_{c_i}

1. Initialize $EAC^{\rho} = \emptyset$;
2. Identify and represent the set of all the entities included in c_i as E_{c_i};
3. Identify and represent the set of all the entities included in AL as E_{AL};
4. **for** each a in A **do**
5. **for** each v in V_a **do**
6. **if** $(\frac{\{e_k|\exists e_k \in E_{c_i}: v \in f_{a\_e}(e_k,a)\}}{|E_{c_i}|} - \frac{\{e_{k'}|\exists e_{k'} \in E_{AL}: v \in f_{a\_e}(e_{k'},a)\}}{|E_{AL}|}) > T_p$ **then**
7. $EAC^{\rho} \leftarrow \{< a, v >\}$;
8. **end if**
9. **if** $(\frac{\{e_{k'}|\exists e_{k'} \in E_{AL}: v \in f_{a\_e}(e_{k'},a)\}}{|E_{AL}|} - \frac{\{e_k|\exists e_k \in E_{c_i}: v \in f_{a\_e}(e_k,a)\}}{|E_{c_i}|}) > T_n$ **then**
10. $EAC^{\rho} \leftarrow \{< a, !v >\}$;
11. **end if**
12. **end for**
13. **end for**
14. $\rho_{c_i} \leftarrow EAC^{\rho}$;

Definition 3. $FP_{\Pi|AL}$: $FP_{\Pi|AL} = \{< rq, d > |\exists < rq, d > \in AL^+ : d_{\Pi}(rq) = \textit{permitted}\}$, each record of which indicates that the authorization decision made by Π becomes permitted for the access request rq of the negative logsAL^-;

Definition 4. $TN_{\Pi|AL} : TN_{\Pi|AL} = \{< rq, d > |\exists < rq, d > \in AL^+ : d_{\Pi}(rq) = \textit{denied}\}$, each record of which indicates that the authorization decision made by Π is still denied for the access request rq of the negative logs AL^-;

Definition 5. $FN_{\Pi|AL} : FN_{\Pi|AL} = \{< rq, d > |\exists < rq, d > \in AL^+ : d_{\Pi}(rq) = \textit{denied}\}$, each record of which indicates that the authorization decision made by Π becomes denied for the access request rq of the positive logs AL^+.

According to the definitions, we find that the restricted rules may produce large amounts of the $FN_{\Pi|AL}$ records, while the relaxed rules may produce large amounts of the $FP_{\Pi|AL}$ records. To further improve the correctness of the ABAC policy, the extracted rules are repeatedly corrected in terms of the original logs as well as the initial extracted policy, and the optimization process is presented as follows:

Step 1. The log records $FP_{\Pi|AL}$ and $FN_{\Pi|AL}$ are used as the training data sets to extract the policy patterns Π_{FN}, and Π_{FP}, respectively.

Step 2. Π_{FN} and Π_{FP} are respectively compared with Π, in order to identify the redundant or missing attribute conditions. Specifically, rules similar to those in Π_{FN} or in Π_{FP} are selected from Π during the optimization process in the following two ways:

(1) For any rule ρ_i of Π_{FN}, if there exists a rule ρ_j of Π that is similar to ρ_i, then the redundant attribute conditions are removed from ρ_j; if there exist no such rules, ρ_i is regarded as a missing rule and then is added into Π.

(2) For any rule ρ_i of Π_{FP}, if there exists a rule ρ_j of Π that is similar to ρ_i, then the missing attribute conditions are added into ρ_j. The detailed process is presented as shown in Algorithm 2.

4. **Experimental Analysis.** In this section, we implement experiments using the synthetic and real-world datasets, in order to demonstrate the efficiency and effectiveness of

ALGORITHM 2: Policy optimization

Algorithm 2. Policy optimization

Input: Initial policy set Π, policy patterns Π_{FN}, Π_{FP}, and effective attributes EAC
Output: Optimized policy set Π'

1. Initialize $\Pi' = \Pi$;
2. **for** each ρ_i in $\Pi_{FN}.P$ **do**
3. **for** each ρ_j in $\Pi'.P$ **do**
4. $sim(\rho_i, \rho_j) = \frac{\sum_{EAC_e \in \{EAC_U, EAC_O, EAC_S\}} |EAC_e{}^{\rho_i} \cap EAC_e{}^{\rho_j}|}{\sum_{EA_e \in \{EAC_U, EAC_O, EAC_S\}} |EAC_e{}^{\rho_i} \cup EAC_e{}^{\rho_j}|}$;
5. **if** ρ_j is similar to ρ_i **then**
6. $\rho_j \leftarrow EAC^{\rho_j} \backslash (EAC^{\rho_j} \backslash EAC^{\rho_i})$;
7. **else**
8. $\rho_j \leftarrow EAC^{\rho_j} \cup EAC^{\rho_i}$;
9. **end if**
10. **end for**
11. **end for**
12. **for** each ρ_k in $\Pi_{FP}.P$ **do**
13. **for** each ρ_l in $\Pi'_m.P$ **do**
14. $sim(\rho_k, \rho_l) = \frac{\sum_{EAC_e \in \{EAC_U, EAC_O, EAC_S\}} |EAC_e{}^{\rho_k} \cap EAC_e{}^{\rho_l}|}{\sum_{EA_e \in \{EAC_U, EAC_O, EAC_S\}} |EAC_e{}^{\rho_k} \cup EAC_e{}^{\rho_l}|}$;
15. **if** ρ_l is similar to ρ_k **then**
16. $\rho_l \leftarrow EAC^{\rho_l} \cup (EAC^{\rho_k} \backslash EAC^{\rho_l})$;
17. **end if**
18. **end for**
19. **end for**

the proposal. All the experiments are implemented on a standard desktop PC with an Intel i5–7400 CPU, 4 GB RAM, and a 160 GB hard disk running a 128–bit Windows 10 operating system. All simulations are compiled and run under the Python environment.

4.1. **Efficiency Evaluation of the Proposal.** First, three real-world datasets are considered: The University, Healthcare, and Project Management from the work [9], which have been usually used for evaluating the performances of different policy engineering methods recently. To evaluate the performance of the proposal in the preprocessing and policy extraction phases, the number of the policy rules and the execution time are considered as two main measures.

We repeatedly carry out experiments 10 times for each dataset and take the average value of the experimental result as output. Specifically, number of rules is 10, 7, and 12, respectively; execution time is 0.02 s, 0.02 s, and 0.03 s, respectively. We compare its performance with other methods such as the Xu-Stoller [9] and VisMAP [13] and find that the number of the partitioned ABAC rules of the proposal is less than or equal to the other two methods; meanwhile, execution time of these three methods is almost the same. This is because the size of the users, objects, and access logs in each dataset is small. Thus, our method performs as well as the Xu-Stoller and VisMAP methods using the small-scale datasets.

We next generate synthetic datasets using the particular parameters, since we find that there exist no suitable real-world datasets of large scales for the experiments. Specifically, the number of users takes 10 different value among 100 and 1000, the number of objects

takes 200, 400, 600 and 800, and the attributes of the entities are randomly selected from the three small-scale datasets above. To further evaluate the performance of the proposal, we consider the number of the rule clusters and the running time as measures and compare its performance with that of the VisMAP. Additionally, as described in the preprocessing phase, we partition the log records by clustering the original access logs, while the VisMAP directly separates an original matrix before rearrangement. If the constructed datasets are already sufficiently visual, then the access logs never need to be partitioned. Therefore, we first consider the following cases with no partitions.

We take the number of entities including the users and objects as input, repeatedly carry out the experiments using different datasets and take the average value of the rule number as well as that of the execution time. The results are presented as shown in Figures 2, and in Figure 3, respectively.

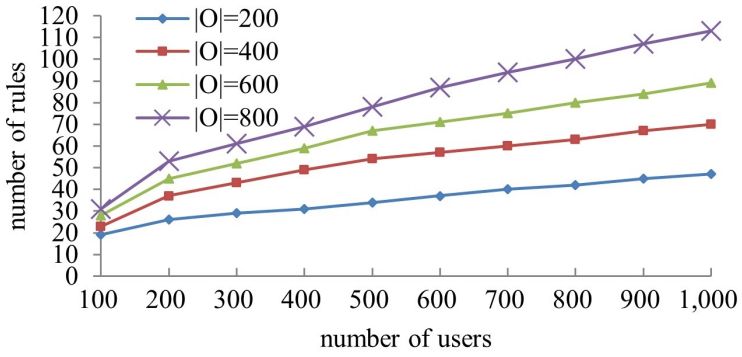

Figure 2. Evaluation of the number of rules

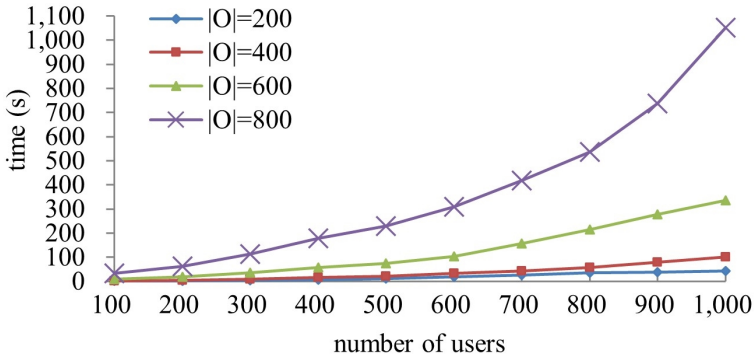

Figure 3. Evaluation of the execution time

Figure 2 shows the varying trend of the number of extracted rules with the increasing number of users as well as that of objects. It is observed that the rule number changes from 20 to 46 with 200 objects when the number of users changes from 100 to 1000, which tends to grow slightly. It is observed that, however, the rule number increases from 33 to 122 when the number of objects is 800, which varies obviously. On the other hand, from the viewpoint of the varying objects, it is observed that the rule number also increases when the number of users is fixed. In general, the number of the extracted rules varies slightly with the small-scale entities. That is attributed to the fact that the less the user number, the fewer rules discovered in the policy extraction.

Figure 3 shows the varying trend of execution time with the increasing number of users as well as that of objects. It is observed that execution time tends to grow linearly and is

always below 340 s when the number of objects is less than 600. However, the execution time arrives to around 1100 s with 1000 users and 800 objects, which is inefficient. This is because the scale of the access logs increases with the increasing number of users and that of objects. Thus, it needs more time to determine whether or not the partition is implemented before actually extracting the ABAC rules.

To demonstrate the efficiency of the policy extraction according to Algorithm 1, the experimental parameters are considered for generating synthetic datasets. These parameters contain numbers of users, objects, effective attribute–value pairs, rules used to construct the datasets and the maximum length of any rule, which are denoted as $|U|, |O|, |EAV|, |RC|$, and $|RL|$, respectively. Meanwhile, to compare the performance of the proposal with the Xu-Stoller, the generated datasets are converted into the data format that can be performed by the Xu-Stoller, and we assume that each access-control record contains only one permission. To study how these parameters would affect the extracted results, values of $|U|$ and $|O|$ are fixed as 800, and 200, respectively. The other parameter settings are considered as shown in the middle three columns of Table 1. The experimental results using different combinations of parameters as well as those of the Xu-Stoller are presented in the last two columns in Table 1. It is observed that the proposal performs as well as the Xu-Stoller for extracting policy rules.

Table 1. Comparison of the extracted rules.

$\|S\|$	$\|O\|$	$\|EAV\|$	$\|RC_{max}\|$	$\|RL_{max}\|$	$\|P_{Xu-Stoller}\|$	$\|P_{theproposal}\|$
800	200	40	30	5	26	25.53
800	200	45	30	5	24.62	24.61
800	200	50	20	5	16.65	16.51
800	200	50	40	5	34	34.28
800	200	50	50	5	41	40.75
800	200	50	30	4	24.63	23.33
800	200	50	30	3	23.66	23.49
800	200	50	30	2	22	22
800	200	60	30	5	25.64	24.29
800	200	70	30	5	25.61	24.47
800	200	80	30	5	26.58	25.76

4.2. **Effectiveness Evaluation of the Proposal.** Next, the synthetic access logs are randomly generated using a particular policy set such as the datasets UniversityP, HealthcareP, ProjectManagementP, UniversityPN, HealthcarePN, and ProjectManagementPN from the work [24]. The authorization rules of the policy are constructed with arbitrary attributes and their attribute values, which can evaluate the performance of the policies extracted from the access logs of different sizes and changing structural features. To construct the synthetic input data, the authorization tuples are generated in order to evaluate the ABAC policy. Real-world datasets are the open access logs that are provided by the datasets Amazon Kaggle and Amazon UCI. Amazon Kaggle records the access requests of the employees as well as the authorization results for the resource access, and it also specifies the attribute-property values and the resource identifiers of the employees. Further, Amazon Kaggle contains12000 users and 7000 object resources, while Amazon UCI contains more than 36000 users, 27000 permissions and 33000 attribute properties.

The Precision, Recall, Accuracy, and F1 score are considered as criteria to evaluate how well the extracted policies match with the original ones in our work. According to

Definitions 2-5, they can be represented as:

$$Precision_{\Pi|AL} = \frac{|TP_{\Pi|AL}|}{|TP_{\Pi|AL}| + |FP_{\Pi|AL}|} \tag{7}$$

$$Recall_{\Pi|AL} = \frac{|TP_{\Pi|AL}|}{|TP_{\Pi|AL}| + |FN_{\Pi|AL}|} \tag{8}$$

$$Accuracy_{\Pi|AL} = \frac{\left|TP_{\Pi|AL}\right| + \left|TN_{\Pi|AL}\right|}{\left|TP_{\Pi|AL}\right| + \left|TN_{\Pi|AL}\right| + \left|FP_{\Pi|AL}\right| + \left|FN_{\Pi|AL}\right|} \tag{9}$$

$$F_1 \quad_{\Pi|AL} = 2 \times \frac{Precision_{\Pi|AL} \times Recall_{\Pi|AL}}{Precision_{\Pi|AL} + Recall_{\Pi|AL}} \tag{10}$$

The Accuracy measure may have errors over the unbalanced datasets, while the F_1 score can be chosen as the desired criterion for evaluating the policy correctness. This is because the higher F_1 score, the higher the policy quality. This is means that the relationship between the extracted policies and the original access policies is closer.

The weighted structural complexity (WSC) is another important criterion to evaluate the policy quality, which aims to perform a generalized evaluation for the scale of a given ABAC policy. It is represented as:

$$WSC(\Pi) = WSC(P) = \sum_{\rho \in P} WSC(\rho) \tag{11}$$

$$WSC(\rho) = WSC(EAC, op) = w_1 \times WSC(EAC_u) + w_2 \times WSC(EAC_o) + w_3 \times WSC(EAC_s) \tag{12}$$

where $WSC(EAC_e) = |EAC_e|, |EAC_e|$ represents the number of attribute–value pairs included in the effective attribute conditions of entity e, w_i represents a specific weight that is used to adjust its contribution to the complexity of the rule, and $w_1 + w_2 + w_3 = 1$. Obviously, the less the value of WSC, the more convenient and concise the policy management.

To evaluate the performance of the policy optimization according to Algorithm 2, we repeatedly conduct the experiments using different datasets, including the synthetic and the real-world. In light of different evaluation criteria such as the running time, accuracy, F1 score and complexity, the best results of their performances are taken. The results of the proposal as well as those of methods such as the Xu-Stoller [9] and Cotrini [23] are presented as shown in Table 2, where " \ " indicates that the experimental results are unknown or unsatisfactory.

From the results of the table, it is observed that the proposal outperforms the other two methods, especially on the UniversityP, ProjectManagementP, Amazon Kaggle, Amazon UCI datasets. Specifically, for the UniversityP, running time, accuracy, F_1 score, WSC are 9.6 s, 97%, 76.1%, and 32, respectively; for the ProjectManagementP, the results of the proposal are 13.62, 90.33%, 53.01%, and 63, respectively; for the Amazon Kaggle, the results of the proposal are 205.1, 97.6%, 96.67%, and 75, respectively; for the Amazon UCI, the results of the proposal are 1231.71, 95.31%, 94.19%, and 95, respectively.

Furthermore, Figure 4 and Figure 5 present the comparisons of the three methods for F_1 score, and complexity, respectively. It is observed from Figure 4 that the changing trend of F_1 score of the proposal is similar to that of the Cotrini method, and both the methods outperform the Xu-Stoller method; it is observed from Figure 5 that the changing trend of the complexity of the proposal is flat and is very close to that of the Xu-Stoller method, and both the methods can extract more concise and higher-quality policies than the Cotrini method.

Table 2. Performance comparison with other methods.

Method	Policy set	Time (s)	Accuracy (%)	F1 score (%)	Complexity (WSC)
Xu-Stoller [9]	UniversityP	227	94.74%	65.87%	34
Cotrini [23]		126	80.74	45.3%	508
The proposal		**9.6**	**97%**	**76.1%**	**32**
Xu-Stoller [9]	HealthcareP	32645	64.43%	63.61	16
Cotrini [23]		529	72.72%	64%	272
The proposal		10.1	**81.15%**	**70.71%**	59
Xu-Stoller [9]	ProjectManagementP	\	3.54%	0.71%	29
Cotrini [23]		3587	91.57%	54.12%	77
The proposal		**13.62**	**90.33%**	53.01%	**63**
Xu-Stoller [9]	UniversityPN	4230	73.37%	16.1%	34
Cotrini [23]		204	93.55%	88.5%	1389
The proposal		**22**	92.11%	84.6%	47
Xu-Stoller [9]	HealthcarePN	45348	79.25%	73.09%	17
Cotrini [23]		3587	86.46%	79.2%	462
The proposal		**10.2**	**89.2%**	78.7%	84
Xu-Stoller [9]	ProjectManagementPN	\	2.19%	0.62%	45
Cotrini [23]		2848	82.75%	62.66%	100
The proposal		**25.83**	81.91%	54.47%	**61**
Xu-Stoller [9]	Amazon Kaggle	\	1.69%	0.7%	51
Cotrini [23]		237	84.25%	91.39%	2431
The proposal		**205.1**	**97.6%**	**96.67%**	**75**
Xu-Stoller [9]	Amazon UCI	\	1.55%	0.67%	43
Cotrini [23]		1345	70.93	75.64%	1247
The proposal		**1231.71**	**95.31%**	**94.19%**	**95**

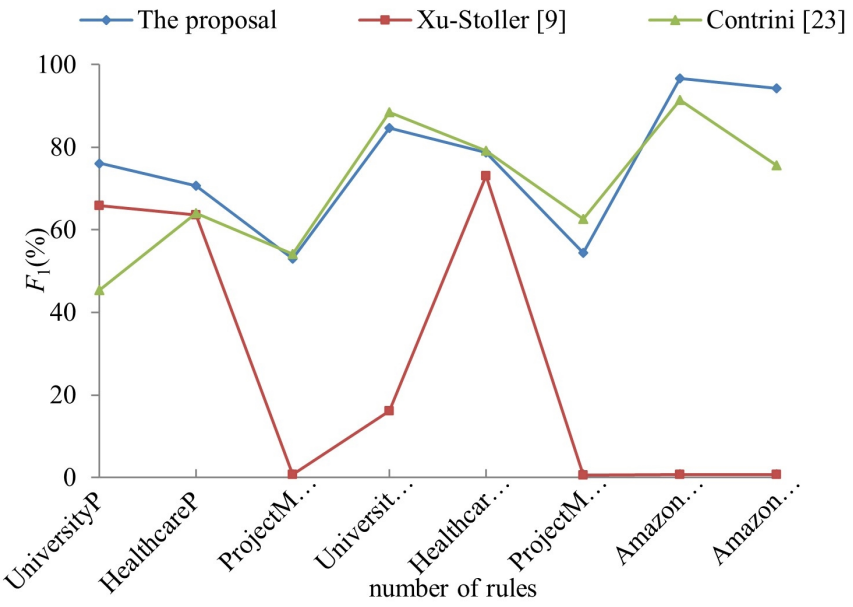

Figure 4. Comparison of the F_1 score

4.3. **Discussion.** From the above performance evaluation and comparison for the policy extraction and optimization, we find the proposal has the following main advantages:

(1) The authorization rules for ABAC with the negative attribute conditions may be more flexible and convenient. However, the negative conditions are not considered during the rule-extraction process using the existing methods. To address this issue, the attribute

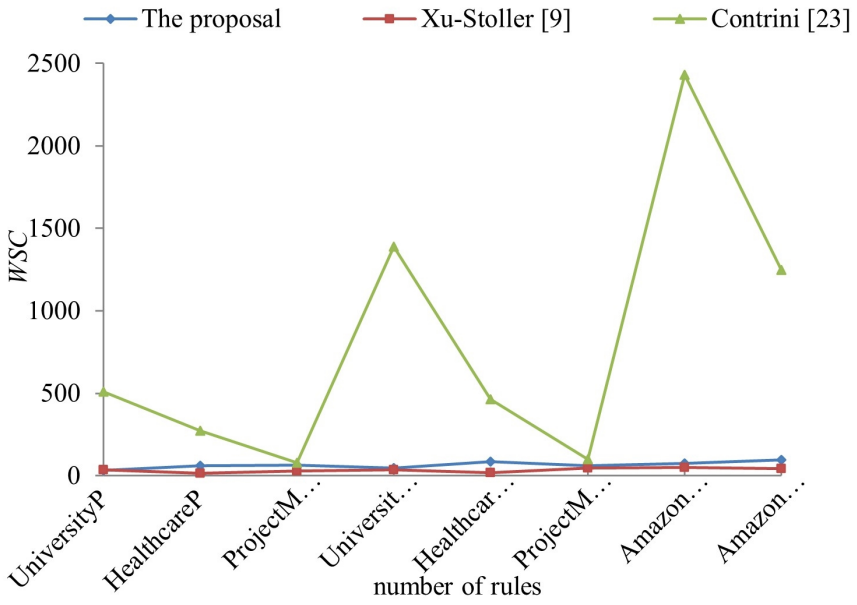

Figure 5. Comparison of the complexity

conditions including both the positive and negative types are effectively extracted to construct ABAC rules in our method, so that the policy descriptions for access requests become more flexible and convenient, while enhancing the interpretability of the ABAC policy.

(2) The ABAC policy should be as concise and correct as possible. However, there are numerous redundant and inaccurate rules extracted using the existing methods, and these initial derived policies should be optimized. To address this issue, based on principles of the correctness and conciseness, the evaluation criteria of the policy quality are presented in our method, and the efficiency and effectiveness of the proposal are demonstrated through the experiments.

(3) For a given set of access logs that contain access requests and system authorization decisions, the clustering partition technology is used to determine the initial number of policy rules in our method, which reduces the scale of the policy engineering.

Compared to the existing research approaches, features of the proposal are presented as shown in Table 3, where a tick ✓ indicates that the feature is available. Nevertheless,

Table 3. Comparison of features.

Feature	Zheng et al. [14]	Sun et al. [15]	Das et al. [13]	Xu et al. [9]	Cotrini et al. [23]	Proposed method
Enhancing the policy interpretability	✓	✓	✓			✓
Reducing the scale of the policy engineering		✓	✓			✓
Accuracy analysis of the policy				✓	✓	✓
Complexity analysis of the policy				✓	✓	✓

the security issues of the proposal such as the separation of duties constraint, various cardinality constraints on entities and attributes, as well as the conflict-detection problem,

however, are not considered during either the policy extraction or optimization phase, which are the main limitations of our work.

5. **Conclusion.** A novel policy extraction method for attribute-based access control based on access logs was proposed in this study. First, we utilized the partition technology to construct an initial set of rule clusters from the given access logs, and then we presented an algorithm to extract the positive and negative attribute conditions from the ini-tial cluster set and constructed effective rules. Next, we proposed a policy optimization algorithm to repeatedly refine the extracted rules and improve the policy quality according to different policy categories. As a result, the proposed method flexibly suited various requirements of organizations with more powerful policy expressiveness, improved the policy accuracy and reduced the complexity of the policy engineering. The experimental analysis showed that it could enhance the policy interpretability while guaranteeing the policy quality. Our future work will focus on studying how to implement the proposal in practical scenarios with the IoT, blockchain, and online social networks.

Acknowledgements. This work was partially supported by the Natural Science Foundation of China (61501393), the Foundation of Henan Educational Committee, under Contract No. 20B520031, and the Foundation of Guangxi Key Laboratory of Trusted Software (No. KX202061)

REFERENCES

[1] T.-Y. Wu, Q. Meng, Y.-C. Chen, S. Kumari, and C.-M. Chen, "Toward a secure smart-home IoT access control scheme based on home registration approach," *Mathematics*, vol. 11, no. 9, 2123, 2023.

[2] T.-Y. Wu, F. Kong, Q. Meng, S. Kumari, and C.-M. Chen, "Rotating behind security: An enhanced authentication protocol for IoT-enabled devices in distributed cloud computing architecture," *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, 36, 2023.

[3] T.-Y. Wu, L. Wang, X. Guo, Y.-C. Chen, and S.-C. Chu, "SAKAP: SGX-based authentication key agreement protocol in IoT-enabled cloud computing," *Sustainability*, vol. 14, no. 17, 11054, 2022.

[4] G. Batra, V. Atluri, J. Vaidya, and S. Sural, "Deploying ABAC policies using RBAC systems," *Journal of Computer Security*, vol. 27, no. 4, pp. 483–506, 2019.

[5] M. Narouei, H. Khanpour, H. Takabi, N. Parde, and R. D. Nielsen, "Towards a top-down policy engineering framework for attribute-based access control," in *22nd ACM on Symposium on Access Control Models and Technologies*, pp. 103–114, Indianapolis, IN, USA, 21–23 June, 2017.

[6] N. V. Verde, J. Vaidya, V. Atluri, and A. Colantonio, "Role engineering: From theory to practice," in *Second ACM Conference on Data and Application Security and Privacy*, San Antonio, TX, USA, pp. 181–192, 7–9 February, 2012.

[7] M. Alohaly, H. Takabi, and E. Blanco, "Towards an automated extraction of ABAC constraints from natural language policies," in *34th IFIP TC 11 International Conference on ICT Systems Security and Privacy Protection*, Lisbon, Portugal, pp. 105–11, 925–27 June 2019.

[8] A. Roy, S. Sural, A. K. Majumdar, J. Vaidya, and V. Atluri, "Enabling workforce optimization in constrained attribute-based access control systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1901–1913, 2019.

[9] Z. Xu and S. D. Stoller, "Mining attribute-based access control policies," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp.533–545, 2015.

[10] S. Das, B. Mitra, V. Atluri, J. Vaidya, and S. Sural, "Policy engineering in RBAC and ABAC," in *From Database to Cyber Security; Springer: Cham, Switzerland*, pp. 24–54, 2018.

[11] Y. Benkaouz, M. Erradi, and B. Freisleben, "Work in progress: K-nearest neighbors techniques for ABAC policies clustering," in *2016 ACM International Workshop on Attribute Based Access Control*, New Orleans, LA, USA, pp. 72–75, 11 March 2016.

[12] S. Das, S. Sural, J. Vaidya, and V. Atluri, "Poster: Using gini impurity to mine attribute-based access control policies with environment attributes," in *23nd ACM on Symposium on Access Control Models and Technologies*, Indianapolis, IN, USA, pp. 213–215, 13–15 June 2018.

[13] S. Das, S. Sural, J. Vaidya, V. Atluri, and G. Rigoll, "VisMAP: Visual mining of attribute-based access control policies," in *15th International Conference on Information Systems Security*, Hyderabad, India, pp. 79–98, 16–20 December 2019.

[14] G. Zheng and Y. Xiao, "A research on conflicts detection in ABAC policy," in *7th International Conference on Computer Science and Network Technology*, Dalian, China, pp. 408–412, 19–20 October 2019.

[15] W. Sun, H. Su, and H. Xie, "Policy-engineering optimization with visual representation and separation-of-duty constraints in attribute-based access control," *Future Internet*, vol. 12, no. 10, p. 164, 2020.

[16] S. Das, S. Sural, J. Vaidya, and V. Atluri, "HyPE: A hybrid approach toward policy engineering in attribute-based access control," *IEEE Letters of the Computer Society*, vol. 1, no. 2, pp. 25–29, 2018.

[17] P. Iyer and A. Masoumzadeh, "Mining positive and negative attribute-based access control policy rules," in *23nd ACM on Symposium on Access Control Models and Technologies*, Indianapolis, IN, USA, pp. 161–172, 13–15 June 2018.

[18] J. C. John, S. Sural, and A. Gupta, "Attribute-based access control management for multicloud collaboration," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 19, p. e4199, 2017.

[19] T. Talukdar, G. Batra, J. Vaidya, V. Atluri, and S. Sural, "Efficient bottom-up mining of attribute-based access control policies," in *3rd IEEE International Conference on Collaboration and Internet Computing*, San Jose, CA, USA, pp. 339–348, 15–17 October 2017.

[20] J. C. John, S. Sural, and A. Gupta, "Authorization management in multi-cloud collaboration using attribute-based access control," in *15th International Symposium on Parallel and Distributed Computing*, Fuzhou, China, pp. 190–195, 8–10 July 2016.

[21] J. C. John, S. Sural, and A. Gupta, "Optimal rule mining for dynamic authorization management in collaborating clouds using attribute-based access control," in *10th IEEE International Conference on Cloud Computing*, Honolulu, HI, USA, pp. 739–742, 25–30 June 2017.

[22] M. Gautam, S. Jha, S. Sural, J. Vaidya, and V. Atluri, "Poster: Constrained policy mining in attribute-based access control," in *22nd ACM on Symposium on Access Control Models and Technologies*, Indianapolis, IN, USA, pp. 121–123, 21–23 June 2017.

[23] C. Cotrini, T. Weghorn, and D. Basin, "Mining ABAC rules from sparse logs," in *2018 IEEE European Symposium on Security and Privacy*, IEEE, pp. 31–46, 2018.

[24] L. Karimi, M. Aldairi, J. Joshi, and M. Abdelhakim, "An automatic attribute-based access control policy extraction from access logs," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2304-2317, 2021.

[25] W. Sun, H. Su, and H. Liu, "Role-engineering optimization with cardinality constraints and user-oriented mutually exclusive constraints," *Information*, vol. 10, no. 11, p.342, 2019.