Hindawi
Security and Communication Networks
Volume 2021, Article ID 3970485, 11 pages
https://doi.org/10.1155/2021/3970485

Research Article

Efficient Access Control Permission Decision Engine Based on Machine Learning

Aodi Liu,[1,2] **Xuehui Du**,[1,2] **and Na Wang**[1,2]

[1]*Information Engineering University, Zhengzhou 450000, China*
[2]*He'nan Province Key Laboratory of Information Security, Zhengzhou 450000, China*

Correspondence should be addressed to Xuehui Du; dxh37139@163.com

Received 15 January 2020; Revised 20 September 2020; Accepted 8 February 2021; Published 17 February 2021

Academic Editor: Petros Nicopolitidis

Access control technology is critical to the safe and reliable operation of information systems. However, owing to the massive policy scale and number of access control entities in open distributed information systems, such as big data, the Internet of Things, and cloud computing, existing access control permission decision methods suffer from a performance bottleneck. Consequently, the large access control time overhead affects the normal operation of business services. To overcome the above-mentioned problem, this paper proposes an efficient permission decision engine scheme based on machine learning (EPDE-ML). The proposed scheme converts the attribute-based access control request into a permission decision vector, and the access control permission decision problem is transformed into a binary classification problem that allows or denies access. The random forest algorithm is used to construct a vector decision classifier in order to establish an efficient permission decision engine. Experimental results show that the proposed method can achieve a permission decision accuracy of around 92.6% on a test dataset, and its permission decision efficiency is significantly higher than that of the benchmark method. In addition, its performance improvement becomes more obvious as the scale of policy increases.

1. Introduction

The continuous development of big data [1], the Internet of Things [2], cloud computing [3], and other new information technologies provides numerous advantages in various aspects of daily life. However, it also poses many new challenges to data protection owing to the frequent occurrence of various security breaches. For example, in May 2019, the personal data of more than 49 million Instagram users was leaked; that is, unauthorized users could access personal data stored in the AWS cloud database. Similarly, in January 2019, owing to improper allocation of shared resource permissions, hundreds of thousands of internal files of more than 90 companies were leaked. Furthermore, in March 2018, Facebook suffered a data breach. Cambridge Analytica illegally accessed the personal data of more than 50 million Facebook users without their authorization and used the data to build mathematical models for analyzing citizens' political preferences. Thus, effective protection of data resources is a fundamental requirement for data application and sharing.

As one of the core technologies for ensuring data security, access control technology [4, 5] can prevent unauthorized use of data resources and effectively protect data resources by managing users' permissions. However, the continuous development of new distributed and open computing paradigms, such as big data and the Internet of Things, has contributed to a steady increase in the number of entities and scale of policy in existing access control systems, thereby reducing their operating efficiency. The existing access control permission decision engine is essentially a multivalue logic system with equivalent operators in different fields. Its performance is negatively correlated with the number of entities and scale of policy. Thus, an increase in the number of entities and scale of policy will result in a performance bottleneck for the access control system and affect the normal operation of the core business system. In addition, the existing access control permission decision

mechanism can directly obtain the user's access control policy information, which entails the risk of private policy information disclosure. Hence, there is a need for distributed access control deployment, especially in the open computing environment, and the permission decision engine should be deployed at multiple node locations, as all nodes are at a risk of being attacked by hackers.

To solve the above-mentioned problems, this study improves the policy decision point (PDP) of the attribute-based access control (ABAC) model [6–8]. Specifically, an efficient permission decision engine scheme based on machine learning (EPDE-ML) is proposed. It transforms the permission decision problem into a binary classification problem in which access requests are allowed or forbidden. The random forest algorithm is used to predict the decision result, thus providing effective support not only for the efficient implementation of access control but also for the distributed deployment of the decision engine through the compose permission decision structure. Experimental results show that the proposed method can achieve a permission decision accuracy of 92.6%. As the scale of policy increases, the time cost of EPDE-ML method remains stable at around 0.115 s. Thus, the EPDE-ML method outperforms the traditional permission decision method. Moreover, the EPDE-ML engine can make permission decisions without exposing users' private policy information, thereby reducing the security risk of the system and safely realizing efficient decision-making regarding users' access permissions under the mass policy environment.

The remainder of this paper is organized as follows. Section 2 reviews related studies on access control permission decision technology. Section 3 formalizes the related concepts and explains the implementation framework and processing flow in detail. Section 4 describes the proposed permission decision algorithm, which is based the on the random forest algorithm. Section 5 evaluates the effectiveness and feasibility of the EPDE-ML method on the basis of simulation experiments and analyzes the experimental results. Finally, Section 6 concludes the paper.

2. Related Work

Higher access control execution efficiency has long been a major research objective in the field of computer security. Researchers have modified existing algorithms or proposed new solutions to achieve better performance.

One approach is to improve the access control permission decision performance by optimizing the access control policy set. Wang et al. [9] proposed a multilevel optimization-based evaluation engine (MLOBEE). The policy set is optimized prior to the start of the permission decision, which reduces the size of the policy and adjusts the execution order of the policy. Further, multilevel cache technology is used to reduce the communication loss of policy matching. Deng and Zhang [10, 11] proposed a method for improving the PDP decision performance by eliminating policy conflicts and decomposing access control policy sets; however, the improvement in the permission decision performance was limited. Liu et al. [12] converted the access control policy with a hierarchical structure and multiple complex conflict resolution mechanisms into an equivalent policy with a flat structure and a single conflict resolution mechanism to improve the permission decision performance. The access control policy set has a variable number of access control policies, and the policies have a variable number of rules. Marouf et al. [13] believed that the order of the policies in the policy set has a significant impact on the permission decision efficiency. They proposed an adaptive policy optimization method to perform k-means clustering on the access control policy set. The decision performance was optimized by reordering the policies. To provide efficient access decisions for Web services, Mourad and Jebbaou [14] proposed a semantics-based real-time policy evaluation algorithm that evaluates rules at the rule, policy, and policy set levels.

Another approach is to improve the permission decision performance through distributed permission decision processing. Deng et al. [15] designed a distributed permission decision model (XPDP) to overcome the limitation of single PDP computing performance and further improved the decision efficiency by clustering policies on the basis of subjects' attributes and reordering policy sets on the basis of similarity. Kateb et al. [16] proposed an automated method that reconstructs a single global policy as a policy with fewer rules in order to disperse a single system PDP into multiple PDPs that can work together. Such policy reconfiguration can improve the distributed permission decision performance and reduce the time required to evaluate the access request. However, distributed parallel decision technology is a resource-intensive method owing to its complexity. Distributed PDP also involves additional synchronous communication overhead, and multiple sub-PDPs increase the possibility of private policy information exposure. In addition, some researchers have improved the permission decision performance by enhancing the policy query efficiency. Ros and Lischika [17] proposed a permission decision optimization method based on two tree structures: match tree and combination tree. The match tree uses a binary search algorithm to rapidly search for the policy matching the access request, and the combination tree evaluates the access request on the basis of the matching policy. To overcome the loss of the original policy semantics in the permission decision process, Ngo et al. [18] proposed a decision graph method based on data interval partition aggregation, which can parse and convert the complex logical expression in the policy into a decision tree structure, thereby improving the evaluation performance of the policy effectively.

In addition, some studies have focused on specific application scenarios. To address the access decision problem in social networks, Morovat and Panda [19] designed a new permission decision method that includes a transformation engine module and a request engine module. The transformation engine module first parses the access request and access control policies into standard formats using natural language processing technology. Then, the request engine module deduces the final decision result. With regard to the stateful ABAC policy, Bui et al. [20] proposed a fast access

control algorithm with distributed evaluation (FACADE). The algorithm uses a special concurrent control scheme for multiversion timestamp sorting to handle potentially conflicting status updates. It reduces the time cost of high-throughput access by minimizing the length of the message chain on the critical path.

Some studies have also focused on the privacy protection of policy information through the introduction of cryptography in the process of access control. Martiny et al. [21] specified policy objects based on ontology and made access control permission decisions using various privacy enhancement technologies. Harbach et al. [22] used homomorphic cryptography to mask the access control mechanism in order to implement hidden policies, hidden credentials, and hidden permission decisions. In addition, Kan et al. [23] proposed a bloom-filter-based matching method for policies and attributes, and they implemented privacy protection of users' policy information through ciphertext policy attribute-based encryption (CP-ABE).

In summary, existing solutions for alleviating permission decision performance issues mainly focus on the following schemes: adjusting the policy set, decomposing the policy set, and distributing parallel permission decision, which is essentially the optimization of the permission decision method on the basis of the logical operation of policy matching. This study innovatively solves the related problems by training an access control permission decision engine on the basis of machine learning using the current policy set.

3. Related Concepts and Implementation Framework

3.1. Related Concepts. Definition 1 is as follows: attribute is used to describe the characteristic information of entities participating in the access control process. It is composed of attribute name and attribute value. It includes four types of attributes, which can be expressed as quaternions (S, R, O, E).

S represents the subject attribute, which is used to describe the attribute information possessed by the initiator of the access request (role, gender, etc.). R represents the resource attribute, which is used to describe the attribute information of the resource that can be accessed (name, security level, etc.). O represents the operation attribute, which is used to describe various operation behaviors of the subject on the resource (read, write, etc.). E represents the environment attribute, which is used to describe the environment constraint information when access control occurs (time, place, etc.).

Definition 2 is as follows: attribute tuple is a set of specific class attributes that characterize access control entities. It is the embodiment of the dynamic assignment relationship of attributes, which can be expressed as X-tuple = $\{a1, a2, \ldots an\}$, X $\in \{S, R, O, E\}$.

Definition 3 is as follows: access control policy comprises the rules governing subjects' access to resources. It is a concrete embodiment of the authorization behavior of the subject with respect to the resource, which can be expressed as ACP = (S-tuple, R-tuple, O-tuple, E-tuple, Sign). Sign$\in$ {permit, deny} indicates access that is allowed or denied.

Definition 4 is as follows: access request is a description of the visitor of the resource, the accessed resource, and the requested operation. It can be expressed as AR= (S-tuple, R-tuple, O-tuple). Access request contains at least one subject attribute, one resource attribute, and one action attribute.

Definition 5 is as follows: permission decision is a decision response that allows or denies users access to the corresponding resources in the given access control policy evaluation environment, which can be expressed as a mapping function: Decision: AR$\longrightarrow${permit, deny}.

The access control permission decision engine based on machine learning finds a function Decision () through machine learning and maps the user's access request to binary decision results {permit, deny}. It transforms the permission decision problem into a binary classification problem in order to determine whether the entity attributes meet the constraints of the access control policy.

3.2. Implementation Framework. The traditional access control permission decision method is processed by traversing the access control policy set that matches the access request and performing logical operations on the access request, as shown in Figure 1. The process is as follows.

(1) The user sends an access request for the target resource to the permission decision engine.

(2) After receiving the access request from the user, the permission decision engine performs attribute resolution on the request. It matches the policy information associated with the entity attribute information of the request in the policy administration point (PAP).

(3) PAP queries the relevant policy set from among all the policy sets and sends the matched policy set back to the decision engine in the form of a match response.

(4) The decision engine makes the permission decision based on the match response. If the result is permit, the user can directly access the target resource. If the result is deny, the user is denied access to the target resource.

However, with a considerable increase in access control policies and the simultaneous arrival of a large number of access control requests from users, the traditional permission decision method suffers from a performance bottleneck. The performance bottleneck mainly occurs in two links. First, in the correlation policy matching, it is necessary to retrieve the policy information related to the access request from the massive policy information. As the scale of policy increases, so does the time overhead. Second, in the process of permission decision, the traditional method makes permission decisions based on the logical implication relationship between the request and the policy. The number of policies related to the request is not

unique; there is a one-to-many relationship. With an increase in the number of request-related policies, the time overhead will also increase. In addition, policy administration and permission decision are tightly coupled. The policy information of users is exposed in the decision engine, which brings additional security risks to access control services. To solve the above-mentioned problems, this study proposes a novel efficient permission decision engine scheme based on machine learning (EPDE-ML). Its process is shown in Figure 2.

The online decision engine in EPDE-ML is composed of the offline permission decision model trained by the current access control policy information. The engine does not interact with the real access control policy during the permission decision. Policy administration and permission decision are relatively independent in order to achieve privacy protection of the policy information. Meanwhile, there is no need to query related access control policies in the process of the permission decision, which can be deployed and run independently of the policies. Thus, it is an efficient, secure, and lightweight access control scheme.

3.3. Compose Permission Decision Structure. For the complex access control business process in distributed and open environments, EPDE-ML supports a variety of flexible deployment methods and forms a composite permission decision structure, which can further improve the access control permission decision performance.

3.3.1. Cascade Architecture. Each subpermission decision engine within the composite permission decision structure is called and executed in sequential order. The structure is shown in Figure 3(a). This structure can be used in scenarios where cross-domain data resource access is required between different organizations. Different organizations train their decision engines for policy information in their respective security domains. Users can access the appropriate cross-domain resources only if all the intradomain engines (DE) decide to allow it.

3.3.2. Concurrent Architecture. Each subauthority decision engine within the composite permission decision structure can be executed in parallel and synchronously, and there is no dependency between them. The structure is shown in Figure 3(b). Each subpermission decision engine is the same, and this parallel structure can improve the permission decision efficiency by diverting massive concurrent access requests. Moreover, because of the existence of multiple redundant decision engines, a single point of failure can be avoided and the reliability of the system can be improved.

3.3.3. Condition Architecture. The corresponding subpermission decision engine is executed according to the conditional constraints of the composite permission decision structure, which is shown in Figure 3(c). Different subpermission decision engines can be flexibly selected according to different business interaction conditions. The subpermission decision engines are independent of each other, which improves the flexible execution ability of the decision structure.

4. Permission Decision Algorithm Based on Random Forest

4.1. Core Idea of Algorithm. The overall structure of the access control permission decision engine model based on the random forest algorithm is shown in Figure 4. The structure includes the feature extraction and processing module, model training module, model testing module, and permission decision engine module. The feature extraction and processing stage transforms access control policies from the training and test datasets into access control policy evaluation vectors in the form of one-hot attributes through the process of policy data balance, feature extraction, and attribute feature dimension reduction. After model training and model testing, the final permission decision engine is obtained and used to make decision responses to access requests.

4.2. Policy Data Balancing. The real access control policy set is an unbalanced dataset. There can be a significant difference between the number of allowed policies and the number of denied policies in a policy set. For example, the ratio of the number of allowed policies to the number of denied policies is around 16 : 1 in the real access control policy set [24] published by Amazon. Such unbalanced datasets will degrade the model performance. Therefore, we adopt the adaptive synthetic sampling approach (ADASYN) [25] to generate balanced datasets. The calculation method is as follows.

(1) Calculate the unbalance degree, where M_s is the number of minority class samples, and M_l is the number of majority class samples:

$$d = \frac{M_s}{M_l}, \quad d \in (0, 1]. \tag{1}$$

(2) Calculate the data that needs to be synthesized. When $\alpha = 1$, GN is equal to the difference between the minority classes and the majority classes. At this time, the data of the majority classes and minority classes are exactly balanced in the synthesized dataset:

$$GN = \alpha \cdot (M_l - M_s), \quad \alpha \in [0, 1]. \tag{2}$$

(3) The Euclidean distance is used to calculate f neighbors of each minority class sample, and N_l is the number of majority class samples among the f neighbors:

$$r_i = \frac{N_l}{f}, \quad r_i \in [0, 1]. \tag{3}$$

(4) Calculate the case of the majority class around the minority class sample:

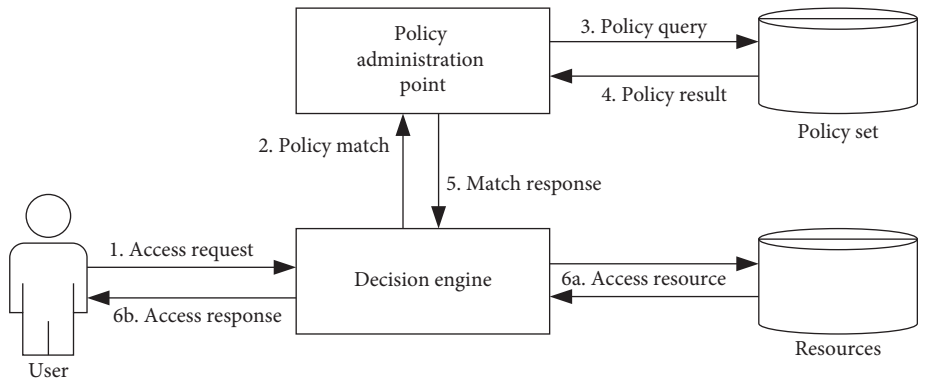

Figure 1: Traditional permission decision method.

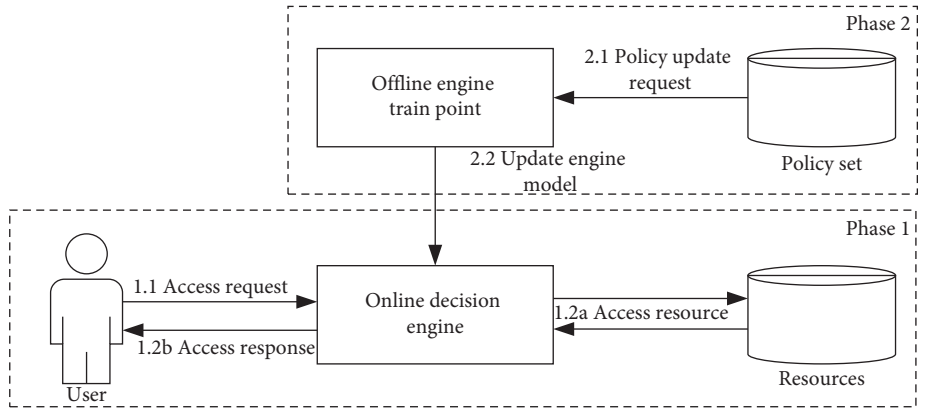

Figure 2: Permission decision method of EPDE-ML.

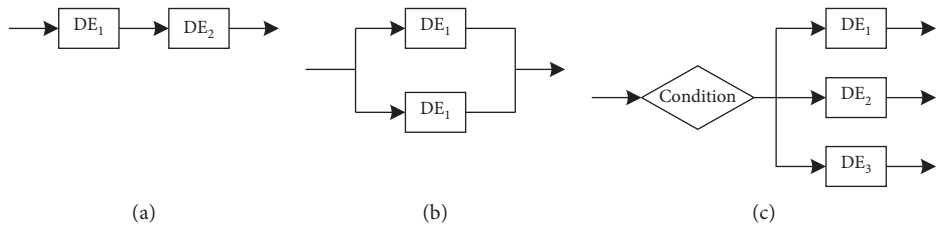

Figure 3: Composite permission decision structure. (a) Cascade architecture. (b) Concurrent architecture. (c) Condition architecture.

$$\widehat{r}_i = \frac{r_i}{\sum_{i=1}^{m_s} r_i}, \quad \sum_{i=1}^{m_s} r_i = 1. \tag{4}$$

(5) Calculate the number of samples to be synthesized for each minority class:

$$gn_i = \widehat{r}_i \cdot GN. \tag{5}$$

(6) Choose one minority class sample $x_{zi}$from among k neighbors around each minority class sample $x_i$to be synthesized for sample data synthesis:

$$s_i = x_i + \beta \cdot (x_{zi} - x_i), \quad \beta \in [0, 1]. \tag{6}$$

4.3. Attribute Feature Dimension Reduction. A chi-squared test [26] is used to reduce the dimension of the attribute features. It is a hypothesis test method based on χ^2 distribution, which is commonly used to compare the relationship between the observed data and the data that we expect to get according to the hypothesis. It can be used to score and sort the features and choose the features that rank highly to achieve feature dimension reduction. Further, by choosing the features with good decision effects, efficient training and classification can be realized:

$$X^2(t, c) = \sum_{e_t \in 0,1} \sum_{e_c \in 0,1} \frac{\left(N_{e_t e_c} - E_{e_t e_c}\right)^2}{E_{e_t e_c}}, \tag{7}$$

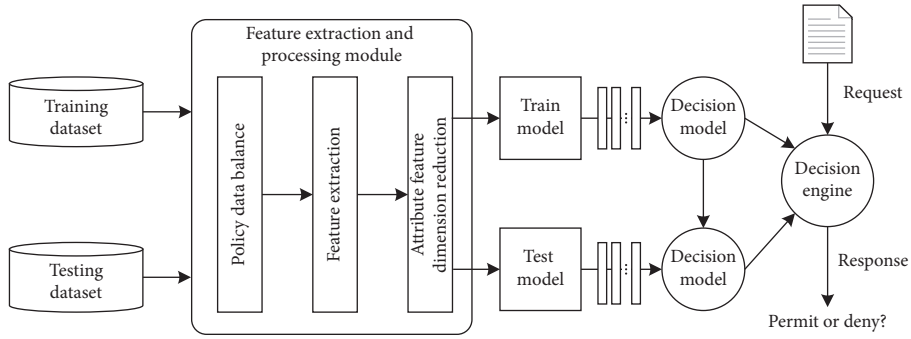

Figure 4: Permission decision engine model based on random forest.

where t represents the presence or absence of relevant features, c represents the permission decision result (permit is 1, deny is 0), N represents the actual observed value, and E represents the expected value. For example, E_{10} represents the occurrence of the corresponding feature t and permission decision result is $c = 0$.

4.4. Model Training. The random forest (RF) is a well-known ensemble learning method that can be used to build prediction models to solve classification and regression problems. Ensemble learning methods train multiple learning models to obtain better prediction results. The random forest creates a complete forest consisting of several randomly unrelated classification and regression trees (CART) to obtain the best possible predicted results. The ensemble training process of the access control permission decision engine model based on RF is shown in Figure 5.

(1) Repeated sampling with replacement is conducted on the basis of the bootstrap method from the training complete set Sample. The training complete set is divided into k training subsets {Subset_1, Subset_2, . . .Subset_{k-1}, Subset_k}. The number of samples in both the complete set and the subsets is N.

(2) A CART decision tree is constructed for each training subset Subset_i. Specifically, m features are randomly selected from among all the attribute features. The node then selects an optimal feature from the extracted m features and applies it to the node for the splitting operation. The newly generated tree node is further split on the basis of the GINI value in the remaining m-1 features until the leaves of the tree can no longer be split.

(3) k CART decision trees are obtained, corresponding to k training subsets. For the input variable, each CART decision tree will output one decision result DecisionResult (T_i). If the permission decision is access-allowed, the DecisionResult (Ti) value of the tree is 1. If the permission decision is access-denied, the DecisionResult (T_i) value of the tree is 0. Then, the aggregation vote yields the final decision results of all the decision trees:

$$\text{Vote}(x) = \sum_{i=1}^{k} \text{DecisionResult}(T_i). \tag{8}$$

(4) For the user attribute information of the input, the final permission decision formula can be obtained as

$$\text{Permission}(\text{request}) = \begin{cases} 1, & \left(\frac{\text{Vote}(x)}{k}\right) > 0.5, \\ 0, & \left(\frac{\text{Vote}(x)}{k}\right) \le 0.5. \end{cases} \tag{9}$$

If $Permission(request) = 1$, the user is allowed to access the corresponding resources. Otherwise, access is denied.

The CART decision tree makes the data purer by splitting the nodes, and its output will be closer to the real value. For the classification problem, the GINI value is used to evaluate the purity of the nodes in the tree. The GINI value is calculated as follows:

$$\text{GINI} = 1 - \sum_{i \in I} p_i^2, \text{Gain} = \sum_{j \in J} p_j \cdot GINI_j. \tag{10}$$

The larger the GINI value, the worse the effect of the splitting mode. Therefore, the classification tree can be minimized by selecting the attribute with the smallest GINI value of the child node as the basis for splitting. In addition, to reduce overfitting, the cost-complexity pruning (CCP) method is used to reduce the complexity of the decision tree. CCP removes the left and right child nodes of nonleaf nodes with the minimum surface error gain value. If multiple nonleaf nodes have the same minimum surface error gain value, the nonleaf node with the largest number of nonleaf nodes is selected for pruning. The surface error gain value can be calculated as follows:

$$R(T) = \sum_{i}^{m} r_i(t) \cdot p_i(t),$$
$$\alpha = \frac{R(t) - R(T)}{N(T) - 1}, \tag{11}$$

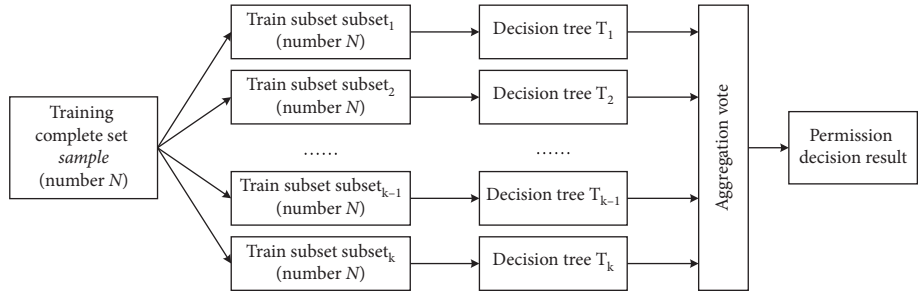

Figure 5: Training process of permission decision engine.

where $R(t)$ is the error cost of the leaf node, $R(t) = r(t) \cdot p(t)$, $r(t)$ is the error rate of the node, $p(t)$ is the data node ratio, $R(T)$ is the error cost of the subtree, $r_i(t)$ is the error rate of the child node, $p_i(t)$ is the data node ratio of node i, and $N(T)$ is the number of nodes in the subtree.

5. Experiment and Analysis

5.1. Datasets and Experimental Environments. To verify the effectiveness of the proposed method, we used Amazon's real access control policy set [24] to conduct experiments. The dataset contains more than 32,000 pieces of real access control policy information that include 10 different categories of user attribute information. These 10 attribute categories cover more than 8000 types of attributes. We conducted data balance processing on the experimental dataset and randomly divided the data into the training dataset (80% of the policy data) and test dataset (20% of the policy data). In addition, for effective comparison with the traditional permission decision method, we built access control policy sets with policy sizes of 1000, 2000, 3000, 4000, 5000, 6000, 7000, and 8000 and used them in performance comparison tests. This dataset is extracted from Amazon's real access control policy set. Each access request is sent five times, and the permission decision time is obtained by calculating the average response time of all requests. The hardware and software specifications for the experiment were as follows: operating system; Win10 64-bit; CPU, Intel® Core™ i7-8750H@2.21 GHz; GPU, GeForce GTX 1050 Ti max-q; memory size, 16 GB; and software platform, Python 3.6.

5.2. Evaluation Index. We used the following performance indexes to evaluate the permission decision effect of access control. The confusion matrix of the permission decision results is defined in Table 1.

In Table 1, $D_{PP'}$ represents the number of samples correctly permitted access, $D_{PD'}$ represents the number of samples wrongly denied access, $D_{DP'}$ represents the number of samples wrongly permitted access, and $D_{DD'}$ represents the number of samples correctly denied access. The corresponding evaluation indexes are calculated as follows.

(1) Accuracy represents the ratio of the number of correctly predicted samples to the total number of samples. The formula is as follows:

$$\text{Acc} = \frac{D_{PP'} + D_{DD'}}{D_{PP'} + D_{PD'} + D_{DP'} + D_{DD'}}. \quad (12)$$

(2) Precision represents the ratio of the number of correctly permitted samples to the number of predicted permitted samples. The formula is as follows:

$$\text{Pre} = \frac{D_{PP'}}{D_{PP'} + D_{DP'}}. \quad (13)$$

(3) Recall represents the ratio of the number of correctly predicted samples to the number of real permitted samples, which is a measure of coverage. The formula is as follows:

$$\text{Re} = \frac{D_{PP'}}{D_{PP'} + D_{PD'}}. \quad (14)$$

(4) F1 is the weighted harmonic average of precision and recall. The formula is as follows:

$$F1 = \frac{2^* \text{Pre}^* \text{Re}}{\text{Pre} + \text{Re}}. \quad (15)$$

5.3. Experimental Results and Analysis. To evaluate the performance of the proposed method, we designed the following four experiments: comparison of receiver operating characteristic (ROC) curve and area under the curve (AUC) of different methods before and after data balancing, comparison of performance indicators of different machine learning methods, and comparison of permission decision times of different methods.

(1) In the comparison of ROC curve and AUC value of different methods before and after data balancing, the ROC curve can indicate the proportional relationship between the number of samples of the decision results correctly predicted by the model and the number of samples wrongly predicted by the model. Moreover, the ROC curve does not change with the distribution of the sample set; hence, it is an important evaluation index of machine learning. According to the experimental results in Figures 6 and 7, the performance of each algorithm is generally poor before data balancing; the LR and SVM algorithms are unable to make accurate and effective decision responses. However, the performance of

Table 1: Confusion matrix of permission decision results.

Real results	Predicted results	
	Permitted access	Denied access
Permitted access	$D_{PP'}$	$D_{PD'}$
Denied access	$D_{DP'}$	$D_{DD'}$

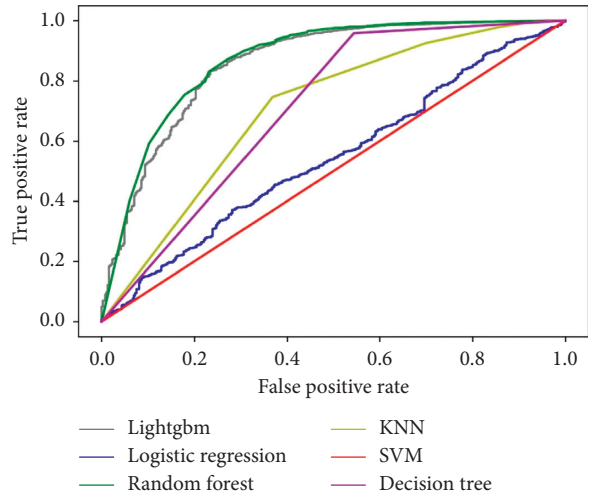

Figure 6: ROC curve of each algorithm before data balancing.

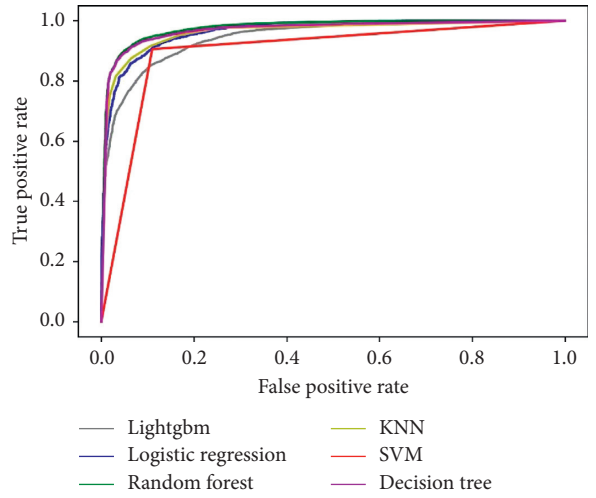

Figure 7: ROC curve of each algorithm after data balancing.

each algorithm is generally improved after data balancing. The proposed permission decision algorithm based on the random forest algorithm achieves an optimal AUC value of 0.975 (as shown in Table 2).

(2) In the comparison of performance indicators of different machine learning methods, the same attribute features are selected, and the Accuracy, Precision, Recall, and *F*1 values of different permission decision methods are compared. The experimental results are shown in Figure 8. Compared with Lightgbm, LR, KNN, SVM, and DT, the proposed method shows better performance in terms of comprehensive permission decisions. In addition, the time required for training and updating different methods has an important influence on the dynamic and timely updating of the system access control policy. Therefore, we also tested the model training and update time required by the permission decision engine on the basis of different methods. As shown in Figure 9, the Lightgbm and KNN models have the longest training times, while the other methods have similar training times. Table 3 is the performance comparison of different methods.

(3) In the comparison of permission decision times of different methods, as shown in Figures 10 and 11, the traditional permission decision method based on

Table 2: AUC values for different methods.

Methods	Lightgbm	LR	RF	KNN	SVM	DT
AUC value before data balancing	0.851	0.526	0.852	0.691	0.500	0.692
AUC value after data balancing	0.945	0.965	**0.975**	0.960	0.899	0.969

The bold value is the experimental result of our method.

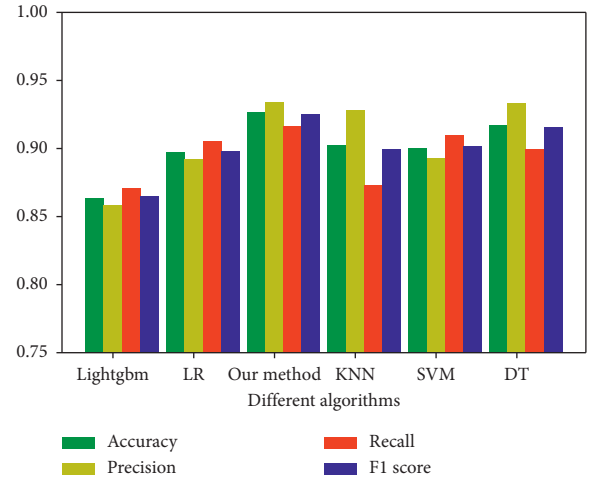

Figure 8: Comparison of performance indicators of different methods.

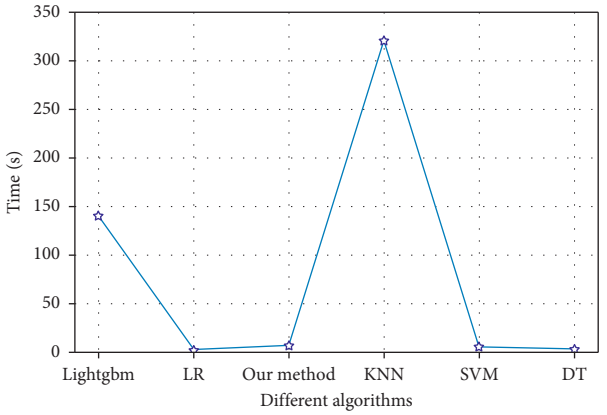

Figure 9: Comparison of training times of different methods.

Table 3: Performance comparison of different methods.

Methods	Lightgbm	LR	RF	KNN	SVM	DT
Accuracy	0.863	0.897	**0.926**	0.902	0.899	0.917
Precision	0.858	0.892	**0.934**	0.928	0.892	0.933
Recall	0.871	0.905	**0.916**	0.872	0.909	0.899
F-measure	0.864	0.898	**0.925**	0.899	0.911	0.916

The bold values are the experimental result of our method.

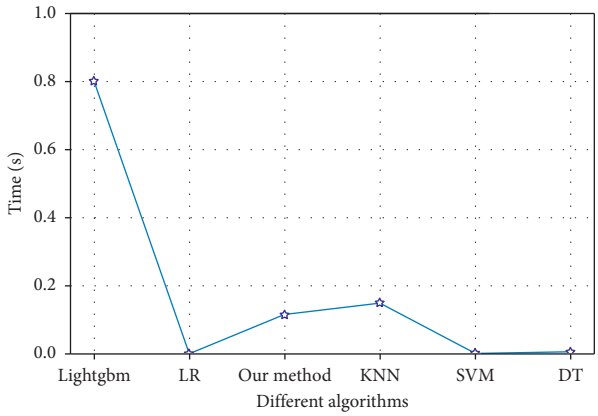

Figure 10: Comparison of permission decision times.

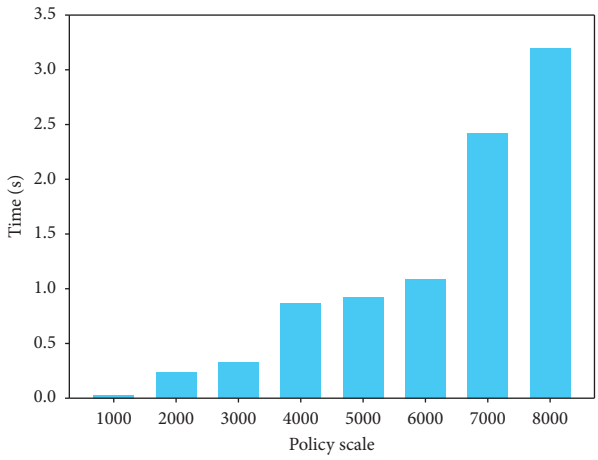

Figure 11: Permission decision time of traditional method under different policy scales.

logical operation has a positive correlation between the decision time and the scale of policy. With an increase in the scale of policy, the permission decision time will increase significantly. However, regardless of the scale of policy, as long as the attribute category remains unchanged, the permission decision time tends to be stable in the machine-learning-based method. Compared with other methods, the proposed method has a stable decision time of around 0.115 s. Its overall performance is better, and it shows better adaptability to the access control requirements of massive real-time permission decisions.

6. Conclusion

This study compared and analyzed existing technical schemes for access control permission decision. To overcome the performance issues of existing methods, an efficient permission decision engine scheme based on machine learning (EPDE-ML) was proposed. We transformed the problem of permission decision into a binary classification problem of machine learning so that the access control system operation is not affected by the scale of policy or number of entities. Thus, the permission decision efficiency was effectively improved. In addition, EPDE-ML only needs to make a decision response according to the access request information in the permission decision process. It does not require communication or interaction with the access control policy set; thus, it can realize privacy protection of sensitive policy information. Moreover, the EPDE-ML engine can support the deployment of distributed composite permission decisions in open distributed environments. The experimental results showed that the EPDE-ML scheme exhibits good permission decision performance and it can meet the real-time requirements of highly concurrent access control requests. In the future, we will further investigate the applicability of EPDE-ML under the condition of access control scenario migration. In addition, we will investigate how to optimize the representation method of attribute-based access request vectorization to improve the model as well as the decision performance.

Data Availability

The data supporting this article are from previously reported studies and datasets, which have been cited.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (nos. 2018YFB0803603 and 2016YFB0501901) and National Natural Science Foundation of China (no. 61802436).

References

[1] H. Li, M. Zhang, D. G. Feng, and Z. Hui, "Research on access control of big data," *Chinese Journal of Computers*, vol. 40, no. 1, pp. 72–91, 2017.

[2] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the Internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 99, pp. 1594–1605, 2018.

[3] T. Wei, Z. Geng, X. Yang, and D. Wang, "Attribute-based access control with constant-size ciphertext in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 617–627, 2017.

[4] D. Servos and S. L. Osborn, "Current research and open problems in attribute-based access control," *ACM Computing Surveys*, vol. 49, no. 4, pp. 1–45, 2017.

[5] F. Paci, A. Squicciarini, and N. Zannone, "Survey on access control for community-centered collaborative systems," *ACM Computing Surveys*, vol. 51, no. 1, pp. 1–38, 2018.

[6] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, 2015.

[7] L. Fang, L. H. Yin, Y. C. Guo, and B. X. Fang, "A survey of Key technologies in attribute-based access control scheme," *Chinese Journal of Computers*, vol. 40, no. 07, pp. 1680–1698, 2017.

[8] J. Xin, R. Krishnan, and R. Sandhu, "A unified attribute-based access control model covering DAC, MAC and RBAC," in *Proceedings of the 26th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy*, pp. 41–55, Paris, France, July 2012.

[9] Y.-Z. Wang, D.-G. Feng, L.-W. Zhang, and M. Zhang, "XACML policy evaluation engine based on multi-level optimization technology," *Journal of Software*, vol. 22, no. 2, pp. 323–328, 2011.

[10] F. Deng and L.-Y. Zhang, "Elimination of policy conflict to improve the PDP evaluation performance," *Journal of Network and Computer Applications*, vol. 80, pp. 45–57, 2017.

[11] F. Deng, P. Chen, L.-Y. Zhang, X.-Q. Wang, S.-D. Li, and H. Xu, "Policy decomposition for evaluation performance improvement of PDP," *Mathematical Problems in Engineering*, vol. 2014, no. 2, 14 pages, Article ID 610278, 2014.

[12] A. X. Liu, F. Chen, J. Hwang, and T. Xie, "Designing fast and scalable XACML policy evaluation engines," *IEEE Transactions on Computers*, vol. 60, no. 12, pp. 1802–1817, 2011.

[13] S. Marouf, M. Shehab, A. Squicciarini, and S. Sundareswaran, "Adaptive reordering and clustering-based framework for efficient XACML policy evaluation," *IEEE Transactions on Services Computing*, vol. 4, no. 4, pp. 300–313, 2011.

[14] A. Mourad and H. Jebbaoui, "SBA-XACML: set-based approach providing efficient policy decision process for accessing Web services," *Expert Systems with Applications*, vol. 42, no. 1, pp. 165–178, 2015.

[15] F. Deng, J. Lu, S. Y. Wang, J. Pan, and L. Y. Zhang, "A distributed PDP model based on spectral clustering for improving evaluation performance," *World Wide Web-internet and Web Information Systems*, vol. 22, no. 4, pp. 1–22, 2018.

[16] D. E. Kateb, T. Mouelhi, and Y. L. Traon, "Refactoring access control policies for performance improvement," in *Proceedings of the 3rd Joint WOSP/SIPEW International Conference on Performance Engineering*, pp. 323–334, Boston, MA, USA, April 2012.

[17] S. P. Ros and M. Lischka, "Graph-based XACML evaluation," in *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, pp. 83–92, Newark, NJ, USA, June 2012.

[18] C. Ngo, Y. Demchenko, and C. de Laat, "Decision diagrams for XACML policy evaluation and management," *Computers and Security*, vol. 49, no. 5, pp. 1–16, 2015.

[19] K. Morovat and B. Panda, "Request evaluation for policy-based attribute access control in social network cloud," in *Proceedings of the IEEE International Conference on Computer and Information Technology*, pp. 399–406, Fiji, Oceania, December 2016.

[20] T. Bui, S. D. Stoller, and S. Sharma, "Fast distributed evaluation of stateful attribute-based access control policies," in *Proceedings of the Ifip Conference on Data and Applications Security and Privacy*, pp. 101–119, Philadelphia, PA, USA, July 2017.

[21] K. Martiny, D. Elenius, and G. Denker, "Protecting privacy with a declarative policy framework," in *Proceedings of the IEEE International Conference on Semantic Computing*, pp. 227–234, Laguna Hills, CA, USA, February 2018.

[22] M. Harbach, S. Fahl, M. Brenner, T. Muders, and M. Smith, "Towards privacy-preserving access control with hidden policies, hidden credentials and hidden decisions," in *Proceedings of the Tenth International Conference on Privacy*, pp. 17–24, Paris, France, July 2012.

[23] Y. Kan, H. Qi, L. Hui, Z. Kan, S. Zhou, and X. Shen, "An efficient and fine-grained big data access control scheme with privacy-preserving policy," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 563–571, 2017.

[24] Access control dataset. 2019, https://www.kaggle.com/c/amazon-employee-access-challenge/data.

[25] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: adaptive synthetic sampling approach for imbalanced learning," in *Proceedings of the International Joint Conference on Neural Network*, pp. 1322–1328, Hong Kong, China, July 2008.

[26] A. Kumar, S. Roy, and P. Ranjan, "Dimensionality reduction for high dimensional data," in *Proceedings of the International Conference on Information and Communication Technology for Competitive Strategies*, pp. 1–6, Udaipur, Rajasthan, India, November 2014.