Computing and Informatics, Vol. 42, 2023, 615–650, doi: 10.31577/cai_2023_3_615

ATTRIBUTE-BASED ACCESS CONTROL POLICY GENERATION APPROACH FROM ACCESS LOGS BASED ON THE CATBOOST

Shan Quan, Yongdan Zhao, Nurmamat Helil*

College of Mathematics and System Science
Xinjiang University
China
e-mail: shanquan_owen@163.com, zydky2021@126.com, nur924@sina.com

Abstract. Attribute-based access control (ABAC) has higher flexibility and better scalability than traditional access control and can be used for fine-grained access control of large-scale information systems. Although ABAC can depict a dynamic, complex access control policy, it is costly, tedious, and error-prone to manually define. Therefore, it is worth studying how to construct an ABAC policy efficiently and accurately. This paper proposes an ABAC policy generation approach based on the CatBoost algorithm to automatically learn policies from historical access logs. First, we perform a weighted reconstruction of the attributes for the policy to be mined. Second, we provide an ABAC rule extraction algorithm, rule pruning algorithm, and rule optimization algorithm, among which the rule pruning and rule optimization algorithms are used to improve the accuracy of the generated policies. In addition, we present a new policy quality indicator to measure the accuracy and simplicity of the generated policies. Finally, the results of an experiment conducted to validate the approach verify its feasibility and effectiveness.

Keywords: ABAC policy, access logs, policy mining, ensemble learning, CatBoost

1 INTRODUCTION

In the big data era, big data platforms can help the information systems of organizations and enterprises overcome data isolation; support the integration of multi-

* Corresponding author

source heterogeneous data; and support cross-industry, cross-department, and cross-platform data sharing and exchange. As a result, data are now among the most strategic assets of any government, organization, or enterprise. Dengguo et al. [1] defined big data as the process of obtaining useful knowledge and predicting future trends, analyzing and grasping data's essential characteristics, and using the results of the analysis to distinguish the true from the false. Undoubtedly, big data has enduring value. However, it comes with the problem of data security. Ensuring that unauthorized entities do not access data is a security problem that must be solved in the process of data use. Access control is an essential solution to this problem.

In the big data environment, the access control system has many subjects, objects, and dynamic changes. Data structures and sources are complex and diverse. User types, demands for information sharing, and privacy needs are great. Moreover, access permissions are constantly changing [2]. Early access control models such as discretionary access control (DAC) [3, 4] and mandatory access control (MAC) [5] are not very suitable for addressing access control policies in the big data environment. The role-based access control (RBAC) model [6] maps users to roles through which they possess permissions. As the basis of the modern access control model, RBAC has been one of the popular research areas in access control, but RBAC relies heavily on user identity. The attribute-based access control (ABAC) model [7] later emerged as a fine-grained access control mechanism that relies on attributes. ABAC solves the problems of expressing and enforcing fine-grained access control and large-scale user dynamic expansion in a complex information system. Moreover, ABAC embeds entity attributes into the access control policy (ACP). As the subject, object, environment, and operation attributes have the ability to describe the access control and constraints in ABAC, the model has sufficient flexibility and extensibility.

With the rapid development of cloud computing, big data, artificial intelligence, and other technologies, the number of entities in information systems has exploded. ABAC uses subject and object attributes as essential criteria for permission access. The introduction of the environment attributes enables ABAC to support dynamic access control [8, 9, 10]. Unlike RBAC, ABAC does not need to design complex roles in advance, thus effectively avoiding the role explosion in RBAC [7]. However, when the number of subjects and objects and the number of subject and object attributes become large, it is challenging to specify ABAC policy manually. This is time-consuming and expensive, which makes ABAC's deployment in practical applications difficult [10]. Therefore, the research on ABAC policy mining is of great significance and can promote the development and popularization of the ABAC model.

As it is challenging to define ABAC policy manually, this paper proposes an approach to ABAC policy generation from access logs based on the CatBoost algorithm. This is an integrated learning method that can automatically learn ABAC policy from historical access logs. First, we reconstruct the attributes of the policies that need to be mined by weighting. Subsequently, we propose the rule extraction algorithm, rule pruning algorithm, and rule optimization algorithm to improve the

accuracy of the generated policy. In addition, we propose a new policy quality indicator, namely the policy quality comprehensive indicator, which measures the accuracy and conciseness of the generated policy.

The rest of this article is organized as follows. In Section 2, we review the research background and related work. In Section 3, we summarize the prior knowledge of ABAC and the CatBoost machine learning (ML) algorithm and detail the preparatory work of ABAC policy generation. In Section 4, we introduce the generation process of the ABAC policy and give the related implementation algorithm in detail. In Section 5, we provide the experimental results and evaluation. Finally, in Section 6, we present the conclusion and research prospects.

2 RELATED WORK

The ABAC model is widely used in large distributed environments, web service systems, grid computing, and information sharing and management [1]. In ABAC deployment, one of the critical challenges is how to infer ACP from the logs of past decisions (*permit* or *deny*) on the access requests made by users. In the ABAC access control system, two primary sources of information describe the relations between subjects and objects: the original access control system and the access logs [11]. The basic idea of policy mining is to combine subject, object, environment, and operation attribute data to mine ABAC policies from the relations between the subject and object. Therefore, mining ABAC policies from access logs has attracted the attention of researchers.

The initial research field of policy mining was RBAC role mining. Vaidya et al. [12] introduced an approach of role mining that finds the best role from the user-permission assignment relations by decomposing the user-permission Boolean matrix. Molloy et al. [13] proposed an RBAC role mining algorithm based on formal concepts. Molloy et al. [14] proposed a role mining approach that allows noisy data. Most role mining approaches assume that the data used are correct and noise-free, which is often not the case. Thus, this approach improves the quality of role mining. Currey et al. [15] proposed a multi-objective role mining approach that minimizes unnecessary permissions as the formal goal of role mining. Jafarian et al. [16] transformed the role mining problem into a constraint satisfaction problem. This approach effectively combines top-down and bottom-up patterns. The top-down pattern starts from the security requirements and then gradually refines the business and then dissolves into independent functional units to generate policies. The bottom-up pattern starts with access requests and uses the common ground among access requests to generate policies. Combining the two patterns makes the generated policies easier to understand and maintain and of higher quality.

Some scholars have proposed RBAC policy mining approaches based on ML. Molloy and Chari [17] proposed an RBAC role mining approach with permissions, which is based on ML algorithms. Their approach has advantages in generality, coverage, and stability. Narouei and Takabi [18, 19] proposed an approach of uti-

lizing natural language processing (NLP) technology called semantic role labeling (SRL), which extracts ACPs from unrestricted natural language documents, defines roles, and constructs an RBAC model. It is a top-down pattern for role mining. As this approach considers all predicates in the ACPs sentence, it leads to some false positives, which makes their approach's precision relatively low (precision of 75 %). Anderer et al. [20] created a library of role mining benchmark instances, which includes some new, synthetically generated benchmark instances of different sizes for evaluating and comparing role mining algorithms. The benchmark instances leave more space between the number of roles derived from the two common decompositions of the role mining problem (RMP) and the actual minimum number of roles, thus making them better, multifaceted, and able to thoroughly evaluate the role mining algorithm.

As ABAC is widely used in access control, researchers have also proposed ABAC policy mining approaches. Chari and Molloy [21] proposed mining ABAC rules automatically from access logs instead of manually making and maintaining the ABAC rule set. They used cross entropy to exclude user attributes to mine a rule set. Xu and Stoller proposed ABAC policy mining algorithms from access logs [11], RBAC [22], and the access control list (ACL) and attribute data [23]. Their algorithms iterate over access control tuples and build candidate rules, and then generalize them by replacing the conjunctions in the attribute expressions with constraints. Iyer and Masoumzadeh [24] proposed an approach to mine positive and negative ABAC policies. It can extract (*permit* or *deny*) the ACP at the same time. The mining policy is also relatively concise, thus making it superior to a previous approach [23]. Chakraborty et al. [25] defined the existence problem of the ABAC rule set and provided an algorithm to solve it. They further introduced the concept of the infeasible rule set modification in ABAC and the modification algorithm. Talukdar et al. [26] proposed an algorithm that finds the most general rule from a set of candidate rules, which can automatically build a reasonable ABAC policy. The main advantage of this approach is that the running time is stable and is not affected by the number of attributes. Narouei et al. [27] proposed an approach of ABAC policy mining based on the particle swarm optimization algorithm and ABAC policy mining under the minimal perturbation problem, and proposed a global optimization function to obtain the optimal ABAC state while making it as similar as possible to the existing state. Medvet et al. [28] proposed an evolutionary approach of multi-objective strategy mining based on genetic operators. It generates strategies through iterative, evolutionary search. Each iteration learns new rules and makes the set of access control tuples smaller to improve the quality of mining rules. Das et al. [29] proposed an ABAC policy mining algorithm based on the Gini coefficient impurity. The algorithm considers the environment attributes and their associated values and uses the approach based on the decision tree (DT) to build the policy. Although the generated rules are few and compact, access control decisions can be made faster.

Owing to the rapid development of big data, artificial intelligence and other related technologies (e.g., ML) have been widely used. As a result, some researchers

have proposed ABAC policy mining algorithms based on ML. Cotrini et al. [30] proposed an algorithm named Rhapsody to mine ABAC rules from sparse logs. They also defined the concept of reliability to measure the reliability of the extracted rules. The algorithm also considers whether the generated policies are overly permissive. Karimi and Joshi [31] proposed an approach that uses unsupervised learning to detect specific patterns in a set of access records and then extract ABAC policies from these patterns. In addition, they provided two algorithms, rule pruning and policy refinement, which are used to improve policy quality. Das et al. [32] provided a visual ABAC policy mining approach. It represents the existing access requests in the form of a binary matrix and then transforms the problem of finding the best representation of the binary matrix into a minimization problem by extracting rules from the visual access control matrix.

Some scholars have proposed ABAC policy mining approaches based on neural network (NN) and reinforcement learning (RL). Narouei et al. [33, 34] provided an information extraction approach from natural language documents via a recurrent neural network (RNN) and SRL. It can identify access control policy statements, and its performance was 5.58 % higher than that of the support vector machine model. Alohaly et al. [35, 36] proposed a convolutional neural network attribute extraction approach. Their approach *F1-score* performs well; it can generate a practical framework for analyzing natural language access control policies; and it can identify the attributes of the subjects and object elements. Karimi et al. [37] proposed an adaptive ABAC policy learning approach that can realize the automation of decisions. It is a kind of RL. This approach shows good performance in policy transfer using the learning feedback mechanism, and it is superior to the approach based on supervised learning.

Here, we focus on reviewing the approaches of [38] and [39]. The [38] and [39] use restricted Boltzmann machine (RBM) and multi-layer perceptron (MLP) to mine ABAC policies, respectively. Mocanu et al. [38] presented an ABAC policy mining approach based on deep learning that uses the RBM algorithm to train logs and extract rules. They first summarize knowledge from logs and generate a set of candidate rules in binary vector format and then convert the candidate rule set from the binary vector format to an acceptable format. Finally, the reconstruction error of all log entries in the obtained model is calculated, and the maximum value is taken as the threshold. Then, they generate all possible rule combinations, calculate the reconstruction error in the obtained model, and add the rules whose reconstruction error is less than the threshold to the candidate rules. The implicit distribution of data can be found through this approach. This approach has strong anti-noise ability, but it does not further optimize and analyze the rule set. Cappelletti et al. [39] deduced ABAC policy by comparing different symbolic and non-symbolic ML techniques. They used MLP to infer ABAC policies from access logs and turn them into a classification problem. MLP is a neural network model, which maps multiple input datasets to a single output dataset. It provides a deep feed-forward artificial network and generates a set of outputs from one set of inputs and the other end. Its feature is that several layers of input nodes are connected as a directed graph

between the input and output layers. MLP had a relatively good policy decision result in the experiment compared with other approaches.

ABAC is the most prominent access control model. Therefore, to improve the efficiency and accuracy of access request decisions, ABAC policy mining is a topic worth studying. As logs reflect the ACPs and user behaviors implemented in an organization, mining ACPs or rules from logs can help us reconstruct and simplify complex and dynamic policies. Researchers have successively proposed policy mining algorithms based on Rhapsody, unsupervised learning, neural networks, and RL. Rhapsody, unsupervised learning, NNs, RL, and other ML-based policy mining algorithms are more practical and effective, but they have some deficiencies. For example, in these algorithms, the decision is overly permissive, and only policy attributes are extracted. Although these algorithms improve decision efficiency in various ways, in practice, there are still some problems that require further study, such as no optimized rules and poor accuracy.

3 PRELIMINARIES FOR ABAC POLICY MINING

This paper uses the ML algorithm based on CatBoost to study how to mine more accurate and reasonable policies. The CatBoost algorithm is primarily used for classification, prediction, and regression. CatBoost has a wide range of application scenarios and can deal with gradient bias and prediction shift, which improves the accuracy and generalization ability of the algorithm. For the ABAC policy mining problem, we choose the CatBoost algorithm mainly because of its practicality, robustness, accuracy, and extensibility. This paper makes the following contributions:

1. We propose an ABAC policy generation method based on CatBoost, in which we learn ABAC policy from historical access logs.
2. We perform a weighted reconstruction of the attributes for the ABAC policy to be mined, which helps improve the accuracy and rationality of ABAC rule extraction.
3. We propose a rule extraction algorithm, rule pruning algorithm, and rule optimization algorithm to improve the accuracy of the generated policies.
4. We propose a new policy quality indicator, namely the policy quality comprehensive indicator, to measure the accuracy and simplicity of the generated policies.

In this section, we give a brief overview of ABAC, data mining (DM), the CatBoost ML algorithm, and some preparations for ABAC policy mining.

In this article, we try to follow the National Institute of Standards and Technology ABAC standard [7]. We use user attributes, object attributes, and session attributes to refer to access requester attributes, object attributes, and environment attributes (or conditions), respectively.

3.1 Learning CatBoost

The full name of CatBoost is Categorical Boosting. This algorithm is a machine learning algorithm proposed by the Russian search giant Yandex in 2017. It belongs to the boosting family of algorithms in integrated learning. It is an integrated algorithm combining gradient boosting and Oblivious Trees [40, 41, 42]. It is suitable for heterogeneous (different types) data and can handle gradient deviation and prediction deviation problems. Therefore, this algorithm can not only improve the quality and prediction speed of the classification model but also significantly improve the accuracy and generalization ability of the algorithm. The main advantages of the CatBoost algorithm are as follows [40, 41, 42]:

1. Practicability: It can process categorical and numerical data and it supports categorical variables without requiring preprocessing of the non-numerical features;
2. Robustness: High-quality models can be obtained without parameter adjustment, and very good results can be obtained by using the default parameters, thus reducing the time spent on parameter adjustment and the need for superparameter tuning;
3. Accuracy: It uses a new gradient lifting algorithm to build the model, thus reducing overfitting and improving the accuracy of the model;
4. Extensibility: It supports user-customized loss functions.

3.2 Preliminaries

Generally, when processing attributes in data, it is necessary to determine the attribute types beforehand, as the chosen processing methods differ against different types of attributes. Attributes are used to describe the properties or characteristics of an entity that vary from entity to entity and change over time (e.g., teachers' job titles, students' ages, and courses). In general, attributes can be divided into five categories: ordinal, nominal, interval, ratio [43], and binary. A binary attribute has only two states, denoted by `true` and `false` or 0 and 1, respectively.

Nominal, binary, and ordinal attributes are called categorical attributes, which are qualitative and discrete. Interval and ratio attributes are also called numeric attributes. A numeric attribute is quantitative, and its value can be discrete or continuous. The attributes' specific descriptions are shown in Table 1.

Let $En = U \cup O \cup E$ be the set of all entities and $A = A_u \cup A_o \cup A_e$ be the set of all attributes in the ABAC system. Here, U, O, and E are the ABAC system's sets of users (subjects), objects, and environments, respectively. Each element (entity) in U, O, and E is expressed by the Boolean combination of related attributes. In addition, OP is a set of operations in the system. A_u, A_o, and A_e are the set of user (subject) attributes, object attributes, and environment attributes, respectively.

Attribute Types		Attribute Description	Attribute Example	Attribute Analysis
Categorical attribute	Ordinal attribute	Attribute values have an order or size	Education, grade etc.	Median, percentile etc.
	Nominal attribute	Represents the category, code, or status	Native place, name, occupation, etc.	Mode, entropy, etc.
	Binary attribute	With only two categories or states	Flip a coin for positive or negative, nucleic acid test results for positive or negative, etc.	Contingency correlation etc.
Numerical attribute	Interval attribute	Comparing the difference is significant	Temperature, time, date, etc.	Mean, standard deviation, etc.
	Ratio attribute	Calculating the ratio or difference is necessary	Age, length, percentage of project completed, etc.	Geometric mean, harmonic mean, etc.

Table 1. Classification of attributes

Definition 1 (Attribute Domain)**.** Let the attribute $a \in A$. The set of all valid values of a is called the attribute domain of a, denoted as $V(a)$.

Definition 2 (Attribute Relation)**.** Define the binary relation $F = \{\langle a, v\rangle \mid a \in a, v \in V(a)\}$ as an attribute relation.

Definition 3 (Access Request)**.** The access request (ar) is a four-tuple $ar = (u, o, e, op)$, which is explained as follows: User $u \in U$ sends an access request to the system, requesting that it perform the operation $op \in OP$ on the object $o \in O$ under the environmental condition $e \in E$. u, e, o are determined by specific attribute relations.

Definition 4 (Access Control Decision)**.** An access control decision is a five-tuple $acd = (ar, d) = (u, o, e, op, d)$, composed of a user, an object, the environment, an operation, and the decision. Here, $d \in \{permit, deny\}$.

An access control decision result is either *permit* or *deny*. When the decision is *permit*, the user (requester) can perform the given operation on the given object under the given environment. When the decision is *deny*, the user cannot perform the given operation on the given object under the given environment.

Definition 5 (Access Logs)**.** The access logs (L) are a set of access control decisions.

As the decision result is *permit* or *deny*, we can divide the access logs (L) into positive access logs (L^+) and negative access logs (L^-). That is,

- $L^+ = \{(ar, d) \mid d = permit\}$,
- $L^- = \{(ar, d) \mid d = deny\}$.

Definition 6 (Access Rule)**.** An access rule refers to a multi-tuple $r = (F, op, d)$, where F is an attribute relation that includes relations regarding users, objects, and environments. It can be written as $F = F_u \cup F_o \cup F_e$. op is an operation, and d is a decision.

Example 1. A rule $r = (\{\langle Position, Student\rangle, \langle Location, Campus\rangle, \langle Type, Book\rangle, \langle Id_{library}, Id_{student}\rangle\}, borrow, permit)$ can be explained as "if a student is on campus and his/her student number matches the library code, he/she is permitted to borrow a book from the library."

Definition 7. Given the four-tuple $t = (u, o, e, op)$ from an access request $ar = (u, o, e, op)$ or an access control decision $acd = (u, o, e, op, d)$, and the rule $r = (F_r, op_r, d_r)$, if $F_u \cup F_o \cup F_e \subseteq F_r$, where F_u, F_o and F_e is the attribute relation of user u, object o, and environment e in the access request or the access control decision, $op = op_r$; then, we say the four-tuple satisfies rule r, denoted as $t \models r$. For simplicity, we say the access request ar (access control decision acd) satisfies rule r, denoted as $ar \models r$ ($acd \models r$).

In Definition 7, we mainly consider the satisfiability between the four-tuple (u, o, e, op), from an access request ar (an access control decision acd) and a rule.

Thus, regarding the rule set R in an ABAC system,

$$R \subseteq F_U \times F_O \times F_E \times OP \times D,$$

where F_U, F_O, and F_E are the set of attribute relations of all users in U, objects in O, and environments in E, respectively; $D = \{permit, deny\}$.

Definition 8. Permission $pe = (o, e, op)$ is defined as the operation of a user (subject) on an object under an environment, which is expressed by an object, the environment-related attribute relations, and an operation.

Definition 9 (ABAC Instance)**.** An ABAC instance is a subset of the multivariate relation $AR \times d$, denoted as I_{AR}, where AR represents the set of access requests (ar), and d is the set of decision results (*permit* or *deny*). I^+_{AR} and I^-_{AR} are defined as subsets of I_{AR}, where the decision in this instance is *permit* or *deny*, respectively.

Position \ Location	Campus		Home			
Professor	√	√	√	•	•	√
	√	√	√	•	√	•
Associate Professor	√	√	×	•	•	•
	√	√	•	•	•	×
Lecturer	√	√	•	•	×	•
	√	•	•	×	•	•
Student	•	•	•	•	×	×
	•	•	×	•	•	•

Table 2. An instance of the access log. Each tick √, cross ×, and circle • denotes an access request (i.e., a user). The ticks √ and crosses × denote logged requests that have been permitted and denied, respectively. The circles • denote users who have not requested permission yet.

Definition 10 (Rule Confidence)**.** Let $AR \times d$ be an instance of ABAC; then, the confidence of rule r is defined as follows:

$$Conf(r) = \frac{|I_{AR}^{+}|}{|I_{AR}|}, \tag{1}$$

where I_{AR} represents the set of all requests in the instance that satisfy rule r, and I_{AR}^{+} denotes the set of all permitted requests in the instance that satisfy rule r. If $|I_{AR}| = 0$; then, we define $Conf(r) = 0$.

Example 2. In Table 2, the confidence of rules $r_1 = (\{\langle Location, Campus\rangle\}, borrow, permit)$ is $\frac{11}{16} \approx 0.69$, and the confidence of rules $r_2 = (\{\langle Position, Professor\rangle\}, borrow, permit)$ is $\frac{8}{12} \approx 0.67$.

Definition 11 (Rule Refinement)**.** Given two rules $r = (F, op, d)$ and $r' = (F', op', d')$, if $F \subset F'$ ($F_u \subset F'_u \wedge F_o \subset F'_o \wedge F_e \subset F'_e$), $op = op'$, $d = d'$, then r' adds new constraints on r. r' is called refinement of r, the refinement relation is denoted as $r \propto r'$.

Definition 12. For the given refinement relation $r \propto r'$, we say rule r' is overly permissive if $Conf(r') < Conf(r)$.

Example 3. As shown in Table 2, consider two refinements of the rule $r_1 = (\{\langle Location, Campus\rangle\}, borrow, permit)$:

- $r_{11} = (\{\langle Location, Campus\rangle, \langle Position, Professor\rangle\}, borrow, permit)$;
- $r_{14} = (\{\langle Location, Campus\rangle, \langle Position, Student\rangle\}, borrow, permit)$.

These refinements have confidence 1.0 and 0, respectively.

As shown in Example 3, we can see that the rule $r_{14} = (\{\langle Location, Campus\rangle, \langle Position, Student\rangle\}, borrow, permit)$, and its confidence is decreased to 0; so, this rule is overly permissive.

Definition 13 (Rule Credibility)**.** Let $AR \times d$ be an instance of ABAC; the credibility of rule r is defined as

$$Cre_T(r) = \min_{r' \in F_T(r)} \{Conf(r), Conf(r')\}, \tag{2}$$

where $F_T(r) = \{r' \mid |[r']| \geqslant T\}$, $[r']$ is the set of refinements of r, and T is a parameter specified by a policy administrator. Generally, the optimal value of T is the minimum number of times the refinement r' satisfies an access request. If $F_T(r) = 0$, we define $Cre_T(r) = Conf(r)$.

Example 4. We compute the rule credibility for the rules $r_1 = (\{\langle Location, Campus\rangle\}, borrow, permit)$ and $r_2 = (\{\langle Position, Professor\rangle\}, borrow, permit)$ for the instance of Table 2.

Consider the below refinements of rule $r_1 = (\{\langle Location, Campus\rangle\}, borrow, permit)$:

- $r_{11} = (\{\langle Location, Campus\rangle, \langle Position, Professor\rangle\}, borrow, permit)$;
- $r_{12} = (\{\langle Location, Campus\rangle, \langle Position, AssociateProfessor\rangle\}, borrow, permit)$;
- $r_{13} = (\{\langle Location, Campus\rangle, \langle Position, Lecturer\rangle\}, borrow, permit)$;
- $r_{14} = (\{\langle Location, Campus\rangle, \langle Position, Student\rangle\}, borrow, permit)$.

$Cre_4(Conf(r_1)) = \min\{Conf(r_1), Conf(r_{11}), Conf(r_{12}), Conf(r_{13}), Conf(r_{14})\} = \min\{0.69, 1.0, 1.0, 0.75, 0.0\} = 0$.

Consider the below refinements of rule $r_2 = (\{\langle Position, Professor\rangle\}, borrow, permit)$:

- $r_{21} = (\{\langle Position, Professor\rangle, \langle Location, Campus\rangle\}, borrow, permit)$;
- $r_{22} = (\{\langle Position, Professor\rangle, \langle Location, Home\rangle\}, borrow, permit)$.

$Cre_2(Conf(r_2)) = \min\{Conf(r_2), Conf(r_{21}), Conf(r_{22})\} = \min\{0.67, 1.0, 0.5\} = 0.5$.

Theorem 1. Let $T \geqslant 1$, $K \in [0, 1]$; r is a rule, K is a specified value, and in general, $K \approx \frac{|AR^+|}{|AR|}$, AR^+ is the set of permitted access requests. If there is a refinement relation $r \propto r'$ that satisfies $Conf(r') < K$ and $|[r']| \geqslant T$, then r' is overly permissive.

Proof. From the Definitions 10, 12, and 13, we can see that if a rule r' ($r \propto r'$) is overly permissive, then its confidence must be less than the confidence of the original rule (that is, equal to K), so $Conf(r') < K$ and $|[r']| \geqslant T$. If the refinement r' satisfies $Conf(r') < K$ and $|[r']| \geqslant T$, then the refinement r' is overly permissive. □

Corollary 1. If $Conf(r') \geq K$, $|[r']| \geqslant T$, and r' is not overly permissive, we have credibility $Cre_T(r') \geq K$.

Proof. From the Definitions 10, 12, and 13, similarly, Corollary 1 is also true. □

Definition 14 (Log Credibility)**.** The log credibility is defined as

$$Cre_T(r) = \min_{r' \in F_T(r)} \{Conf(r), Conf(r')\}. \tag{3}$$

However, in the logs, the confidence cannot be directly computed in the instance; so, we denote the set of records in the access log set that meets rule r as $\{r\}_L^I$ $(I \subseteq L)$. In the logs, we have

$$F_T(r) = \{r' \mid |[r']| \geqslant T\}, \tag{4}$$

$$Conf(r) = \frac{|\{r\}_L^{L^+}|}{|\{r\}_L^L|}, \tag{5}$$

where L^+ indicates the positive (*permit*) access logs.

3.3 Policy Generation-Related Methods and Evaluation Metrics

In practice, it is difficult to see the correlation between the features and targets and the correlation between the features. Therefore, using mathematical or engineering methods is necessary to help improve feature selection. This paper uses an embedded method to select the features. For details, see Figure 1.

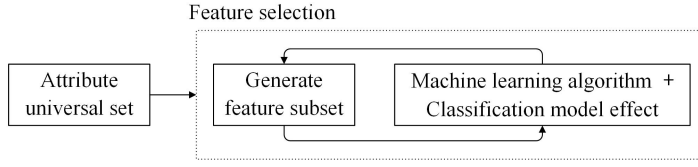

Figure 1. Embedded method for feature selection

The mutual information between the random variables X and Y is the mathematical expectation of mutual information between individual events, which is also used to evaluate the correlation between variables. The mutual information calculation formula is as follows:

$$I(X;Y) = E[I(x_i;y_i)] = \sum_{x_i \in X} \sum_{y_i \in Y} p(x_i, y_i) \log \frac{p(x_i, y_i)}{p(x_i)p(y_i)}, \tag{6}$$

$$I(X;Y) = H(X) - H(X \mid Y) = -\sum_{x_i \in X} p(x_i) \log p(x_i) - \left(-\sum_{x_i \in X, y_i \in Y} p(x_i, y_i) \log p(x_i \mid y_i) \right). \tag{7}$$

We can evaluate the CatBoost ML algorithm by k-fold cross-validation according to accuracy and other indicators. The evaluation is carried out on the cross-validation dataset. Based on this, the following definitions are given.

- *True Positive* (TP): If the access request is permitted based on the actual policy, it is also permitted based on the generated policy;
- *True Negative* (TN): If the access request is denied based on the actual policy, it is also denied based on the generated policy;
- *False Positive* (FP): If the access request is denied based on the actual policy, it is permitted based on the generated policy;
- *False Negative* (FN): If the access request is permitted based on the actual policy, it is denied based on the generated policy.

We can obtain *True Positive Rate* (TPR), *True Negative Rate* (TNR), *False Positive Rate* (FPR), *False Negative Rate* (FNR), *Accuracy* (Acc), *Sensitivity/Recall*, *Precision*, *F1-score* and *Matthews correlation coefficient* (Mcc), through the above definition. The calculation formulas are as follows:

$$TPR = Sensitivity = Recall = \frac{TP}{TP + FN}, \tag{8}$$

$$TNR = Specificity = \frac{TN}{TN + FP}, \tag{9}$$

$$FPR = \frac{FP}{FP + TN}, \tag{10}$$

$$FNR = \frac{FN}{FN + TP}, \tag{11}$$

$$Precision = \frac{TP}{TP + FP}, \tag{12}$$

$$Accuracy(Acc) = \frac{TP + TN}{TP + TN + FP + FN}, \tag{13}$$

$$F1\text{-}score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \times Precision \times Recall}{Precision + Recall}, \tag{14}$$

$$Mcc = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{15}$$

The increase in ACPs makes maintenance more difficult and increases the computational overhead. So, complexity and accuracy are essential evaluation indicators of policy quality. Therefore, weighted structural complexity (WSC) and accuracy are considered in this paper. In addition, the influence of other evaluation indicators on policy quality is considered.

WSC is used to summarize the size of the policy. Molloy et al. introduced *WSC* [13] into the artificial mining of RBAC policy. Later, Xu and Stoller extended it to mining ABAC policy [23]. The simpler the policy, the easier it is to manage in the system. *WSC* is the weighted sum of the number of elements of the ABAC policy with regard to π. Its calculation formula is as follows:

$$WSC_\pi = \sum_{r \in \pi} WSC(r), \tag{16}$$

$$WSC(r) = w_1 \cdot WSC(F_u) + w_2 \cdot WSC(F_o) + w_3 \cdot WSC(F_e) + w_4 \cdot WSC(OP). \tag{17}$$

For $\forall\ F_u,\ F_o,\ F_e,\ OP,\ WSC(F_u) = |F_u|,\ WSC(F_o) = |F_o|,\ WSC(F_e) = |F_e|,\ WSC(OP) = |OP|$, where $w_{i \in \{1,2,3,4\}}$ is the specified weight.

3.4 ABAC Policy Mining

In this section, we discuss ABAC policy mining (ABAC-PM) based on the characteristics of ABAC and the factors and challenges that need to be considered in mining ABAC policies.

Definition 15 (ABAC Policy)**.**[1] An ABAC policy (π) is a set of rules with the same permission. That is,

$$\pi = \left\{ \bigvee_{i}^{n} r_i \mid r_i \in R \wedge F_{o_{r_i}} = F_{o_{r_j}}, F_{e_{r_i}} = F_{e_{r_j}}, op_{r_i} = op_{r_j}, \right.$$
$$\left. d_{r_i} = d_{r_j}, \forall i, j \in \mathbb{N}^*, i \neq j \right\},$$

where $F_{o_{r_i}}$, $F_{o_{r_j}}$, $F_{e_{r_i}}$, $F_{e_{r_j}}$, op_{r_i}, op_{r_j}, d_{r_i}, d_{r_j} represent the subject, object, and environmental attributes relation, operation and decision of two different rules, respectively.

ABAC Policy Mining. If given a set of subjects (S), a set of objects (O), a set of environments (E), a set of operations (OP), and attribute domains and access logs or an ACL, we need to construct an ABAC policy set (Π). We have the following requirements for policy mining:

1. Every access request in the access logs or ACL satisfies at least one rule in policy $\pi \in \Pi$;
2. For any rule $r \in \pi$, it is as concise as possible;
3. The number of rules in π is as small as possible, and the accuracy is as high as possible.

[1] The basic unit of a policy in ABAC is a rule. We do not strictly differentiate between rule and policy in this article.

Suppose a given original access control system has the access control decisions of *permit* and *deny* against requests; it is related to numerous attributes; and the collected data is complex. In that case, it is ideal to use the CatBoost algorithm to mine the ACP and write it as a form of the ABAC policy. The ABAC policy extraction can be regarded as establishing a mapping between the access control decision data, including the user, object, environment attributes, and the set of ABAC policies. This mapping can be represented by the function $LF : L \to Y$, where

1. L represents a set of access control decisions (access logs): The components of each tuple in this work are restricted to categorical or nominal variables (e.g., the category value of an attribute);
2. Y represents a set of numbered labels (set labels), each corresponding to a rule in the ABAC policy π.

We aim to make the loss function (LF) (i.e., the function of classification error in machine learning) smaller and to mine the desired policy with high precision and efficiency.

4 ABAC POLICY GENERATION

Manually defining ABAC policies is expensive and time-consuming; so, an automated approach to mining ABAC policies helps simplify the adoption or migration of ABAC policies. Therefore, this section discusses the ABAC policy generation method based on CatBoost.

4.1 ABAC Policy Extraction

The ABAC policy extraction problem essentially involves finding rule r from a log dataset (L), making the ABAC policy (π) concise and more accurate when making decisions. We use CatBoost to extract the ABAC policy. The specific process of policy extraction is shown in Figure 2, including implementation steps and a summary.

4.2 Access Logs Preprocessing

After collecting the access logs, we first preprocess them to map the attribute values of the subject, object, and environment attributes to a type value. Some attribute values may be missing in the access logs; so, missing attribute values are also handled in this step. In the classification process, missing attribute values are usually replaced by the most common ones. However, the policy extraction approach in this paper is sensitive to the occurrence frequency of each attribute value. Thus, if an attribute is a valid attribute, its missing value is replaced by UNK (unknown) in the corresponding data.

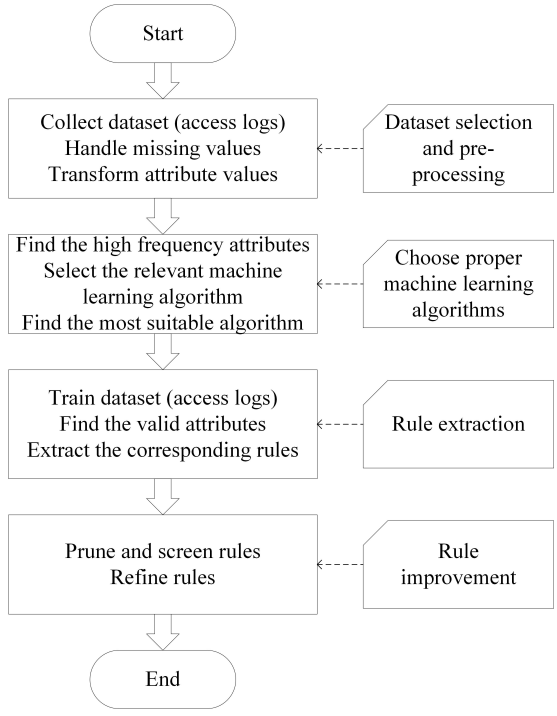

Figure 2. Rule generation process

4.3 Attribute Selection

Owing to the diversity and differences of attributes, the categorical attributes of some entities need to be encoded before attribute selection, primarily by using ordinal encoding and one-hot encoding. We use the recursive feature elimination approach based on CatBoost to select the attributes.

The access log records the attribute information of the subject, object, and environment and the result of the decision operation. First, the attribute combination is transformed into feature vectors. Then, we treat these feature vectors as training data. The decision result of *permit* and *deny* becomes the label of the training data. The model is trained by the CatBoost algorithm, and the trained classification accuracy is regarded as a critical evaluation indicator for attribute deletion. If an attribute is more important, the deletion of the attribute has a more significant impact on the accuracy of the classification algorithm, and its deletion reduces the accuracy of the classification. On the contrary, if an attribute is less important, deleting the attribute has less influence on the accuracy of the classification algorithm.

Then, the scale of the attributes is reduced according to the backward sort method, and the attributes with lower importance are deleted by the iterative method. The Algorithm 1 is a policy attribute selection algorithm based on the greedy elimination algorithm. The final attribute importance sequence is the inverse sequence of the attribute deletion sequence.

Algorithm 1 Attributes selection algorithm

Input: L, A, m // m is a threshold, which is determined by the training model and the access logs

Output: A^*

1: **procedure** SelectAttribute(L, A, m)
2: $A^* \leftarrow \emptyset$
3: $N_a \leftarrow CopyAttributeSet(L, A)$
4: **while** $|N_a| \geq m$ **do**
5: $T_m \leftarrow TrainAccessLogs(L, N_a)$
6: $S \leftarrow SortAttribute(T_m)$
7: $c \leftarrow GetLowerImportanceAttribute(S)$
8: $N_a \leftarrow N_a - \{c\}$
9: **end while**
10: $A^* \leftarrow N_a$
11: **return** A^*
12: **end procedure**

The detailed analysis of the Algorithm 1 is as follows. Line 3, it clones the attribute set. Next, lines 4–9 use the CatBoost algorithm to train the logs and attribute set and sort the attributes; the attributes with lower importance are deleted. Finally, line 10 gets the attributes with higher importance.

4.4 Rule Extraction

Before rule extraction, we propose a method for determining the attribute and operation weight, which is described as follows:

$$AOW = \sum w \cdot A = \sum (w_u \cdot A_u + w_o \cdot A_o + w_e \cdot A_e + w_{op} \cdot OP). \tag{18}$$

Attribute and operation weight (AOW) refers to the weights of the attributes and operations. w_u, w_o, w_e, and w_{op} represent the weight of the user (subject) attributes, object attributes, and environment attributes and operations, respectively. The weights are determined by the training data results of the CatBoost model and are different for different data and models.

Afterward, we need to determine the attribute relation (F), operation set (OP), and the minimum number of times the rule satisfies the access request (T) (Theorem 1) and calculate the confidence degree ($Conf(r)$) and threshold (K) of the rule.

The rule extraction algorithm uses CatBoost to mine frequent sets until no frequent sets can be found.

The Algorithm 2 gives the process of rule extraction in detail.

Algorithm 2 Rule extraction algorithm

Input: A^*, L, D, T, K
Output: R

```
1: procedure ExtractRule(A*, L, D, T, K)
2: R ← ∅
3: F ← GetAttributeRelation(L, A*)
4: OP ← GetOperation(L, D)
5: ar ← GetAccessRequest(F, OP)
6: X ← SaveMatrix(L, F, ar)
7: FreAttrSet ← GetFrequentAttributeSet(X, T)
8: R_c ← GetCandidateRule(FreAttrSet, ar)
9: R_s ← RuleSort(R_c)
10: for all r_i ∈ R_s do
11:     if length(r_i) = 0 then
12:         R_s ← R_s − {r_i}
13:     end if
14: end for
15: for all r_i ∈ R_s do
16:     if Conf(r_i) < K then
17:         R_s ← R_s − {r_i}
18:     end if
19: end for
20: R ← R_s
21: return R
22: end procedure
```

The Algorithm 2 is described as follows. Line 3 gets the attribute relation F. Line 4 gets the corresponding operation set OP. Line 5 gets the corresponding access request ar according to the operation and F. Line 6 finds the F of all access control records in L whose attributes satisfy ar and then saves it as a boolean matrix X. Line 7 finds the frequent attribute set $FreAttrSet$ in X. Lines 8–9 get the candidate rules and the length and then sort them by number of attributes. Lines 10–21 screen candidate rules according to the confidence degree and threshold K and then obtain the rule set R.

4.5 Rule Pruning and Rule Optimization

The rule pruning and rule optimization algorithms are used to improve the quality of the ABAC policies. During the training, two or more sets may map to the same

rule. If there are two similar rules, the difficulty and complexity of policy mining are higher and may also affect the accuracy of the policy, which can reduce the quality of the policy. To solve this problem, we find similar rules, calculate their similarity, and then delete rules that do not affect the quality of the policy. If removing either of these rules does not improve the quality of the policy, we keep both rules. This may happen when there are two similar ABAC rules in the actual rule.

We use Jaccard similarity to measure the similarity between two rules, as follows:

Definition 16. Given two sets, A and B, the Jaccard coefficient is defined as the ratio of the size of the intersection of A and B and the size of the union of A and B. The calculation formula is as follows:

$$J(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|} = \frac{|A \bigcap B|}{|A| + |B| - |A \bigcap B|}. \tag{19}$$

When sets A and B are empty, $J(A,B)$ is defined as 1.

According to the Theorem 16, we can calculate the similarity between rules r_1 and r_2. The formula is as follows:

$$J(r_1, r_2) = \frac{\sum_{v \in \{V(F), V(op)\}} |v_{r_1} \bigcap v_{r_2}|}{\sum_{v \in \{V(F), V(op)\}} |v_{r_1} \bigcup v_{r_2}|}, \tag{20}$$

where v represents their attribute domain, and the calculated results can determine their similarity. We consider that if the Jaccard similarity score is greater than 0.5, there is a significant overlap between them, and the two rules can be considered similar. This means that the size of their common elements is more than half the size of their union of elements. The Algorithm 3 gives the detailed procedure of rule pruning.

The details of the Algorithm 3 are as follows. The input is the set of rules R obtained by the Algorithm 2 and a similarity threshold δ, and the output is the trimmed rule R^*. Lines 3–21 calculate the similarity and select rules: If similarity of two rules is greater than or equal to a given threshold δ, put the two rules in set R_{t1}, and then, calculate the quality of $R - \{r_i\}$ to determine which can improve the quality of the policy, and save them to set R_{t2}. If there are multiple similar rules and their quality is greater than q, delete the one with the worst quality. Finally, we obtain a new rule set R^*.

Definition 17 (Rule Conflict Relation)**.** Given a rule set (R) and two rules $(r_1, r_2) \in R$, if $r_1 = r_2$, and one of them has the decision result of *permit* and the other has *deny*, then r_1 and r_2 have a conflicting relation.

Definition 18 (Rule Hierarchy Relation)**.** Given a rule set (R) and two rules (r_1, $r_2 \in R$), r_1 is called the senior rule of r_2, denoted as $r_1 \leq r_2$, and r_1 and r_2 have a rule hierarchical relation if $(F_{u_{r_1}} \subseteq F_{u_{r_2}}) \wedge (F_{o_{r_1}} \subseteq F_{o_{r_2}}) \wedge (F_{e_{r_1}} \subseteq F_{e_{r_2}})$, $op_{r_1} = op_{r_2}$,

Algorithm 3 Rule pruning algorithm

Input: R, δ // δ is a threshold, tentatively set at 0.5
Output: R^*

procedure PruneRule(R, δ)
$R^* \leftarrow \emptyset; R_{t1} \leftarrow \emptyset; R_{t2} \leftarrow \emptyset$
$q(R) \leftarrow$ *CalculateRuleQuality*(R) // $q(R)$ refers to the accuracy
for all $r_i \in R$ **do**
    for all $r_j \in R$ and $r_i \neq r_j$ **do**
        if Similarity$(r_i, r_j) \geq \delta$ **then**
            $R_{t1} \leftarrow R_{t1} \cup \{r_i, r_j\}$
        end if
    end for
end for
for all $r_i \in R_{t1}$ **do**
    $q(r_i) \leftarrow$ *CalculateRuleQuality*$(R - \{r_i\})$
    if $q(r_i) \geq q(R)$ **then**
        $R_{t2} \leftarrow R_{t2} \cup \{r_i\}$
    end if
end for
for all $r_i \in R_{t2}$ **do**
    if $q(r_i) == \min\{q(r_i), r_i \in R_{t2}\}$ **then**
        $R^* \leftarrow R - \{r_i\}$
    end if
end for
return R^*
end procedure

$d_{r_1} = d_{r_2}$. It includes rule-inclusion relation $((F_{u_{r_1}} \subset F_{u_{r_2}}) \wedge (F_{o_{r_1}} \subset F_{o_{r_2}}) \wedge (F_{e_{r_1}} \subset F_{e_{r_2}})$, $op_{r_1} = op_{r_2}$, $d_{r_1} = d_{r_2}$) and rule-equality relation $((F_{u_{r_1}} = F_{u_{r_2}}) \wedge (F_{o_{r_1}} = F_{o_{r_2}}) \wedge (F_{e_{r_1}} = F_{e_{r_2}})$, $op_{r_1} = op_{r_2}$, $d_{r_1} = d_{r_2}$).

If r_1 is senior to r_2, then r_2 is a more restrictive rule than r_1. If there are hierarchical rules r_1 and r_2 ($r_1 \leq r_2$) in the ABAC system, redundancy occurs, which leads to more FP in the decision. To reduce FP, we prune the extracted rule set by removing the overly permissive rule (r_1).

For example, here are two rules:

$$r_1 = (\{\langle Position, Student\rangle, \langle Location, Campus\rangle, \langle Type, Book\rangle\}, borrow, permit),$$

$$r_2 = (\{\langle Position, Student\rangle, \langle Location, Campus\rangle, \langle Time, 10{:}00\text{–}22{:}00\rangle, \langle Type, Book\rangle\}, borrow, permit).$$

Rule r_1 permits students to borrow books on campus, whereas rule r_2 permits students to borrow books on campus only during the specified time period. Here,

r_1 is the senior rule of r_2; so, r_2 is more restrictive. Therefore, access requests that would otherwise be denied are permitted owing to the overly permissive rule r_1, resulting in higher FP. To reduce the potential of more FP, we remove r_1 from the extracted rule set.

In the training process, owing to the lack of some samples in the training data or other reasons, some rules may be missing in the generated ABAC rules. Thus, some rules are ignored in the rule pruning process. This problem inevitably leads to FN because according to the missing rule, the access request that was originally permitted is denied owing to the generated rule. However, if some attribute relations are omitted in the process of extracting rules, for example, some attributes or attribute values of a rule are lost, the extracted rules are more relaxed than the actual rules, and the decisions that should be denied according to the actual rules are permitted instead. As mentioned in the above example, r_1 is more permissive than r_2, resulting in the FP phenomenon.

We provide rule optimization algorithms to solve the above problem, as shown in Algorithm 4. This process is similar to the training process in ML, where the training data includes the access control decisions that produce FP or FN. For example, in the FP scenario, a request should be denied according to the actual policy. Despite this, it is permitted according to the generated policy owing to the extraction of overly permissive rules in the policy generation process. In the FN scenario, a request should be permitted according to the actual policy. Moreover, it is denied according to the generated policy, which is closely related to the missing samples in the data.

The Algorithm 4 is described as follows. Through k-fold cross-validation, we obtain classification evaluation indicators (such as *FNR*, *FPR*, and *Acc*) to determine the generated rule results. In addition, we check if there are hierarchical (inclusive or equal) relations and conflicting relations among the rules using the accuracy. Lines 7–25 handle rules with hierarchy and conflict by preserving one of the equal rules and deleting the senior rule and the conflict rules. We finally achieve an optimized rule set R_{Q_p}.

For policies generated on the access logs, to improve the quality of mining the ABAC policy, we combined *WSC* and *Accuracy* (*Acc*) to define the comprehensive indicator of policy quality. The formula is as follows:

$$Q_p = \frac{1}{\frac{\alpha}{Acc} + \frac{1-\alpha}{\Delta WSC}}, \tag{21}$$

where ΔWSC is calculated as follows:

$$\Delta WSC = \frac{WSC_{max} - WSC_{\pi}}{WSC_{max} - WSC_{min}}. \tag{22}$$

When $WSC_{\pi} = WSC_{max}$, $\Delta WSC = 0$, define $Q_p = 0$. When $WSC_{\pi} = WSC_{min}$, $\Delta WSC = 1$, it indicates no policy generated; so, also define $Q_p = 0$. Let $\alpha = \frac{1}{1+\beta^2}$, $\beta \in \mathbb{R}$ in (21), then β determines the importance degree of *Acc* to

Algorithm 4 Rule optimization algorithm

Input: D, L, R^*
Output: R_{Q_p}

1: **procedure** OptimizeRule(D, L, R^*)
2: $Acc \leftarrow GetIndicator_C(D, L, R^*)$
3: $WSC \leftarrow GetWSC(R^*)$
4: $\Delta WSC \leftarrow GetIndicator_W(R^*, WSC)$
5: $Q_p \leftarrow FindParameter(Acc, \Delta WSC)$
6: $R_{Q_p} \leftarrow FilterRule(R^*, Q_p)$
7: **for** all $r_o \in R_{Q_p}$ **do**
8: $R_o \leftarrow FindRule(R_{Q_p}, r_o)$
9: **if** $R_o \neq \emptyset$ **then**
10: **for** all $r_i, r_j \in R_o$ **do**
11: **if** $r_i == r_j$ **then**
12: $R_{Q_p} \leftarrow R_{Q_p} - (\{r_i\}\ or\ \{r_j\})$
13: **end if**
14: **if** $r_j \leq r_i$ **then**
15: $R_{Q_p} \leftarrow R_{Q_p} - \{r_j\}$
16: **else**
17: $R_{Q_p} \leftarrow R_{Q_p} - \{r_i\}$
18: **end if**
19: **if** r_i Conflicts with r_j **then**
20: $R_{Q_p} \leftarrow R_{Q_p} - \{r_i\} - \{r_j\}$
21: **end if**
22: **end for**
23: **end if**
24: **end for**
25: **return** R_{Q_p}
26: **end procedure**

the policy complexity. When $\beta = 1$, the two indicators Acc and ΔWSC have the same weight, indicating the same importance. When $\beta < 1$, the weight of Acc is significant, indicating that Acc is more important. When $\beta > 1$, ΔWSC has a significant weight, meaning that ΔWSC is more important. ΔWSC is the normalized value of the WSC, putting the ΔWSC value in the interval [0, 1]. WSC_{max} and WSC_{min} represent the complexity of the weighted structure of the most complex and simplest policies, respectively. The most complex policy can be understood as the policy corresponding to each access control decision that contains all attributes of the subject, object, and environment. The simplest policy can be understood as the null policy, namely $WSC_{min} = 0$.

In addition, the Algorithm 5 shows the policy generation step in detail. The Algorithm 5 is described as follows. Line 2 is for preprocessing the access logs. Line 3 is for selecting attributes. Lines 4–5 are for extracting and pruning rules.

Line 6 calculates the policy/rule quality indicator $Q(WSC, Acc)$. Finally, lines 7–10 are used to refine the mined rules until the best rule set is obtained.

Algorithm 5 CatBoost-based ABAC policy generation algorithm

Input: L, m, D, T, δ, K, Q // Q refers to the policy/rule quality indicator (WSC, Acc)

Output: R_e

1: **procedure** GenerateABACRule$(L, m, D, T, \delta, K, Q)$
2: $A \leftarrow Preprocess(L)$
3: $A^* \leftarrow SelectAttribute(L, A, m)$
4: $R \leftarrow ExtractRule(A^*, L, D, T, K)$
5: $R^* \leftarrow PruneRule(R, \delta)$
6: $q \leftarrow CalculateRuleQuality(R^*)$
7: **while** $q < Q$ or $Acc \geq 0.95$ **do**
8: $\quad R_{Q_p} \leftarrow OptimizeRule(D, L, R^*)$
9: $\quad q \leftarrow CalculateRuleQuality(R_{Q_p})$
10: **end while**
11: **return** R_{Q_p}
12: **end procedure**

Some parameters are additionally used to adjust the model to improve the accuracy. We use $p(0)$ and $p(1)$ to denote the probability estimations of the *permit* and *deny* decision against the access requests, respectively. If $p(1) > p(0)$, the decision is *permit*; if $p(1) < p(0)$, the decision is *deny*; and if $p(1) = p(0)$, the decision cannot be determined.

Next, we use the cross entropy to calculate the loss of this model because the training goal is to minimize the cross entropy of the two categories (*permit* and *deny*). The calculation formula is as follows:

$$LF = -\sum_{i=1}^{n}\{w(0) \cdot y_i(0) \cdot \log[p_i(0)] + w(1) \cdot y_i(1) \cdot \log[p_i(1)]\}, \tag{23}$$

where n is the number of access requests in the training process; and $y_i(0)$, $y_i(1)$ represent the decision to *deny* and *permit*, respectively.

If the decision results are wrong, $\lceil y_i(0), y_i(1) \rfloor = \lceil 1, 0 \rfloor$. If decision results are correct, $\lceil y_i(0), y_i(1) \rfloor = \lceil 0, 1 \rfloor$. When the probability estimations of $y_i(0)$ and $y_i(1)$ are $p_i(0)$ and $p_i(1)$, respectively, and any decision result is correct, for all access requests, each decision completely matches the actual decision. This indicates that the loss function has reached its minimum absolute value and is equal to 0. In the loss function, to balance the *permit* and *deny* decision results in the data training process, we define the weights $w(1) = \frac{(N-N_1)}{N}$ and $w(0) = 1 - w(1)$, where $w(1)$ and $w(0)$ are the weights of *permit* and *deny*, respectively. N is the input quantity of the dataset, and N_1 is the number of *permit* decisions.

5 EVALUATION

5.1 Simple Evaluation

We compared the ABAC policy generation approach from access logs based on the CatBoost with five related approaches in the following nine aspects. The comparison results are shown in Table 3.

	Xu et al. [23]	Medvet et al. [28]	Iyer et al. [24]	Cotrini et al. [30]	Mocanu et al. [38]	Ours
Attribute relation	no	no	no	no	no	yes
Negative decision rule	no	no	yes	no	no	yes
Sparse logs	no	yes	no	yes	yes	yes
Noise logs	yes	no	no	no	yes	yes
WSC	yes	yes	yes	no	yes	yes
Policy accuracy	yes	yes	yes	yes	no	yes
Policy complexity	yes	yes	yes	yes	yes	yes
Attribute and operation weight	no	no	no	no	no	yes
Policy quality comprehensive indicator	no	no	no	no	no	yes

Table 3. Comparison of ABAC policy mining approaches

5.2 Experimental Evaluation

5.2.1 Dataset Introduction and Experimental Settings

The experimental environment was set as follows. The processor was based on X64, and the parameters were Intel(R) Core(TM) I7-8565U CPU @ 1.80 GHz 1.99 GHz. The random access memory (RAM) was 8 GB. The operating system was Windows 10 Home version (64-bit). The version number was 21H1. The experimental platform was Anaconda 3-2022.05 version, and the interpreter was Python version 3.9.12. The ABAC policy generation approach was implemented based on CatBoost.

The experimental dataset was from the "Amazon.com-Employee Access Challenge" competition on the Kaggle platform, divided into the training set and test set. For short, this is called the Amazon-employee dataset.[2] This consists of real historical data from 2010 and 2011. Each access tuple in this dataset comprises the tuple corresponding to an employee's access request to a resource and displays the corresponding decision (*permit* or *deny*) result. Its access log is composed of employee attribute values and resource identifiers. It has many subject attributes but a relatively small number of log entries and a sparse log set. Moreover, there is

[2] `https://www.kaggle.com/c/amazon-employee-access-challenge/forums/t/5283/winning-solution-code-and-methodology`

only one object attribute, which may lead to biased experimental results. Table 4 shows the basic information of the training set in this dataset; there are a total of 32 769 access control records.

Attribute Name	Attribute Type	Attribute Information	Attribute Number
ACTION	Operation attribute	ACTION is 1 if the resource was approved and 0 if not.	2
RESOURCE	Object attribute	An ID for each resource	7 518
MGR_ID	Subject attribute	The EMPLOYEE ID of the manager of the current EMPLOYEE ID record; an employee may have only one manager at a time	4 243
ROLE_ROLLUP_1	Subject attribute	Company role grouping category id1 (e.g. US Engineering)	128
ROLE_ROLLUP_2	Subject attribute	Company role grouping category id2 (e.g. US Retail)	177
ROLE_DEPTNAME	Subject attribute	Company role department description (e.g. Retail)	449
ROLE_TITLE	Subject attribute	Company role business title description (e.g. e.g. Senior Engineering, Retail Manager)	343
ROLE_FAMILY_DESC	Subject attribute	Company role family extended description (e.g. Retail Manager, Software Engineering)	2 358
ROLE_FAMILY	Subject attribute	Company role family description (e.g. Retail Manager)	67
ROLE_CODE	Subject attribute	Company role code: this code is unique to each role (e.g. Manager)	343

Table 4. Dataset information

5.2.2 Experiments and Comparison

Through experiments, we compared our approach with [38], which used the RBM algorithm to infer ABAC policies from logs, and [39], which used the MLP algorithm to infer the ABAC policies and made a detailed comparison in the following aspects. Finally, our approach is superior to theirs in most cases, but it needs to be revised in some cases, and it will be analyzed in detail.

Figure 3 compares the attribute importance of the two approaches. It can be seen that after using different approaches, there are apparent differences in the importance of different attributes. For example, it can be seen that the importance of RESOURCE attribute in the [38], which used RBM is almost 0. In our approach,

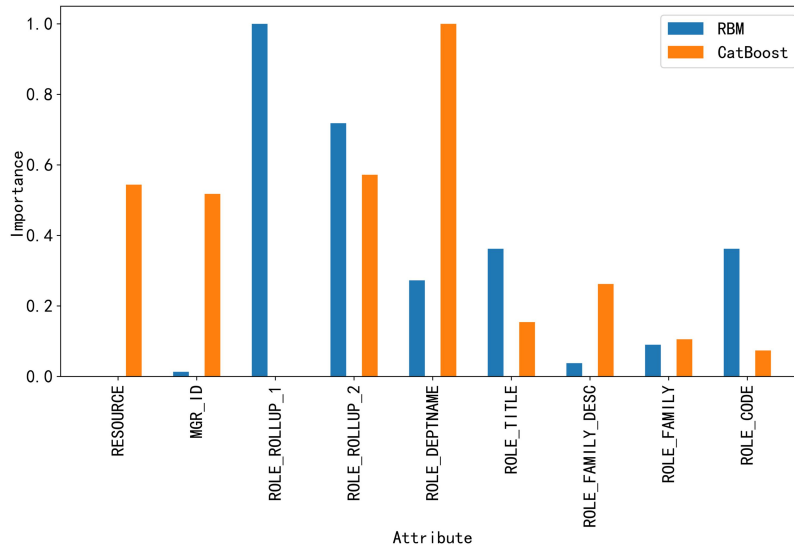

Figure 3. Attribute importance comparison graph

it is 0.77. The attribute ROLE_DEPTNAME has the highest importance, close to 1. The attribute ROLE_ROLLUP_1 has the lowest importance, which is almost 0. The attribute RESOURCE belongs to the object attribute, which is single and extremely important. The attribute ROLE_ROLLUP_1 belongs to the subject attributes and has the lowest importance. Because the MLP algorithm used in [39] is a nonlinear classifier, it cannot analyze the importance of attributes, so it cannot weigh them. But, of course, it can be understood that the weights are the same, and all are equal (specified as 1).

Through the k-fold cross-validation method, we divided the training set in the Amazon Employee dataset into five parts on average, using 1 part as the training set each time and the remaining four parts as the test set. After five times averaging, we obtained our final experimental results. Figure 4 shows the receiver operating characteristic (ROC) curves of the three approaches, which show the accuracy of each approach's access control decision results. According to the area enclosed by the ROC curve and the coordinate axis, that is, the area under curve (AUC) value, we can see that the AUC value obtained by our approach is 0.978. It is slightly higher than the AUC value 0.972, obtained by the RBM algorithm in [38], and the AUC value 0.97, obtained by the MLP algorithm in [39], respectively. It indicates that the accuracy of the decision results obtained by our approach is higher than that obtained by the [38] and [39] approaches.

Figure 5 shows the relations between the access control decision precision and the recall rate. It indicates that the decision precision decreases with the increase of recall rate. It can be seen that there is a turning point when the recall rate is

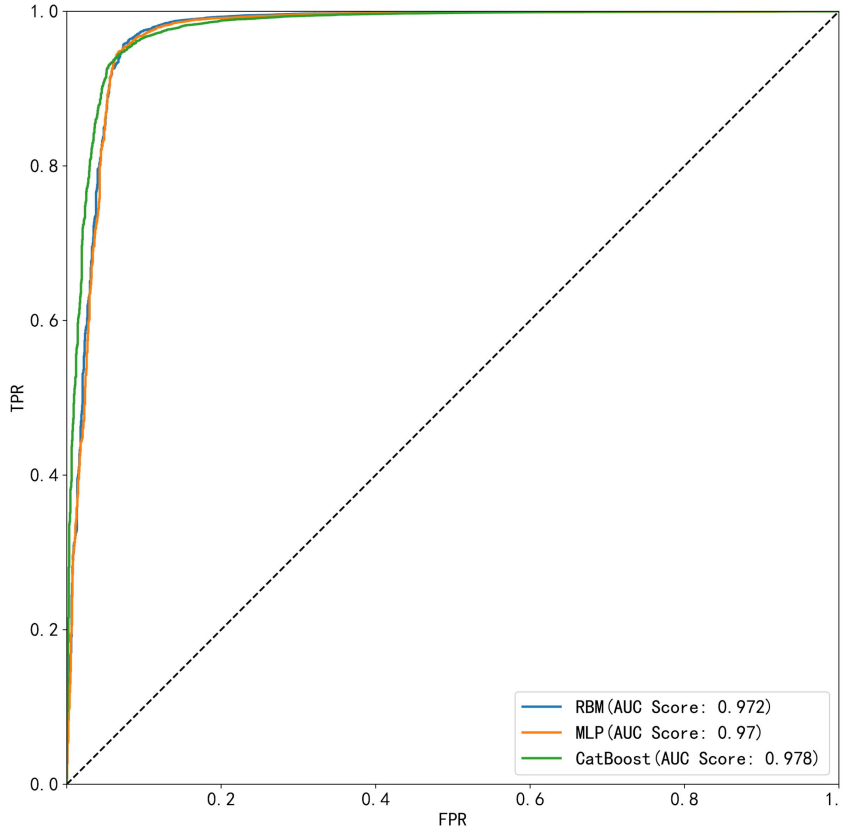

Figure 4. ROC curve

about 0.95. When the recall rate is less than 0.95, the decision precision of our approach is higher than that obtained by using the RBM algorithm and the MLP algorithm in [39]. When the recall rate is greater than 0.95, the decision precision of our approach is slightly lower than that of the approaches in [38] and [39]. On the whole, the decision precision of our approach is better than theirs. By comparing the precision and the value of AUC, we can see that the decision result of our approach is superior to that of [38] and [39].

Our approach is compared with the decision results obtained using the RBM algorithm in [38] and the MLP algorithm in [39] through k-fold cross-validation. In Table 5, it was evident that our approach is superior to the approach used in [38] and [39] in terms of *TPR* and *FNR* in both the test set and training set and inferior to the approach used in [38] and [39] in terms of *TNR*, *FPR*, and *Mcc*. In terms of *precision*, our approach is slightly inferior to that in [38] and [39]. Only in the test set, our approach is slightly superior to that in [38]. In the training set, the

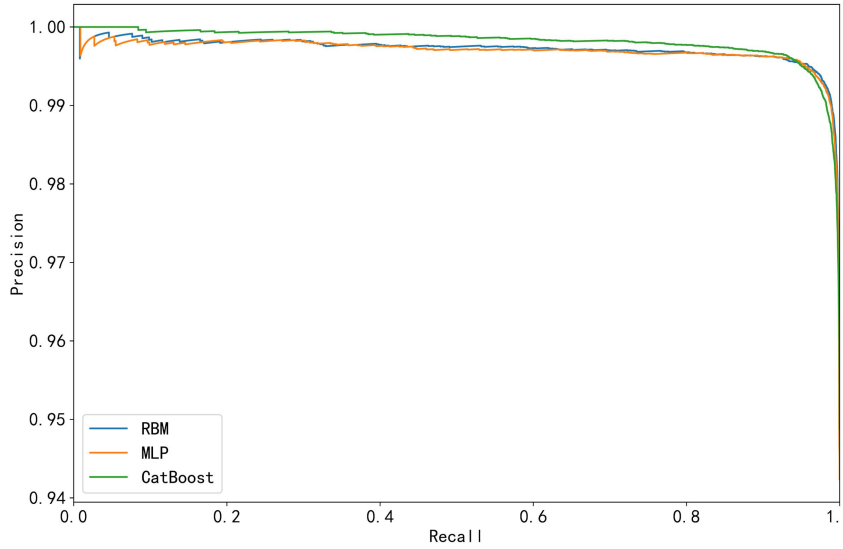

Figure 5. P-R curve

values of *Acc*, *F1-score*, and the AUC of our approach are slightly inferior to those used in [38] and [39]. However, in the test set, our approach is superior to theirs. The decision accuracy of our approach in the test set is 95.74 %. It is higher than 94.54 % in [38] and 93.13 % in [39]; the AUC value in the test set is 90.37 %, which is much higher than 84.87 % in [38] and 84.02 % in [39].

Approach	TPR	TNR	FPR	FNR
RBM-training	0.996312	0.903896	0.096104	0.003688
RBM-test	0.978699	0.366947	0.633053	0.021301
MLP-training	0.992705	0.945455	0.054545	0.007295
MLP-test	0.959819	0.436975	0.563025	0.040181
CatBoost-training	0.997933	0.672078	0.327922	0.002067
CatBoost-test	0.991770	0.361345	0.638655	0.008230

Approach	Precision	Accuracy	F1-score	Mcc	AUC
RBM-training	0.994016	0.990883	0.995163	0.916287	0.998517
RBM-test	0.964076	0.945377	0.971332	0.399686	0.848675
MLP-training	0.996582	0.989929	0.994640	0.911994	0.998199
MLP-test	0.967312	0.931340	0.963551	0.373990	0.840196
CatBoost-training	0.979904	0.978791	0.988836	0.790621	0.994601
CatBoost-test	0.964230	0.957431	0.977806	0.490341	0.903660

Table 5. Comparison of the predicted results for access control decisions

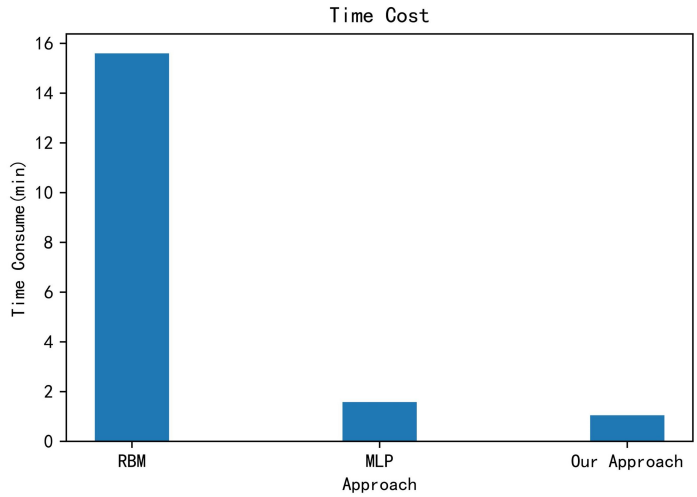

Figure 6. Comparison of time consumption

Figure 6 shows the comparison of the time consumed by our approach and the two approaches used in [38] and [39] in making decisions on test sets. Through cross-validation, we made access control decision predictions on the training set containing 80 %, with 26 215 access control records. Our approach consumes less time than the approaches used in the [38] and the [39].

5.2.3 Result Analysis

By comparing various aspects of the three approaches, it is reasonable to choose the ABAC policy generation approach based on the CatBoost. Figure 7 shows the loss function obtained by our approach in the training set. It can be seen that with the increase in the number of iterations, the value of the loss function gradually decreases and tends to be stable when the number of iterations is about 3 000. Therefore, we decide to use the number of iterations to conduct the final experiment 3 000 times.

Figures 8 and 9 represent the confusion matrix of the access logs before and after rule pruning and rule optimization, respectively. As shown in Figures 8 and 9, the access control decision prediction was made on the training set containing 80 %, and there were a total of 26 215 access control records. After pruning and optimization, the number of TP records increased from 24 597 to 24 624; the number of TN records increased from 900 to 1 035; and the accuracy improved considerably. The number of wrong decisions decreased by 162; FP records decreased from 640 to 505; and FN records decreased from 78 to 51. The accuracy rate of the decision results was improved from 97.46 % to 98.00 %.

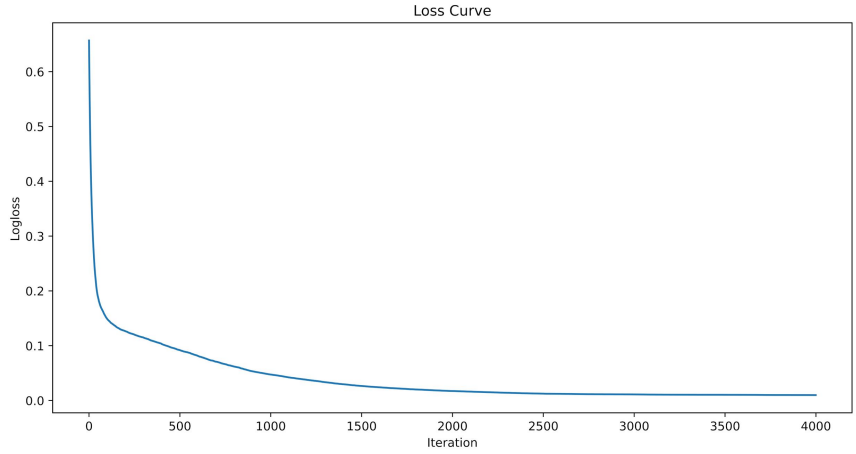

Figure 7. Loss iteration graph

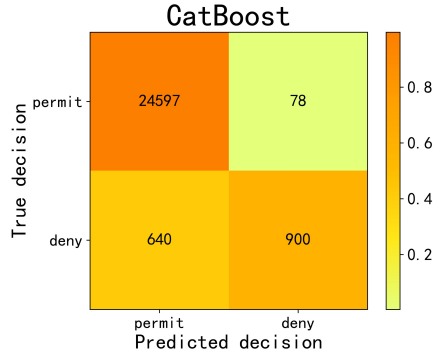

Figure 8. Before rule pruning and optimization

To sum up, the ABAC policy generation approach based on the CatBoost algorithm is a decision approach to predict access decisions according to historical access logs (or access control records). Through experiments, we compared it with the two approaches used in [38] and [39]. The *precision* and Mcc are inferior to these approaches, but our approach is superior to them in other aspects, especially the accuracy of the decision results and the time consumption. However, our approach assumes that the decision result is only *permit* or *deny*. In the actual access control scenario, there will be more complex decision results, such as decision conflict (that is, both *permit* and *deny*), which is the disadvantage of this approach.

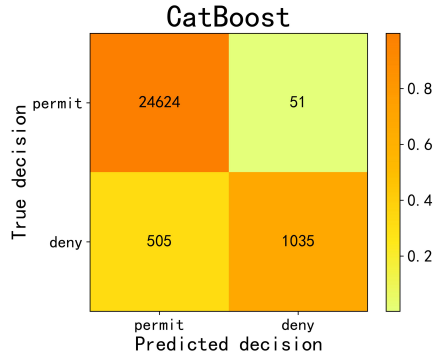

Figure 9. After rule pruning and optimization

6 CONCLUSION

This paper provided an approach to automate ABAC policy (rule) generation via the CatBoost ML algorithm. This approach can discover both positive and negative ACPs. We presented an attribute selection algorithm by weighted reconstruction of the attributes in the quasi-generation policy, thus improving the validity of rule extraction. The rule extraction algorithm, rule pruning algorithm, and rule optimization algorithm were also proposed to improve the precision of the generated policy and significantly improve the accuracy of the generated policy. Most importantly, we proposed a new policy quality indicator, namely the policy quality comprehensive indicator, to measure the accuracy and simplicity of the policy. It is essential to compare the generated policy with the actual policy for further refinement. We evaluated the presented approach on the Amazon-employee dataset and verified its feasibility, effectiveness, and practicability. Finally, through experiments, we demonstrated that although *FPR*, *precision*, and other aspects are slightly inferior to approaches [38] and [39], our approach is superior to theirs in terms of the accuracy and time consume of the generated policy.

In future work, we will continue to improve our approach's accuracy and simplicity, further study how to resolve conflicts and undecidability in access control decisions and research other factors that influence the quality of generated policies.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (Grant No. 61862059).

Conflicts of interest

We declare that we have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

[1] Feng, D.—Zhang, M.—Li, H.: Big Data Security and Privacy Protection. Chinese Journal of Computers, Vol. 37, 2014, No. 1, pp. 246–258, doi: 10.3724/SP.J.1016.2014.00246 (In Chinese).

[2] Li, H.—Zhang, M.—Feng, D.—Hui, Z.: Research on Big Data Access Control. Chinese Journal of Computers, Vol. 40, 2017, No. 1, pp. 72–91, doi: 10.11897/SP.J.1016.2017.00072 (In Chinese).

[3] Graham, G. S.—Denning, P. J.: Protection: Principles and Practice. Proceedings of the May 16-18, 1972, Spring Joint Computer Conference (AFIPS '72 (Spring)), ACM, 1971, pp. 417–429, doi: 10.1145/1478873.1478928.

[4] Sandhu, R. S.—Samarati, P.: Access Control: Principles and Practice. IEEE Communications Magazine, Vol. 32, 1994, No. 9, pp. 40–48, doi: 10.1109/35.312842.

[5] Sandhu, R. S.: Lattice-Based Access Control Models. Computer, Vol. 26, 1993, No. 11, pp. 9–19, doi: 10.1109/2.241422.

[6] Sandhu, R. S.—Coyne, E. J.—Feinstein, H. L.—Youman, C. E.: Role-Based Access Control Models. Computer, Vol. 29, 1996, No. 2, pp. 38–47, doi: 10.1109/2.485845.

[7] Hu, V. C.—Ferraiolo, D.—Kuhn, R.—Schnitzer, A.—Sandlin, K.—Miller, R.—Scarfone, K.: Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST Special Publication 800-162. National Institute of Standards and Technology, Gaithersburg, MD, 2014, doi: 10.6028/NIST.SP.800-162.

[8] Wang, X.—Fu, H.—Zhang, L.: Research Progress on Attribute-Based Access Control. Acta Electronica Sinica, Vol. 38, 2010, No. 7, pp. 1660–1667 (In Chinese).

[9] Fang, L.—Yin, L.—Guo, Y.—Fang, B.: A Survey of Key Technologies in Attribute-Based Access Control Scheme. Chinese Journal of Computers, Vol. 40, 2017, No. 7, pp. 1680–1698, doi: 10.11897/SP.J.1016.2017.01680 (In Chinese).

[10] Das, S.—Mitra, B.—Atluri, V.—Vaidya, J.—Sural, S.: Policy Engineering in RBAC and ABAC. In: Samarati, P., Ray, I., Ray, I. (Eds.): From Database to Cyber Security: Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday. Springer, Cham, Lecture Notes in Computer Science, Vol. 11170, 2018, pp. 24–54, doi: 10.1007/978-3-030-04834-1_2.

[11] Xu, Z.—Stoller, S. D.: Mining Attribute-Based Access Control Policies from Logs. In: Atluri, V., Pernul, G. (Eds.): Data and Applications Security and Privacy XXVIII (DBSec 2014). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 8566, 2014, pp. 276–291, doi: 10.1007/978-3-662-43936-4_18.

[12] Vaidya, J.—Atluri, V.—Guo, Q.: The Role Mining Problem: Finding a Minimal Descriptive Set of Roles. Proceedings of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT '07), 2007, pp. 175–184, doi: 10.1145/1266840.1266870.

[13] Molloy, I.—Chen, H.—Li, T.—Wang, Q.—Li, N.—Bertino, E.—Calo, S.—Lobo, J.: Mining Roles with Multiple Objectives. ACM Transactions on Information and System Security, Vol. 13, 2010, No. 4, Art. No. 36, doi: 10.1145/1880022.1880030.

[14] Molloy, I.—Li, N.—Qi, Y.—Lobo, J.—Dickens, L.: Mining Roles with Noisy Data. Proceedings of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT '10), 2010, pp. 45–54, doi: 10.1145/1809842.1809852.

[15] Currey, J.—McKinstry, R.—Dadgar, A.—Gritter, M.: Informed Privilege-Complexity Trade-Offs in RBAC Configuration. Proceedings of the 25th ACM Symposium on Access Control Models and Technologies (SACMAT '20), 2020, pp. 119–130, doi: 10.1145/3381991.3395597.

[16] Jafarian, J. H.—Takabi, H.—Touati, H.—Hesamifard, E.—Shehab, M.: Towards a General Framework for Optimal Role Mining: A Constraint Satisfaction Approach. Proceedings of the 20th ACM Symposium on Access Control Models and Technologies (SACMAT '15), 2015, pp. 211–220, doi: 10.1145/2752952.2752975.

[17] Molloy, I.—Park, Y.—Chari, S.: Generative Models for Access Control Policies: Applications to Role Mining over Logs with Attribution. Proceedings of the 17th ACM Symposium on Access Control Models and Technologies (SACMAT '12), 2012, pp. 45–56, doi: 10.1145/2295136.2295145.

[18] Narouei, M.—Takabi, H.: Towards an Automatic Top-Down Role Engineering Approach Using Natural Language Processing Techniques. Proceedings of the 20th ACM Symposium on Access Control Models and Technologies (SACMAT '15), 2015, pp. 157–160, doi: 10.1145/2752952.2752958.

[19] Narouei, M.—Takabi, H.: Automatic Top-Down Role Engineering Framework Using Natural Language Processing Techniques. In: Akram, R. N., Jajodia, S. (Eds.): Information Security Theory and Practice (WISTP 2015). Springer, Cham, Lecture Notes in Computer Science, Vol. 9311, 2015, pp. 137–152, doi: 10.1007/978-3-319-24018-3_9.

[20] Anderer, S.—Scheuermann, B.—Mostaghim, S.—Bauerle, P.—Beil, M.: RMPlib: A Library of Benchmarks for the Role Mining Problem. Proceedings of the 26th ACM Symposium on Access Control Models and Technologies (SACMAT '21), 2021, pp. 3–13, doi: 10.1145/3450569.3463566.

[21] Chari, S. N.—Molloy, I. M.: Generation of Attribute Based Access Control Policy from Existing Authorization System. Google Patents, 2016 (US Patent US9264451B2).

[22] Xu, Z.—Stoller, S. D.: Mining Attribute-Based Access Control Policies from RBAC Policies. 2013 10th International Conference and Expo on Emerging Technologies for a Smarter World (CEWIT), IEEE, 2013, pp. 1–6, doi: 10.1109/CEWIT.2013.6713753.

[23] Xu, Z.—Stoller, S. D.: Mining Attribute-Based Access Control Policies. IEEE

Transactions on Dependable and Secure Computing, Vol. 12, 2015, No. 5, pp. 533–545, doi: 10.1109/TDSC.2014.2369048.

[24] Iyer, P.—Masoumzadeh, A.: Mining Positive and Negative Attribute-Based Access Control Policy Rules. Proceedings of the 23rd ACM Symposium on Access Control Models and Technologies (SACMAT '18), 2018, pp. 161–172, doi: 10.1145/3205977.3205988.

[25] Chakraborty, S.—Sandhu, R.—Krishnan, R.: On the Feasibility of Attribute-Based Access Control Policy Mining. 2019 20th IEEE International Conference on Information Reuse and Integration for Data Science (IRI), 2019, pp. 245–252, doi: 10.1109/IRI.2019.00047.

[26] Talukdar, T.—Batra, G.—Vaidya, J.—Atluri, V.—Sural, S.: Efficient Bottom-Up Mining of Attribute Based Access Control Policies. 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), 2017, pp. 339–348, doi: 10.1109/CIC.2017.00051.

[27] Narouei, M.—Takabi, H.: A Nature-Inspired Framework for Optimal Mining of Attribute-Based Access Control Policies. Security and Privacy in Communication Networks (SecureComm 2019), Springer, Cham, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 305, 2019, pp. 489–506, doi: 10.1007/978-3-030-37231-6_29.

[28] Medvet, E.—Bartoli, A.—Carminati, B.—Ferrari, E.: Evolutionary Inference of Attribute-Based Access Control Policies. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C. C. (Eds.): Evolutionary Multi-Criterion Optimization (EMO 2015). Springer, Cham, Lecture Notes in Computer Science, Vol. 9018, 2015, pp. 351–365, doi: 10.1007/978-3-319-15934-8_24.

[29] Das, S.—Sural, S.—Vaidya, J.—Atluri, V.: Using Gini Impurity to Mine Attribute-Based Access Control Policie with Environment Attributes. Proceedings of the 23rd ACM Symposium on Access Control Models and Technologies (SACMAT '18), 2018, pp. 213–215, doi: 10.1145/3205977.3208949.

[30] Cotrini, C.—Weghorn, T.—Basin, D.: Mining ABAC Rules from Sparse Logs. 2018 IEEE European Symposium on Security and Privacy, 2018, pp. 31–46, doi: 10.1109/EuroSP.2018.00011.

[31] Karimi, L.—Joshi, J.: An Unsupervised Learning Based Approach for Mining Attribute Based Access Control Policies. 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 1427–1436, doi: 10.1109/BigData.2018.8622037.

[32] Das, S.—Sural, S.—Vaidya, J.—Atluri, V.—Rigoll, G.: VisMAP: Visual Mining of Attribute-Based Access Control Policies. In: Garg, D., Kumar, N. V. N., Shyamasundar, R. K. (Eds.): Information Systems Security (ICISS 2019). Springer, Cham, Lecture Notes in Computer Science, Vol. 11952, 2019, pp. 79–98, doi: 10.1007/978-3-030-36945-3_5.

[33] Narouei, M.—Khanpour, H.—Takabi, H.—Parde, N.—Nielsen, R.: Towards a Top-Down Policy Engineering Framework for Attribute-Based Access Control. Proceedings of the 22nd ACM Symposium on Access Control Models and Technologies (SACMAT '17), 2017, pp. 103–114, doi: 10.1145/3078861.3078874.

[34] Narouei, M.—Takabi, H.—Nielsen, R.: Automatic Extraction of Access

Control Policies from Natural Language Documents. IEEE Transactions on Dependable and Secure Computing, Vol. 17, 2020, No. 3, pp. 506–517, doi: 10.1109/TDSC.2018.2818708.

[35] Alohaly, M.—Takabi, H.—Blanco, E.: A Deep Learning Approach for Extracting Attributes of ABAC Policies. Proceedings of the 23rd ACM Symposium on Access Control Models and Technologies (SACMAT '18), 2018, pp. 137–148, doi: 10.1145/3205977.3205984.

[36] Alohaly, M.—Takabi, H.—Blanco, E.: Automated Extraction of Attributes from Natural Language Attribute-Based Access Control (ABAC) Policies. Cybersecurity, Vol. 2, 2019, Art. No. 2, doi: 10.1186/s42400-018-0019-2.

[37] Karimi, L.—Abdelhakim, M.—Joshi, J.: Adaptive ABAC Policy Learning: A Reinforcement Learning Approach. CoRR, 2021, doi: 10.48550/arXiv.2105.08587.

[38] Mocanu, D. C.—Turkmen, F.—Liotta, A.: Towards ABAC Policy Mining from Logs with Deep Learning. Proceedings of the 18th International Multiconference - Intelligent Systems (IS 2015), 2015.

[39] Cappelletti, L.—Valtolina, S.—Valentini, G.—Mesiti, M.—Bertino, E.: On the Quality of Classification Models for Inferring ABAC Policies from Access Logs. 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 4000–4007, doi: 10.1109/BigData47090.2019.9005959.

[40] Prokhorenkova, L.—Gusev, G.—Vorobev, A.—Dorogush, A. V.—Gulin, A.: CatBoost: Unbiased Boosting with Categorical Features. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 31 (NeurIPS 2018). Curran Associates, Inc., 2018, pp. 6639–6649.

[41] Dorogush, A. V.—Ershov, V.—Gulin, A.: CatBoost: Gradient Boosting with Categorical Features Support. Corr, 2018, doi: 10.48550/arXiv.1810.11363.

[42] Hancock, J. T.—Khoshgoftaar, T. M.: CatBoost for Big Data: An Interdisciplinary Review. Journal of Big Data, Vol. 7, 2020, Art. No. 94, doi: 10.1186/s40537-020-00369-8.

[43] Tan, P.—Steinbach, M. S.—Karpatne, A.—Kumar, V.: Introduction to Data Mining (second Edition). Pearson, 2019, `https://www-users.cse.umn.edu/%7Ekumar001/dmbook/index.php`.

Shan Quan is currently a graduate student in the College of Mathematics and System Science, Xinjiang University, China. His research mainly focuses on statistics and information security.

Yongdan Zhao is currently a graduate student in the College of Mathematics and System Science, Xinjiang University, China. Her research mainly focuses on statistics and information security.

Nurmamat Helil received his B.Sc., M.Sc. and Ph.D. degrees in the School of Mathematical Sciences, Peking University in 2000, 2003 and 2008, respectively. He is Full Professor of the College of Mathematics and System Science, Xinjiang University, China. From April 2010 to April 2011, he worked as a post-doctor in the School of Computer Science and Engineering, Chung-Ang University, Korea. From April 2016 to April 2017, he worked as Visiting Research Scholar in the Department of Computer Science and Engineering, University of Minnesota Twin Cities, USA. His research interests include information system security, access control, and cloud storage security.