scientific reports

Check for updates

OPEN

An efficient density peak cluster algorithm for improving policy evaluation performance

Zhenhua Yu[1], Yanghao Yan[1], Fan Deng[1]✉, Fei Zhang[2] & Zhiwu Li[3]✉

In recent years, the XACML (eXtensible Access Control Markup Language) is widely used in a variety of research fields, especially in access control. However, when policy sets defined by the XACML become large and complex, the policy evaluation time increases significantly. In order to improve policy evaluation performance, we propose an optimization algorithm based on the DPCA (Density Peak Cluster Algorithm) to improve the clustering effect on large-scale complex policy sets. Combined with this algorithm, an efficient policy evaluation engine, named DPEngine, is proposed to speed up policy matching and reduce the policy evaluation time. We compare the policy evaluation time of DPEngine with the Sun PDP, HPEngine, XEngine and SBA-XACML. The experiment results show that (1) when the number of requests reaches 10,000, the DPEngine evaluation time on a large-scale policy set with 100,000 rules is approximately 2.23%, 3.47%, 3.67% and 4.06% of that of the Sun PDP, HPEngine, XEngine and SBA-XACML, respectively and (2) as the number of requests increases, the DPEngine evaluation time grows linearly. Compared with other policy evaluation engines, the DPEngine has the advantages of efficiency and stability.

At present, access control has become not only an important research object in fields of network and information security but also a research hot spot in interdisciplinary subjects of Internet of things[1], Industry 4.0[2], CPS (Cyber-Physical System)[3], Block-chain[4], Cloud computing[5] and big data[6]. With continuous growths of network and information system security requirements, access control has become a key to promoting the rapid development of critical industries.

In the research of access control, policies are used more and more frequently to describe the security requirements of network and information authorization service systems. The access control is an important part of the network security requirement module in authorization service systems. It means that when users access authorized network service systems, the systems control or protect the access to existing resources through the identity authentication, dynamic authorization and other methods. Recently, the XACML[7] (eXtensible Access Control Markup Language) is widely used to define architectures of access control mechanisms and express access control policies. First, in an XACML access control model[8], users send requests to a PEP (Policy Enforcement Point), which dispatches requests to a PDP (Policy Decision Point). Then, the PDP queries the request attributes from a PIP (Policy Information Point) and traverses policies in a PAP (Policy Administrator Point). Finally, the PDP makes evaluation decisions and returns them to the PEP as responses. Obviously, the PDP is one of the most important modules in an XACML access control model. The PDP evaluation performance is crucial for an authorization service system[10]. However, as interactions between users and servers increase, the size and complexity of policy sets increase correspondingly[11]. As a result, it becomes more difficult and time-consuming to evaluate a large number of requests.

In order to improve the PDP evaluation performance, there have been many relevant research achievements in recent years, including distributed authorization systems, decision graphs, eliminations of conflicts and redundancies, etc. However, the PDP evaluation performance is still restricted by the following multiple factors.

(1) Both distributed authorization systems and decision graphs need to spend much time evaluating requests for dealing with large-scale policy sets.
(2) The elimination of conflicts and redundancies is beneficial to optimizing policy sets, but the improvement of the PDP evaluation performance is restrained.

[1]Institute of Systems Security and Control, College of Computer Science and Technology, Xi'an University of Science and Technology, Xi'an 710054, China. [2]Hitachi Building Technology (Guangzhou) Co., Ltd, Guangzhou 510700, China. [3]Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau, China. ✉email: 122340695@qq.com; systemscontrol@gmail.com

(3) It is extremely difficult to construct decision graphs for large-scale policy sets. Although decision graphs consume a lot of storage space, the PDP evaluation performance is still limited.

Therefore, how to improve the PDP evaluation performance is a challenge for large-scale policy sets. Existing evaluation methods are inefficient in dealing with large-scale and complex policy sets. Using clustering algorithms to optimize policy sets is effective for improving policy evaluation performance. Many powerful and novel clustering algorithms are proposed to solve many different problems. Hassan et al.[12] propose a meta-heuristic clustering algorithm, which can process heterogeneous data sets with multiple features. In Ref.[13], an adaptive evolutionary clustering algorithm is developed to solve the Formal Context's ambiguity problem. Mohammed et al.[14] propose a meta-heuristic algorithm by simulating the reproductive behavior of bee colonies, which is applied to classic pressure vessel design problems and achieves good results. Askari[15] modifies the Fuzzy C-Means algorithm to be suitable for data with unequal cluster sizes, noise and outliers, and uneven distribution of quality. However, clustering algorithms have different adaptability to data sets of different shapes and sizes, and it is difficult to achieve excellent effects on large-scale and complex policy sets. In order to greatly improve the evaluation efficiency of PDP, it is urgent to propose a clustering algorithm that can effectively deal with large-scale policy sets. To solve these problems, we propose an improved clustering algorithm and construct an efficient policy evaluation engine. Our contributions are described as follows.

(1) In order to better deal with large-scale complex policy sets, an optimization algorithm based on the DPCA (Density Peak Cluster Algorithm)[16] is proposed, which can realize clustering analyses of arbitrary shapes and multi-dimensional policy sets and automatically determine parameters to improve the clustering performance for policy sets.
(2) In the policy matching stage, policy sets are processed by labeling according to clustering results to form a new policy tag set. The tag set can significantly speed up policy matching and save storage space.
(3) A policy evaluation engine based on the optimization algorithm, named DPEngine, is constructed and has an excellent evaluation performance for large-scale complex policy sets.

The rest of this paper is organized as follows. In section "Related works", some related works are reviewed. Section "Preliminary" introduces the details of XACML and policy evaluation. Section "Approach overview" provides a brief introduction to the proposed policy evaluation engine, named DPEngine. In section "An effective policy evaluation engine", we discuss an improved clustering algorithm and the details of the DPEngine. Section "Experimental results and analyses" shows experiment results and analyses. Finally, the paper is summarized in section "Conclusions".

Related works

In the past few years, most researchers have made great contributions to improve the PDP evaluation performance. These contributions are mainly divided into the following aspects.

Distributed authorization systems. Compared with traditional centralized authorization models, distributed authorization models can handle plenty of requests more efficiently. Wang et al.[17] analyze topological characteristics of different policies and apply a greedy algorithm to divide a policy set into multiple subsets, making them suitable for distributed systems. Daniel et al.[18] redefine a default XACML architecture, such that the PAP can effectively manage policy sets belonging to the XACML architecture. Deng et al.[19] make a policy decomposition technology based on an ant colony algorithm, which decomposes a policy set into multiple sub-policies, so as to reduce the cost of policy deployment. Lischka et al.[20] put forward a distributed method, which analyzes the architecture of XACML policies and makes authorization decisions according to the attributes of policy sets.

Elimination of conflicts and redundancies. Jebbaoui et al.[21] develop a new method based on sets and semantics. First, they design a set with an intermediate representation to reduce the complexity of policies. Second, the meanings of policy rules are analyzed through a structural inference of rules and semantic verification of deductive logic, detecting defects, conflicts and redundancies in rules. The method provides an accurate and effective analysis of XACML policies. Wang et al.[22] present a new policy evaluation engine based on a multi-level optimization technology. The policy evaluation engine adopts a multi-cache mechanism to eliminate redundant rules. Ngo et al.[23] address an XACML model based on decision graphs. Logical expressions in policies are transformed into a decision tree structure. The XACML model can effectively detect conflicts and redundancy in policies. Shaikh et al.[24] introduce a new technique to detect inconsistent policies. They put forward an improved algorithm to analyze and construct a decision tree that includes a Boolean expression normalization policy set, and then an anomaly detection algorithm is executed to secure a final detection result.

Although the elimination of conflicts and redundancies is beneficial to improving the PDP evaluation efficiency, existing research efforts still lack better solutions for dealing with large policy sets.

Decision graphs. Liu et al.[25] develop a policy evaluation engine called XEngine, which advances a fast policy evaluation algorithm to speed up processing requests. Ros et al.[26] raise a new policy evaluation model based on a decision tree structure. The model can not only search applicable rules quickly but also evaluate requests more efficiently, thus further improving the policy evaluation efficiency. Deng et al.[27] introduce an efficient policy evaluation engine combined with an automaton theory. The engine establishes an attribute bitmap for

each policy and makes evaluation results quickly through the attribute bitmaps. Turkmen et al.[28] come up with a policy evaluation method using the SMT (SAT modulo theories). The method not only improves the policy evaluation efficiency but also can be applied to a wide range of other fields.

Decision graphs can effectively speed up the policy evaluation efficiency. However, if the size of a policy set is extraordinarily large, its decision tree will be difficult to build.

Other methods. Marouf et al.[29] propose an adaptive optimization method for XACML policies. They process policy sets with the *k*-means algorithm and dynamically optimize ranking access control policies, thus improving the policy evaluation efficiency. Mourad et al.[30] construct a framework, named SBA-XACML, which is based on a set language. The SBA-XACML framework converts an XACML structure into a readable mathematical grammar. The framework not only improves the policy evaluation efficiency but also allows the detection of conflicts and redundancies. Deng et al.[31] address an optimized XACML policy management scheme based on bitmap storage and HashMap. A policy set is digitized, and then a sequential storage structure based on an array is established, such that rules can be indexed quickly. Cheminod et al.[32] present a comprehensive access control policy refinement and validation method, which can detect errors in the policy execution in time and perform policy matching correctly. Rezvani et al.[33] use ASP (Answer Set Programming) to analyze various characteristics of an XACML policy including redundancies and conflicts, thus improving the policy evaluation efficiency.

With the rapid development and progress of computer technologies, the scale and complexity of policy sets in an authorization service system are increasing. The existing research has been unable to meet the application requirements for large-scale policy sets. This calls for an efficient and stable policy evaluation engine to deal with large-scale policy sets.

Preliminary

Before describing the proposed evaluation engine in detail, this section introduces the concept of XACML and the problem description of policy evaluation.

Concepts of XACML. In XACML, a policy set contains thousands of policies, and each policy has many rules. A rule in a policy set can be expressed in Eq. (1).

$$Rule = \langle Subject,\ Resource,\ Action,\ Condition,\ Effect \rangle \tag{1}$$

where the *Subject* is request proposer, the *Resource* usually contains data, files, and URL. The *Action* typically includes reading and writing. The *Condition* represents constraints. The *Effect* has two values: Permit and Deny. The *Effect* is the core attribute of a rule. It is used to decide whether to accept the current request and perform the corresponding operation. Accordingly, a request can be expressed as

$$\text{Request} = \langle Subject,\ Resource,\ Action,\ Condition \rangle \tag{2}$$

where attributes contained in a request are consistent with those in a rule.

Problem description of policy evaluation. Policy evaluation is a process of matching attributes in a request with attributes of rules in a policy set. The attributes to be matched include *Subject*, *Resource*, *Action* and *Condition*. If attributes in a request are different from those in all rules of a policy set, users will be denied access. On the contrary, the corresponding operation is performed according to the value of *Effect* in the rule. In an authorization service system, policy sets of the access control are very large and complex. Each request needs many matches to find the corresponding policy to perform correct operations. At the same time, the number of requests is also very large, and multitudes of requests also cause a great burden to the system. Therefore, in a large-scale authorization service system, deciding how to improve the efficiency of policy evaluation is of great significance to the access control ability of the whole system.

Approach overview

Based on an optimization algorithm, a policy evaluation Engine, named DPEngine, is conducted to improve the PDP evaluation performance. The DPEngine can load policies, evaluate requests and return evaluation decisions to the PEP. The DPEngine has three main functions, namely, preprocessing policy sets, clustering policy sets, and matching policies. Its working process is shown in Fig. 1.

In the preprocessing part of policy sets, the DPEngine can digitize attributes of each policy in policy sets. Each policy in a policy set has five attributes: *Subject*, *Resource*, *Action*, *Condition*, and *Effect*. These attributes are translated into digital representations by a random sort. The result is a complete digitized policy set, which is beneficial to the clustering and matching parts later. The details of preprocessing are covered in section "Preprocess for policy sets".

After the preprocessing part is finished, the second part is to cluster policy sets. The core of DPEngine is to obtain the best cluster results by adopting an optimization algorithm. In order to improve clustering effects, a modified version of the GWO (Grey Wolf Optimizer)[34] is used to optimize parameters in the optimization algorithm. Then, according to the obtained optimal parameters, we perform clustering to secure final clustering results. The whole clustering process is shown in Fig. 2.

The third part is the policy matching. Based on clustering results, similar policies are grouped together and labeled with the same tags. When a request arrives, its corresponding label can be determined based on

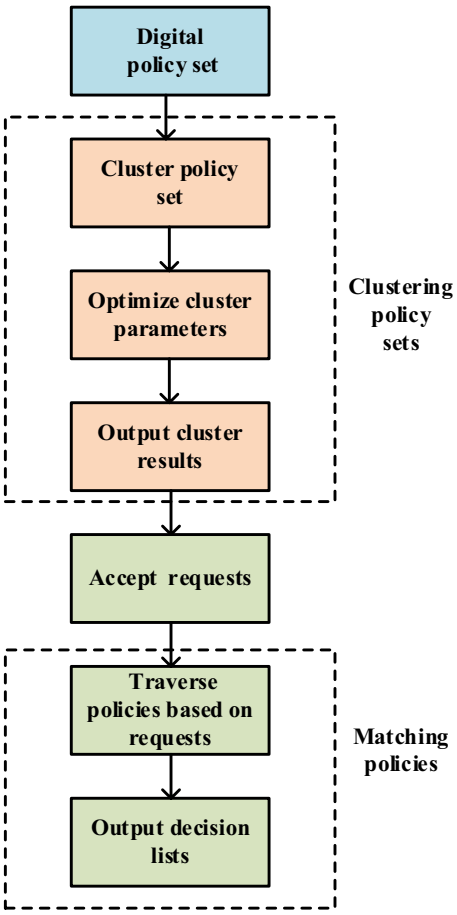


Figure 1. Working process of DPEngine.

its attributes. By quickly retrieving policies with the same tag, a match between policies and requests can be completed. According to the match result, an evaluation decision can be made. The evaluation decision will be returned to the PEP to help a service system make an authorization decision. The details of policy matching are introduced in section "Matching policies".

An effective policy evaluation engine

In order to improve the policy evaluation performance on large-scale complex policy sets, we construct an efficient policy evaluation engine, called DPEngine. The engine can preprocess policy sets, use an optimization algorithm to efficiently and reasonably cluster policy sets, and make fast policy matching according to the clustering results. This section covers the details of the DPEngine.

Preprocess for policy sets. In order to meet the needs of the algorithm, a policy set should be preprocessed. An initial policy is shown in Fig. 3.

In Fig. 3, each policy has five attributes: *Subject*, *Resource*, *Action*, *Condition*, and *Effect*. Each attribute is represented as a consecutive integer starting from 0, based on a random sorting result in a policy set. The *Effect* is represented by 0 or 1. The processed policy set is shown in Table 1.

Clustering for policy sets. In order to perform effective and reasonable clustering on large-scale complex policy sets, we present an optimization algorithm based on the DPCA (Density Peak Clustering Algorithm)[12] and GWO (Grey Wolf Optimizer)[34], which are explained in detail below.

Density peak clustering algorithm. The Density Peak Clustering Algorithm is a classic density clustering algorithm, which can effectively cluster arbitrary shape data sets. This is appropriate for dealing with complex policy sets. According to the preprocessing results of policy sets, each policy is transformed into a data point that has five attributes: *Subject*, *Resource*, *Action*, *Condition*, and *Effect*, thus forming a new policy point set. The algorithm mainly has two assumptions for policy sets:

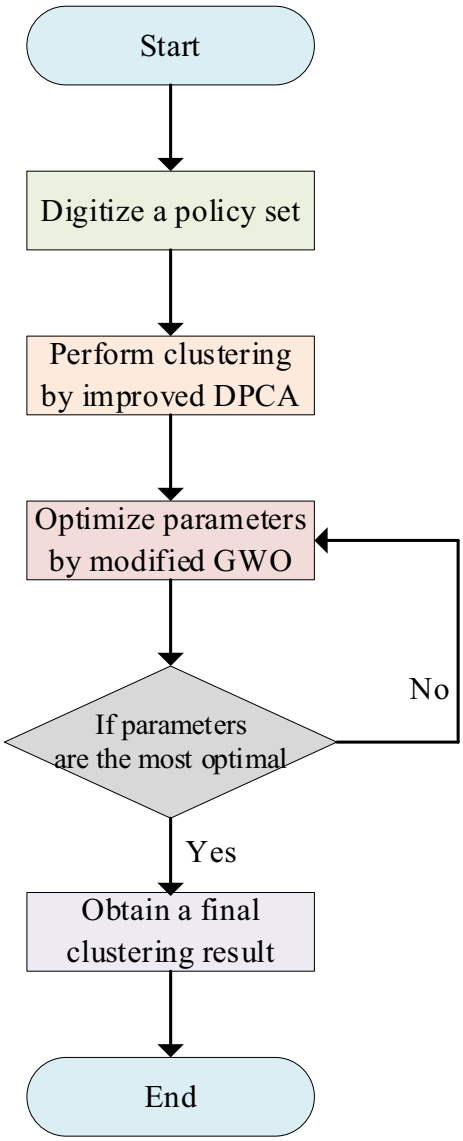


Figure 2. Clustering process of an improved DPCA.

(1) The number of policy points within a certain range around a policy point is called local density. The local density of a core policy point is large enough, and the local density of its surrounding policy points is not higher than it.
(2) The distance between a core policy point and other core policy points with high local densities is far enough.

Based on the two assumptions, the local density of each policy point i can be expressed as

$$\rho_i = \sum_{j \neq i} \chi(dist_{ij} - dist_c) \tag{3}$$

where $dist_{ij}$ denotes the Euclidean distance between policy point i and policy point j, and $dist_c$ represents the cut-off distance. The function $\chi(x)$ can be expressed as

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{4}$$

According to Eqs. (3) and (4), the number of policy points whose distance from policy point i is less than the cut-off distance is taken as the local density of the policy point i.

```
<PolicySet PolicySetId="ASMS" PolicyCombiningAlgId="Permit-Overides">
 <Target/>
 <Policy PolicyId-"A1" RuleCombinationAlgId="Deny-Overrides">
  <Target>
  <Rule RuleId="1" Effect="Deny">
    <Target>
      <Subjects><Subject>      Customer  </Subject>
                <Subject>      Visitor   </Subject></Subjects>
      <Resources><Resource>    Money     </Resource></Resources>
       <Actions><Action>       Add       </Action></Actions>
    </Target>
    <Conditions><Condition>    ALL       </Conditions><Condition>
  </Rule>
  <Rule RuleId="2" Effect="Permit">
    <Target>
      <Subjects><Subject>      Employee  </Subject>
                <Subject>      Secretary </Subject></Subjects>
      <Resources><Resource>    Books     </Resource></Resources>
      <Actions><Action>        Reduce      </Action></Actions>
    </Target>
    <Conditions><Condition>    ALL       </Conditions><Condition>
  </Rule>
  <Rule RuleId="3" Effect="Deny">
    <Target>
      <Subjects><Subject>      Visitor     </Subject>
                <Subject>      Employee  </Subject></Subjects>
      <Resources><Resource>    Time      </Resource></Resources>
       <Actions><Action>       Add         </Action></Actions>
    </Target>
    <Conditions><Condition>    ALL         </Conditions><Condition>
  </Rule>
 </Policy>
</PolicySet>
```

Figure 3. A simple example of an XACML policy set.

No.	Subject	Resource	Action	condition	Effect
0	0	0	0	0	0
1	1	1	1	0	1
2	2	2	0	0	1

Table 1. A processed policy set.

The key lies in the selection of δ_i-cluster center distance that is used to measure the distance between different core policy points. According to the definition of local density, we can calculate the local density of each policy point i, and determine the cluster center distance δ_i according to the density. The densities of all policy points are sorted from large to small. If the policy point i is a point with the highest density, its cluster center distance δ_i is equal to the distance from the farthest policy point j to it, which can be expressed as

$$\delta_i = \max_j(dist_{ij}) \tag{5}$$

If a policy point i is not a point with the highest density, its cluster center distance is equal to the distance with the nearest policy point j whose density is bigger than the point, which can be expressed as

$$\delta_i = \min_{j:\rho_j > \rho_i}(dist_{ij}) \tag{6}$$

As demonstrated by Hassan and Rashid[12], after the cluster center distances and local densities of all policy points are calculated, the result is represented as a two-dimensional decision graph in Fig. 4 by plotting their values according to δ_i and ρ_i as the coordinate axes.

As can be seen from Fig. 4, a core policy point is far away from the coordinate axis, while a normal policy point is near the horizontal axis, and an abnormal policy point is near the vertical axis. In the case of small policy sets, core policy points can be easily obtained by observations. A quantitative analysis can be done with Eq. (5).

$$\gamma_i = \rho_i \times \delta_i \tag{7}$$

According to Eq. (5), the values of γ_i for all policy points are calculated, and then they are ranked from largest to smallest. The larger the γ_i of a policy point is, the more likely it is to be a core policy point. By taking γ_i as a vertical axis and policy point i as a horizontal axis, a two-dimensional plane is drawn, as shown in Fig. 5.

As can be seen from Fig. 5, the curve of normal policy points is relatively smooth, while that of core policy points is extremely steep. There is a clear gap between the two types of policy points. It is easy to distinguish a

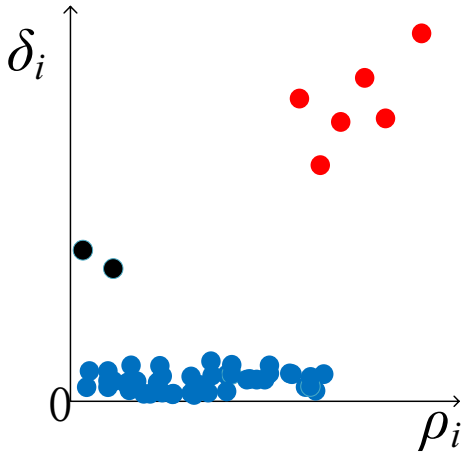


Figure 4. ρ_i–δ_i decision graph.

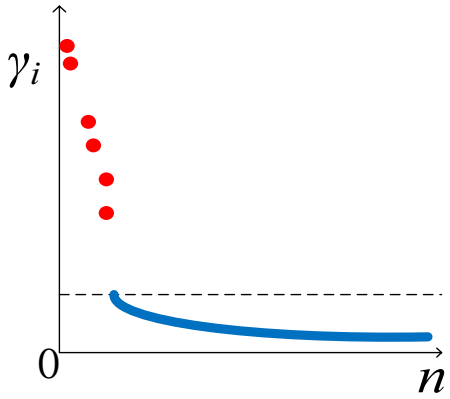


Figure 5. Value of γ_i in descending order.

core policy point from a normal policy point. Therefore, the DPCA can realize a clustering analysis of multi-dimensional policy sets by constructing two-dimensional plane graphs.

However, the number of core policy points is difficult to observe for a large-scale policy set. This shortcoming restricts the cluster performance. At the same time, due to the limitation of clustering performance, further improvement of policy evaluation efficiency is also constrained. Therefore, in order to achieve a better cluster effect, we make an optimization method that adopts the modified GWO[34] to optimize its parameters. The main optimization target is to acquire the optimal number of core policy points.

A modified grey wolf optimizer. In recent years, a large number of swarm intelligence optimization algorithms are applied to various disciplines and fields. The GWO (Grey Wolf Optimizer)[34] is a classic swarm intelligent optimization algorithm which is an optimized search method inspired by the preying activities of grey wolves and has the advantages of strong convergence, fewer parameters and easy implementation. Rashid et al.[35] propose an improved version of the GWO, which is used to optimize parameters of the RNN to improve classification accuracy. Mohammed et al.[36] use the *k*-means algorithm to enhance the limitations of grey wolves attack and search process. Rashid et al.[37] improve the performance of GWO to find the optimal solution by embedding the search phase of GWO into the development phase of WOA (Whale optimization algorithm)[38]. Khandelwal et al.[39] develop a modified GWO to solve the TNEP (transmission network expansion planning) problem. With the ideas of Rashid et al., we also propose a modified version to improve the optimization performance of the GWO. The details of our algorithm improvements are described below.

The GWO seeks the best solution to problems by simulating hunting behaviors of grey wolves. When grey wolves search for their prey, they gradually approach and surround it. A mathematical model simulating the behavior is as follows.

$$\overrightarrow{D} = \left| \overrightarrow{C} \cdot \overrightarrow{X_p}(t) - \overrightarrow{X}(t) \right| \tag{8}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X}_p(t) - \overrightarrow{A} \cdot \overrightarrow{D} \tag{9}$$

$$\overrightarrow{A} = 2\overrightarrow{h} \cdot \overrightarrow{r_1} - \overrightarrow{h} \tag{10}$$

$$\overrightarrow{C} = 2\overrightarrow{r_2} \tag{11}$$

where t is the number of current iterations, $\vec{A}$ and $\vec{C}$ are synergistic coefficient vectors, $X_p(t)$ represents a position vector of the prey, $X(t)$ represents a position vector of the current grey wolves, h decreases linearly from 2 to 0 during a whole iteration, and r_1/r_2 is a random vector in [0, 1].

In order to simulate the search behaviors of grey wolves, it is assumed that wolves *a*, *b*, and *c* have a strong ability to identify potential prey locations. Therefore, during each iteration, the best three wolves *a*, *b* and *c* in the current population are retained, and the positions of other search agents are updated based on their location information. A mathematical model simulating the behavior can be expressed as follows.

$$\overrightarrow{D}_a = \left| \overrightarrow{C}_1 \cdot \overrightarrow{X}_a - \overrightarrow{X} \right| \tag{12}$$

$$\overrightarrow{D}_b = \left| \overrightarrow{C}_2 \cdot \overrightarrow{X}_b - \overrightarrow{X} \right| \tag{13}$$

$$\overrightarrow{D}_c = \left| \overrightarrow{C}_3 \cdot \overrightarrow{X}_c - \overrightarrow{X} \right| \tag{14}$$

$$\overrightarrow{X}_1 = \overrightarrow{X}_a - \overrightarrow{A}_1 \cdot \overrightarrow{D}_a \tag{15}$$

$$\overrightarrow{X}_2 = \overrightarrow{X}_b - \overrightarrow{A}_2 \cdot \overrightarrow{D}_b \tag{16}$$

$$\overrightarrow{X}_3 = \overrightarrow{X}_c - \overrightarrow{A}_3 \cdot \overrightarrow{D}_c \tag{17}$$

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X}_1 + \overrightarrow{X}_2 + \overrightarrow{X}_3}{3} \tag{18}$$

where $\vec{X}a$, $\vec{X}b$, and $\vec{X}c$ respectively represent position vectors of wolves *a*, *b*, and *c* in the current population; $\vec{X}$ represents a wolf position vector; $\vec{D}a$, $\vec{D}b$, and $\vec{D}c$ respectively represent the distances between the current candidate wolf and the three optimal wolves.

In the original algorithm, there are three optimal wolves to help find the best solution. However, the optimal solution may fall in the remaining locations. Therefore, our improvement is to add another wolf to participate in location updates, called wolf *d*. In the proposed improved GWO, the position of wolf *d* is also updated according to the positions of wolves *a*, *b*, and *c*. The following position-updated is used for the wolf *d*:

$$\overrightarrow{D}_d = \frac{1}{3}\left(\overrightarrow{D}_a + \overrightarrow{D}_b + \overrightarrow{D}_c\right) \tag{19}$$

$$\overrightarrow{X}_4 = \overrightarrow{X}_d - \overrightarrow{A}_4 \cdot \overrightarrow{D}_d \tag{20}$$

Since the wolf *d* is added as another optimal solution, Accordingly, Eq. (18) is updated as follows.

$$\overrightarrow{X}(t+1) = \frac{1}{4}\sum_{i=1}^{4} \overrightarrow{X}_i \tag{21}$$

A wolf position updating process is shown in Fig. 6. The position of a candidate solution eventually falls into the random position defined by wolves *a*, *b*, *c* and *d*. Grey wolves mostly search for their prey according to the position of *a*, *b*, *c*, and *d*. In summary, the algorithm creates a random grey wolf population. During iterations, the leading wolves estimate the most likely prey location. Each candidate solution updates its distance from the prey. Finally, the algorithm is terminated after the specific situation is satisfied. The details of the modified Grey Wolf Optimizer are given in Algorithm 1.

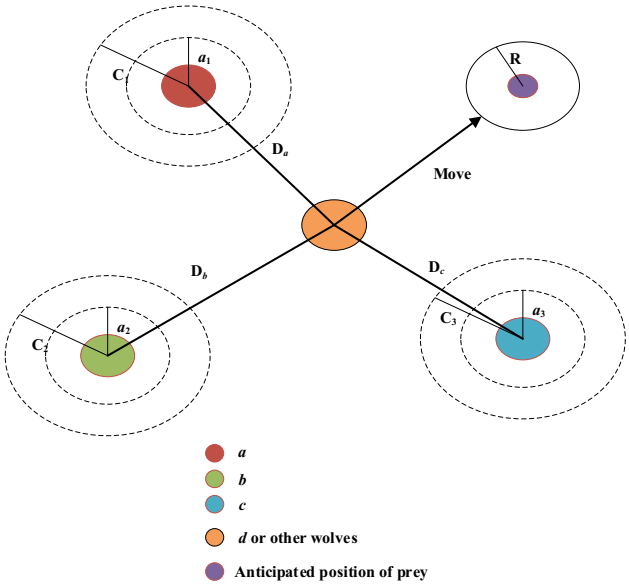


Figure 6. Position updating of wolves.

Algorithm 1. A Modified Grey Wolf Optimizer

```
Initialize a Grey wolf population Y_i (i= 1,2,...,n)
Initialize the synergistic coefficient vectors h, A and C, Max number of iterations n
Y_a= the best search agent
Y_b =the second best search agent
Y_c =the third best search agent
Y_d =another best search agent
while (t < n)
    for each search agent
        Update the position of the current search agent by Eqs. (12)-(21)
    end for
    Update h, A and C
    Calculate the fitness of all search agents
    Update Y_a, Y_b, Y_c, and Y_d
    t=t+1
end while
Return Y_d
```

An optimization algorithm for clustering large-scale policy sets. In order to deal with large-scale policy sets, we propose an optimization algorithm. The modified GWO is adopted to optimize the number of core policy points. In the optimization process, the Silhouette index[40] is taken as an objective function. The number of core policy points will be adjusted repeatedly until the clustering effect reaches the optimal.

At present, there are many clustering evaluation indexes. The Silhouette index is an internal evaluation index, which can evaluate the quality of clustering by the number of clusters. The Silhouette index is calculated according to Eq. (22).

$$Sil(i) = \frac{[b(i) - a(i)]}{\max[a(i),\ b(i)]} \tag{22}$$

where $a(i)$ is an average distance between the policy point i and other policy points in the same cluster. The smaller $a(i)$ is, the higher the probability that the policy point i belongs to the cluster is. $b(i)$ represents the minimum value of the average distance between policy point i and all policy points in other clusters. The larger $b(i)$ is, the lower the probability that the policy point i belongs to other clusters is. Obviously, $Sil(i)$ is a value in $[-1, 1]$. $Sil(i)$ is close to 1, which indicates that the clustering of policy point i is reasonable. If $Sil(i)$ is close to -1, it indicates that policy point i should be classified into another cluster. If $Sil(i)$ is approximately equal to 0, it means that the policy point i is on the boundary of two clusters. By calculating the Silhouette index of different

cluster numbers, we can obtain clustering results that maximize the Silhouette index. In the proposed optimization algorithm, the number of core policy points keeps updating until it finds the global optimal number of core policy points.

After the number of core policy points is obtained, the corresponding number of policy points is selected as core policy points according to their local density and cluster center distance. These core policy points are used as clustering centers to perform clustering and acquire the final clustering results. The specific process of the optimization algorithm is detailed in Algorithm 2.

Algorithm 2. An optimization for clustering large-scale policy sets

Input: A policy point set
Output: A clustering result

1 Initialize policy points $X_i(i= 1,2,...,n)$, a Grey wolf population Y_i $(i= 1,2,...,n)$
2 Initialize the cut-off distance $dist_c$, core policy point number k, synergistic coefficient vectors h, A and C, Max number of iterations m
4 Y_a= the best search agent
5 Y_b =the second best search agent
6 Y_c =the third best search agent
7 Y_d =another best search agent
7 **while** $(t < m)$
8 **for** each search agent
9 Update the position of the current search agent by Eqs. (12)-(21)
10 **end for**
11 Update h, A and C
12 Update Y_a, Y_b, Y_c and Y_d
13 Calculate the local density of each policy point ρ_i
14 **while** $(i <n)$
15 **for** each policy vector
16 if $\rho_i > \rho_j (i\neq j, j=1,2,...,n)$ then
17 Calculate δ_i by Eq. (3)
18 if $\rho_i < \rho_j (i\neq j, j=1,2,...,n)$ then
19 Calculate δ_i by Eq. (4)
20 **end for**
21 **end while**
22 Calculate γ_i of each policy point by Eq. (5)
23 Select Y_a policy points as core policy points
24 Calculate Silhouette indexes of all search agents
25 $t=t+1$
26 **end while**
27 $k \leftarrow Y_a$
28 Select k policy points as core policy points
29 Return the clustering result

In Algorithm 1, from line 1 to line 6, a policy point set, a grey wolf population and related parameters are initialized. The initial optimal solution is calculated. From line 8 to line 16, through continuous iterative updating, the optimal number of core policy points is obtained. From line 17 to line 29, the corresponding core policy points are selected as clustering centers, and the final clustering result is obtained. In Algorithm 1, the number of policy points is n and the max number of iterations is m. The time complexity of the algorithm is $O(mn)$. The proposed optimization algorithm provides a more accurate method to determine the number of core policy points for large-scale policy sets. Therefore, its clustering effect for complex large-scale policy sets is further improved.

Matching policies. The DPEngine obtains clustering results in policy sets by using the proposed optimization algorithm. Based on the clustering results, similar policies are grouped and labeled with the same tags to form a new policy tag set. When requests from the PEP arrive, their corresponding tags are determined by verifying the attributes of requests. By the new policy tag set, a quick retrieval of policies with the same tags can be achieved. Once the retrieval is completed, one can find policies that match the attributes of requests. According to the value of *Effect* in policies, the corresponding decision list is generated and the appropriate authorization decisions can be made. The specific process of policy matching is shown in Fig. 7.

Experimental results and analyses

In this part, the DPEngine is compared with several existing policy evaluation engines, and experimental results are analyzed and discussed. The implementation of DPEngine is based on Python 3.8. The experiments are made on a laptop with a Windows operating system, which has 16 GB of memory and 1.80 GHz AMD Ryzen-7 4800U processors.

In order to verify that the DPEngine can efficiently evaluate complex and large-scale policy sets, we make three experiments as follows.

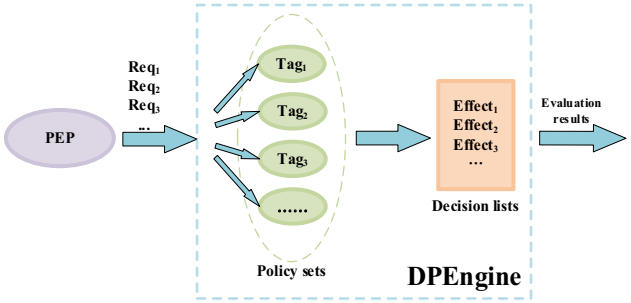


Figure 7. Process of policy matching.

(1) On three different policy sets, an optimization algorithm is used to cluster policies. In order to achieve the best cluster effect, the optimization algorithm adopts the modified GWO algorithm to optimize its parameter. The main optimization parameter is the core policy points number.
(2) We compare and analyze the policy evaluation time of five policy evaluation engines: the DPEngine, Sun PDP[41], HPEngine[42], XEngine[25] and SBA-XACML[30], on three policy sets of different sizes and complexities.
(3) We compare evaluation time experiments of the DPEngine on three policy sets of different sizes and complexities.

The Sun PDP is the first open-source XACML evaluation engine. It has been widely used and is an accepted industry standard. The HPEngine adopts a statistical analysis mechanism to reduce policy size and optimize matching methods. The XEngine is a policy evaluation engine that first transforms an XACML policy into a numeric policy. The SBA-XACML contains formal semantics and algorithms that use mathematical operations to improve policy evaluation performance.

Experimental policy sets. In order to acquire authentic and credible experimental results, we simulate practical application scenarios and select three XACML access control policy sets widely used as follows.

(1) Auction Management System (ASMS)[43]: A policy set of an access control rights management system. It primarily serves an auction process and restricts access to each user.
(2) Virtual Management System (VMS)[44]: A policy set related to a virtual meeting management system. It is mainly used to control access rights of administrators, supervisors, and employees, etc.
(3) Library Management System (LMS)[45]: A policy set for a library access service. Users include administrators, teachers, students, etc.

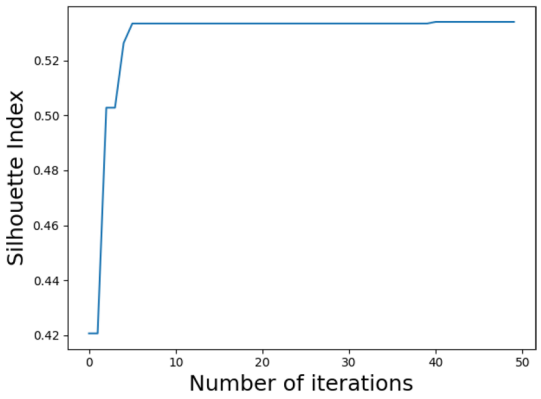
According to the experiment needs, we expand the number of rules for the ASMS, LMS, and VMS to 10,000, 50,000 and 100,000, respectively. We conduct experiments on the three policy sets of different sizes and complexities.

Parameter optimization experiments. Before making the formal policy evaluation experiment, to achieve the best clustering accuracy, we use the modified GWO to optimize the core policy point number k. Experiments are conducted for the ASMS, LMS, VMS with 10,000, 50,000, and 100,000 rules, as shown in Figs. 8, 9, 10. In the optimization process, we choose the Silhouette index as an objective function. Using the modified GWO algorithm, the number of core policy points is continuously optimized through multiple clustering. By multiple clustering, the value of the Silhouette index changes with the number of core policy points. Figures 8, 9 and 10, respectively, represent the process of obtaining the value of the maximum Silhouette index through continuous optimization using the proposed optimization algorithm on ASMS, LMS and VMS policy sets of different scales. When the value of the Silhouette index reaches the maximum and tends to be stable, it indicates that the number k of core policy points has reached the best.

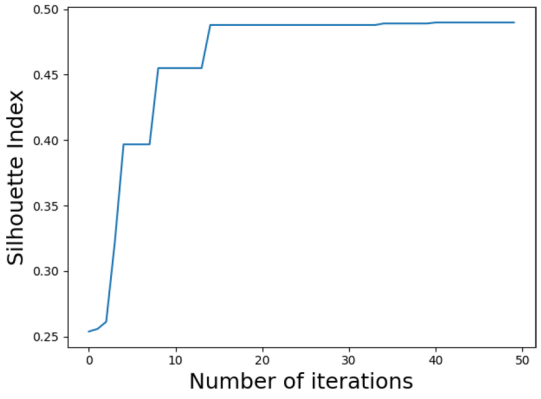
After the optimal number k of core policy points is obtained on the three policy sets, k core policy points are selected as cluster centers to gain the optimal cluster results. The clustering effect has a crucial impact on the speed of policy evaluation. The better clustering effects are of great help to the subsequent policy evaluation experiments.

Comparisons of evaluating performance. After achieving the best clustering effect, for the ASMS, LMS, VMS with 10,000, 50,000, and 100,000 rules, we test the evaluation time of DPEngine, Sun PDP, HPEngine, XEngine and SBA-XACML. In the experiment, the number of requests ranges from 1000 to 10,000, with an interval of 1000. The experimental results are shown in Figs. 11, 12, 13.

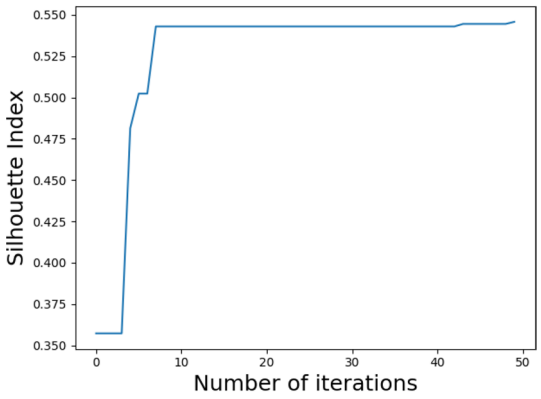
From Figs. 11, 12, 13, we can see that



(a) 10000 rules



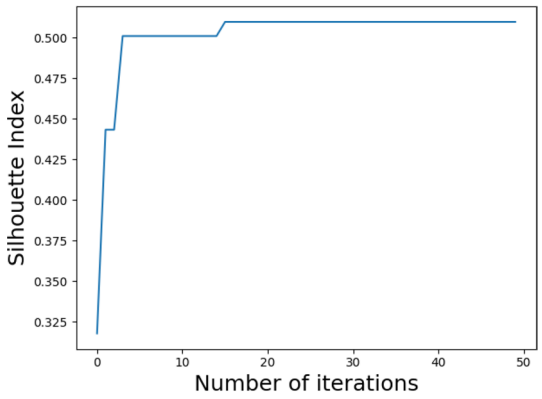
(b) 50000 rules



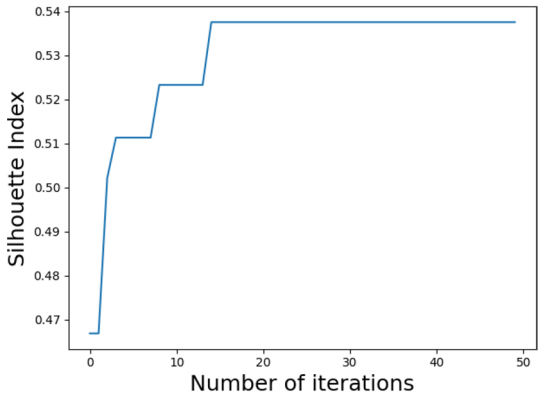
(c) 100000 rules

Figure 8. Parameter optimization on ASMS policy set.

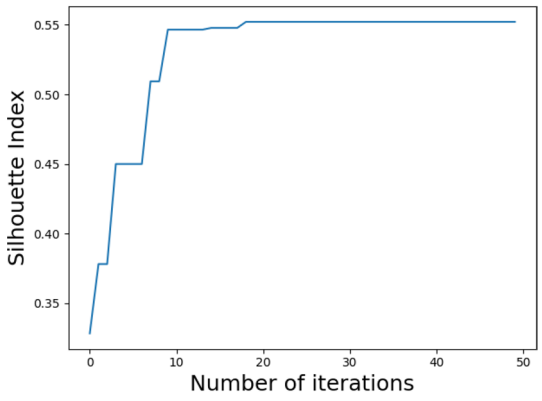
(1) for the three policy sets with 10,000, 50,000 and 100,000 rules respectively, the average evaluation time of DPEngine is much less than that of the Sun PDP, HPEngine, XEngine and SBA-XACML, respectively.
(2) as the number of requests increases, the evaluation time of the Sun PDP, HPEngine, XEngine and SBA-XACML increases rapidly. However, the growth rate of the DPEngine evaluation time is much slower than that of the four evaluation engines above, and there is no significant increase.



(a) 10000 rules



(b) 50000 rules



(c) 100000 rules

Figure 9. Parameter optimization on LMS policy set.

(3) as the scales of the three policy sets grow, the evaluation time curves of DPEngine have stable shapes without significant fluctuations, and the evaluation time for processing a single request is always controlled within approximately 0.6 ms. Thus, the DPEngine has great stability for large-scale policy sets.

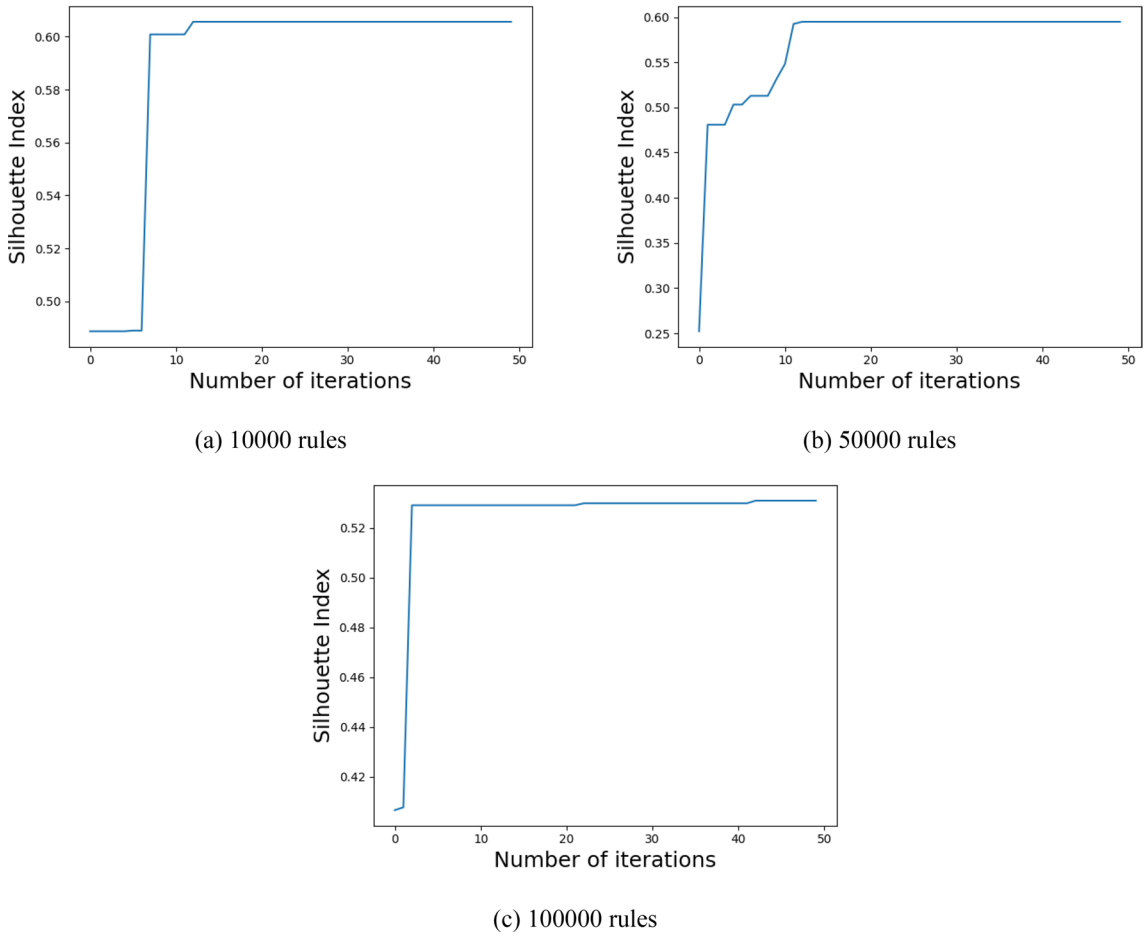


Figure 10. Parameter optimization on VMS policy set.

Comparisons of DPEngine on different policy sets. The experiments are conducted on three policy sets with different sizes. We randomly generate 1000, 2000, …, 10,000 access requests, and record the evaluation time of DPEngine. For different sizes of policy sets with 10,000, 50,000, and 100,000 rules, the variations of evaluation time with the number of requests are shown in Fig. 14.

From Fig. 14, we conclude that

(1) for the three policy sets with 10,000, 50,000 and 100,000 rules respectively, the evaluation time of DPEngine increases linearly as the requests increase without an obvious explosive growth.
(2) for the three policy sets with 10,000, 50,000 and 100,000 rules respectively, the evaluation time of DPEngine has not increased significantly as the sizes of policy sets increase.

In order to further verify the performance of DPEngine, we conduct comparative experiments on the three policy sets with 10,000, 50,000, and 100,000 rules, respectively, which are shown in Fig. 15.

From Fig. 15, we generalize that

(1) since the complexity of policies in the LMS and VMS is higher than that of the ASMS, the evaluation time of DPEngine spent on the LMS and VMS is longer.
(2) the evaluation time curves of DPEngine on policy sets of different complexity increase linearly and slowly without obvious fluctuations, indicating the stability of DPEngine on different policy sets.

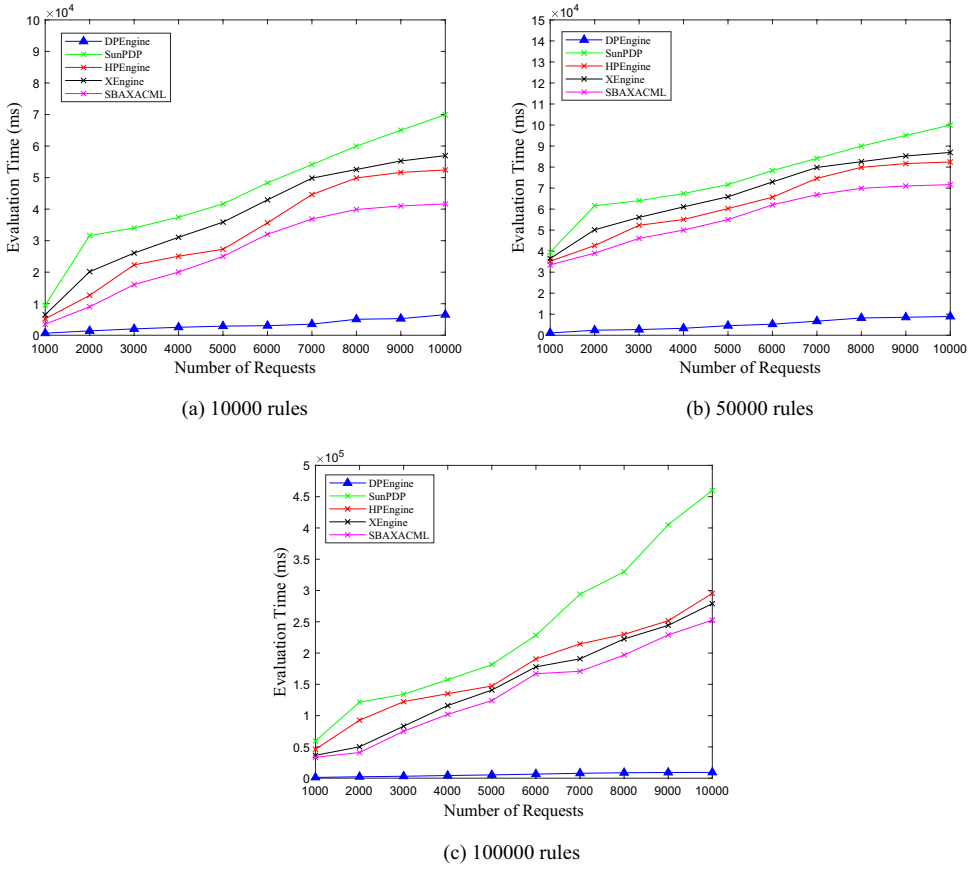
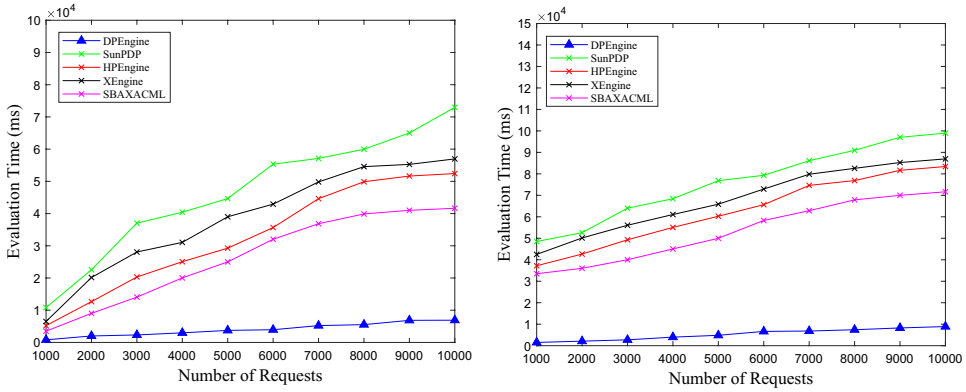


Figure 11. Evaluation performance on ASMS policy set.

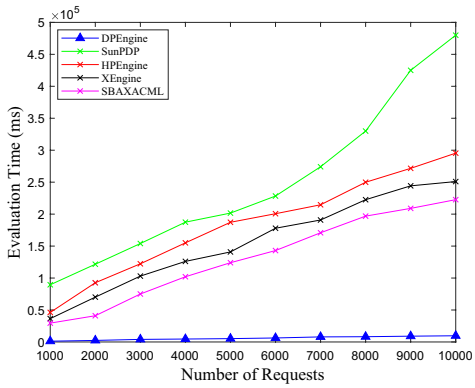
Conclusions

In order to improve the PDP evaluation performance, a policy evaluation engine named DPEngine is proposed in this paper. The DPEngine divides all rules in a policy set into different categories by an optimization algorithm based on the DPCA, and gains labels corresponding to each category. When a request arrives, the DPEngine matches it with a corresponding subject. The DPEngine runs a parallel search in the tags corresponding to the subject to quickly match specific rules. The experimental results show that the DPEngine has significantly improved the PDP evaluation performance of large-scale policy sets compared with the Sun PDP, HPEngine, XEngine, and SBA-XACML. It has better adaptability to complex policy sets. Although the proposed optimization algorithm has excellent cluster performance on XACML policy sets, the space complexity of this algorithm can be further improved. At the same time, some new and more powerful algorithms can further improve the PDP evaluation efficiency. Nowadays, deep learning has been applied in many different disciplines and achieved many amazing results. The success of deep learning also has a lot of reference significance for our research. The idea of adopting some emerging deep models may further improve the evaluation efficiency of PDP. In the future, we plan to explore better methods and continue to improve the PDP evaluation performance. The proposed method can be applied to access control of social networks[46] and parameter estimation of COVID-19 dynamical model[47,48].



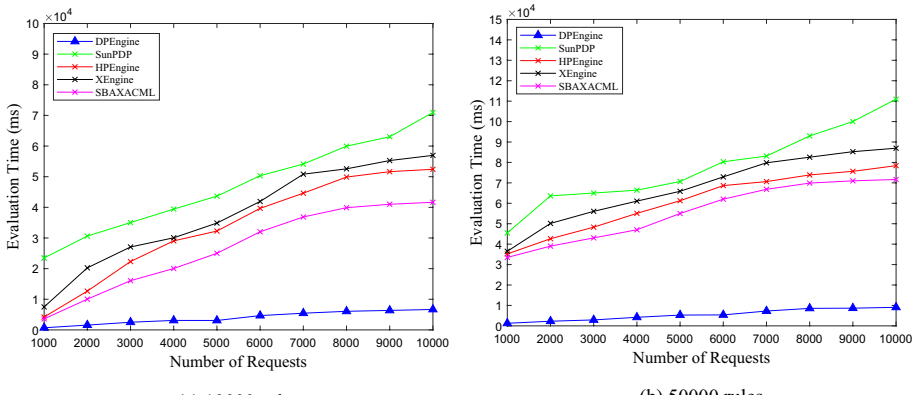
(a) 10000 rules

(b) 50000 rules



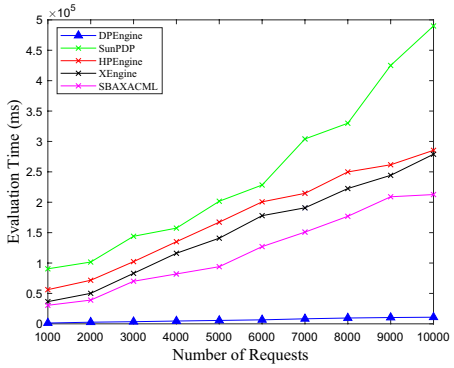
(c) 100000 rules

Figure 12. Evaluation performance on LMS policy set.



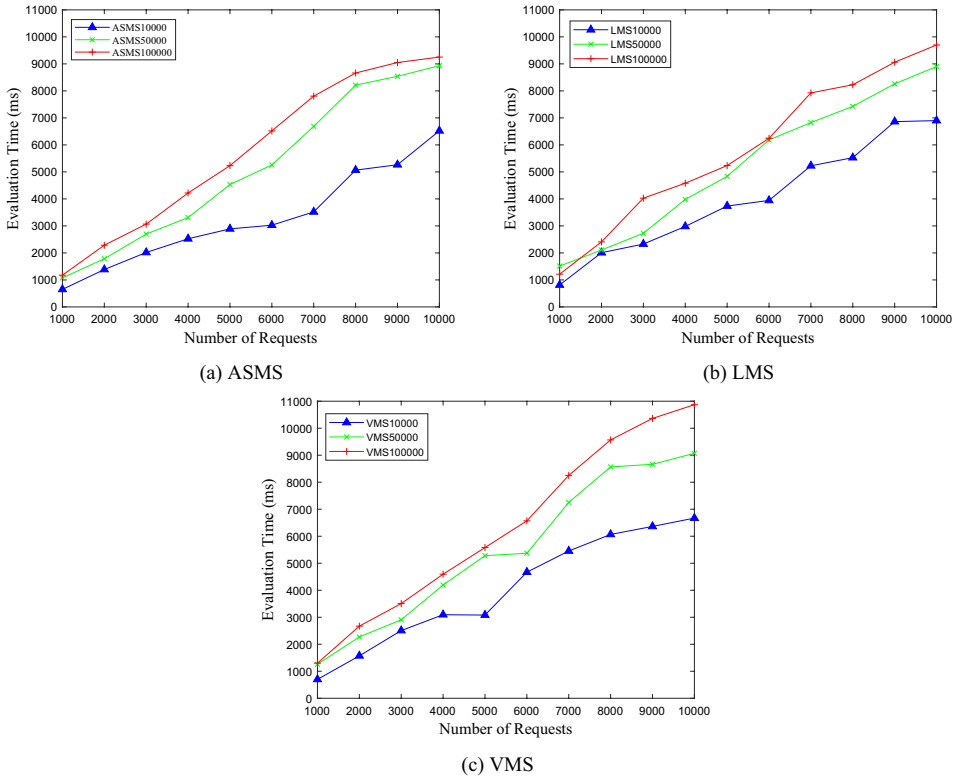
(a) 10000 rules

(b) 50000 rules



(c) 100000 rules

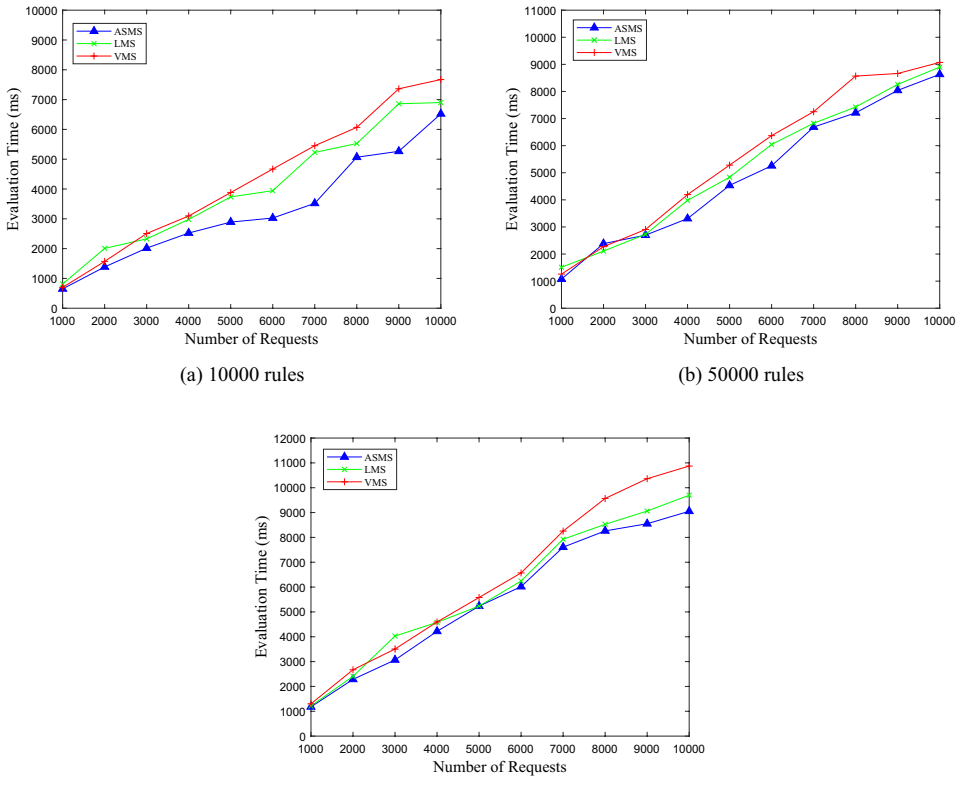
Figure 13. Evaluation performance on VMS policy set.



(a) ASMS (b) LMS

(c) VMS

Figure 14. Comparison on different sizes policy sets.



(a) 10000 rules (b) 50000 rules

(c) 100000 rules

Figure 15. Comparison on different policy sets.

Data availability

The data and code that support the findings of this study are available from the corresponding author upon reasonable request.

Received: 25 September 2021; Accepted: 7 March 2022
Published online: 23 March 2022

References

1. Qin, X., Huang, Y., Yang, Z. & Li, X. LBAC: A lightweight blockchain-based access control scheme for the internet of things. *Inf. Sci.* **554**(1), 222–235 (2021).
2. Sengupta, J., Ruj, S. & Bit, S.-D. A secure fog-based architecture for industrial internet of things and industry 4.0. *IEEE Trans. Ind. Inf.* **17**(4), 2316–2324 (2021).
3. Yu, Z. *et al.* SEI^{2}RS malware propagation model considering two infection rates in cyber-physical systems. Physica A: Statistical Mechanics and its Applications. 127207. https://doi.org/10.1016/j.physa.2022.127207 (2022).
4. Ma, H., Huang, E. & Lam, K. Blockchain-based mechanism for fine-grained authorization in data crowd sourcing. *Futur. Gener. Comput. Syst.* **106**(1), 121–134 (2020).
5. Pal, S., Gao, L., Yan, Z. & Yang, L. Game theoretical analysis on acceptance of a cloud data access control system based on reputation. *IEEE Trans. Cloud Comput.* **8**(4), 1003–1017 (2020).
6. Yu, Z., Sohail, A., Nofal, T. A. & Tavares, J. M. R. S. Explainability of neural network clustering in interpreting the COVID-19 emergency data. *Fractals*. https://doi.org/10.1142/S0218348X22401223 (2021).
7. Hur, J. & Noh, D. K. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Trans. Parallel Distrib. Syst.* **22**(7), 1214–1221 (2021).
8. Zhang, S. & Hong, Y.: Research and application of XACML-based fine-grained security policy for distributed system. In *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer*, 1848–1851 (2013).
9. Qiu, H., Di, X. & Li, J. Formal definition and analysis of access control model based on role and attribute. *Inf. Secur. Appl.* **43**(1), 53–60 (2018).
10. Riad, K. & Cheng, J. Adaptive XACML access policies for heterogeneous distributed IoT environments. *Inf. Sci.* **548**(1), 135–152 (2021).
11. Deng, F. *et al.* An efficient policy evaluation engine for XACML policy management. *Inf. Sci.* **547**(1), 1105–1121 (2021).
12. Hassan, B. A. & Rashid, T. A. A multidisciplinary ensemble algorithm for clustering heterogeneous datasets. *Neural Comput. Appl.* **3**(1), 1–24 (2021).
13. Hassan, B. A., Rashid, T. A. & Mirjalili, S. Formal context reduction in deriving concept hierarchies from corpora using adaptive evolutionary clustering algorithm star. *Complex Intell. Syst.* **7**(1), 2383–2398 (2021).
14. Mohammed, H. M. & Rashid, T. A. Chaotic fitness-dependent optimizer for planning and engineering design. *Soft Comput.* **25**, 14281–14295 (2021).
15. Askari, S. Fuzzy C-Means clustering algorithm for data with unequal cluster sizes and contaminated with noise and outliers: Review and development. *Expert Syst. Appl.* **165**(1), 113856 (2021).
16. Rodriguez, A. & Laio, A. Clustering by fast search and find of density peaks. *Science* **344**(6191), 1492–1496 (2014).
17. Wang, X., Shi, W., Xiang, Y. & Li, J. Efficient network security policy enforcement with policy space analysis. *IEEE/ACM Trans. Netw.* **24**(5), 2926–2938 (2016).
18. Daniel, D., Ginés, D., Félix, G. & Gregorio, M. Managing XACML systems in distributed environments through meta-policies. *Comput. Secur.* **48**(1), 92–115 (2015).
19. Deng, F., Chen, P., Zhang, L.-Y. & Li, S.-D. Study on distributed policy evaluation engine in SOA environment. *J. Huazhong Univ. Sci. Technol.* **42**(12), 106–110 (2014).
20. Lischka, M., Endo, Y. & Cuenca, M.: Deductive policies with XACML. In *Proceedings of the 60th ACM workshop on secure web services*, 37–44 (2009).
21. Jebbaoui, H., Mourad, A., Otrok, H. & Haraty, R. Semantics-based approach for detecting flaws, conflicts and redundancies in XACML policies. *Comput. Electr. Eng.* **44**(1), 91–103 (2015).
22. Wang, Y., Feng, D., Zhang, L. & Zhang, M. XACML policy evaluation engine based on multi-level optimization technology. *J. Softw.* **22**(2), 323–338 (2011).
23. Ngo, C., Demchenko, Y. & De Laat, C. Decision diagrams for XACML policy evaluation and management. *Comput. Secur.* **49**(1), 1–16 (2015).
24. Shaikh, R., Adi, K. & Logrippo, L. A data classification method for inconsistency and incompleteness detection in access control policy sets. *Int. J. Inf. Secur.* **16**(1), 91–113 (2017).
25. Liu, A., Chen, F., Hwang, J.-H. & Xie, T. XEngine: A fast and scalable XACML policy evaluation engine. *IEEE Trans Comput. Sci. Eng.* **60**(12), 1802–1817 (2017).
26. Ros, S.-P. & Lischka, M.: Graph-based XACML evaluation. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, 83–92 (2012)
27. Deng, F., Wang, S.-Y., Zhang, L.-Y., Wei, X.-Q. & Yu, J.-P. Establishment of attribute bitmaps for efficient XACML policy evaluation. *Knowl. Based Syst.* **143**(1), 93–101 (2018).
28. Turkmen, F., Hartog, J.-D., Ranise, S. & Zannone, N. Formal analysis of XACML policies using SMT. *Comput. Secur.* **66**(1), 185–203 (2017).
29. Marouf, S., Shehab, M., Squicciarini, A. & Sundareswaran, S. Adaptive reordering and clustering-based framework for efficient XACML policy evaluation. *IEEE Trans. Serv. Comput.* **4**(4), 300–313 (2011).
30. Mourad, A. & Jebbaoui, H. SBA-XACML: Set-based approach providing efficient policy decision process for accessing web services. *Expert Syst. Appl.* **42**(1), 165–178 (2015).
31. Deng, F. *et al.* Establishment of rule dictionary for efficient XACML policy management. *Knowl. Based Syst.* **175**(1), 26–35 (2019).
32. Cheminod, M., Durante, L., Seno, L., Valenza, F. & Valenzano, A. A comprehensive approach to the automatic refinement and verification of access control policies. *Comput. Secur.* **80**(1), 186–199 (2019).
33. Rezvani, M., Rajaratnam, D., Ignjatovic, A., Pagnucco, M. & Jha, S. Analyzing XACML policies using answer set programming. *Int. J. Inf. Secur.* **18**(4), 465–479 (2019).
34. Mirjalili, S., Mirjalili, S.-M. & Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **69**(1), 46–61 (2014).
35. Rashid, T. A., Abbas, D. K., Turel, Y. K. & Fred, A. L. A multi hidden recurrent neural network with a modified grey wolf optimizer. *PLoS One* **14**(3), e0213237 (2019).
36. Mohammed, H. M., Abdul, Z. K., Rashid, T. A., Alsadoon, A. & Bacanin, N.: A new K-means gray wolf algorithm for engineering problems. *World J. Eng.* (2021)
37. Mohammed, H. & Rashid, T. A. A novel hybrid GWO with WOA for global numerical optimization and solving pressure vessel design. *Neural Comput. Appl.* **32**(3), 14701–14718 (2020).

38. Sai, L. & Huajing, F.: A WOA-based algorithm for parameter optimization of support vector regression and its application to condition prognostics. *Control Conference*, 7345–7350 (2017)
39. Khandelwal, A., Bhargava, A., Sharma, A. & Sharma, H. Modified grey wolf optimization algorithm for transmission network expansion planning problem. *Arab. J. Sci. Eng.* **43**(6), 2899–2908 (2018).
40. Rousseeuw, P.-J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **20**(1), 53–65 (1987).
41. Suns XACML implementation [Online]. http://sunxacml.sourceforge.net/
42. Niu, D.-H., Ma, J.-F., Li, C.-N. & Wang, L. HPEngine: High performance XACML policy evaluation engine based on statistical analysis. *J. Commun.* **35**(8), 205–215 (2014).
43. Mouelhi, T., Traon, Y.-L. & Baudry, B. Transforming and selecting functional test cases for security policy testing. In *Proceedings of the 2nd International Conference on Software Testing Verification and Validation*, 171–180 (2009).
44. Traon, Y., Mouelhi, T., Pretschner, A. & Baudry, B. Test-driven assessment of access control in legacy applications. In *Proceedings of 2008 International Conference on Software Testing, Verification, and Validation*, 238–247 (2008).
45. Mouelhi, T., Fleurey, F., Baudry, B. & Traon, Y.-L. A model-based framework for security policy specification, deployment and testing. In *Proceedings of the Eleventh International Conference on Model Driven Engineering Languages and Systems*, 537–552 (2008).
46. Yu, Z., Lu, S., Wang, D. & Li. Z. Modeling and analysis of rumor propagation in social networks. *Information Sciences* **580**, 857–873 (2021).
47. Yu, Z., Arif, R., Fahmy, M. A. & Sohail, A. Self organizing maps for the parametric analysis of COVID-19 SEIRS delayed model. *Chaos Solitons & Fractals* **150**, 111202. https://doi.org/10.1016/j.chaos.2021.111202 (2021).
48. Yu, Z., Ellahi, R., Nutini, A., Sohail, A. & Sait, S. M. Modeling and simulations of CoViD-19 molecular mechanism induced by cytokines storm during SARS-CoV2 infection. *J. Mol. Liquids* **327**, 114863. https://doi.org/10.1016/j.molliq.2020.114863 (2021).

Author contributions

Conceptualization: F.D.; Methodology: Z.Y., F.D.; Formal analysis and investigation: Z.Y.; Writing—original draft preparation: Y.Y.; Writing—review and editing: Z.Y., F.D.; Funding acquisition: Z.Y.; Resources: Z.Y.; Supervision: F.Z., Z.L.

Funding

This work was supported in part by the National Natural Science Foundation of China under Grant 61873277, in part by the Key Research and Development Program of Shaanxi Province under Grant 2019GY-056, in part by the Natural Science Foundation of Shaanxi Province in China under Grants 2022JM–317 and in part by the Guangzhou Innovation and Entrepreneurship Leading Team Project Funding under Grant 202009020008.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to F.D. or Z.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.