electronics

MDPI

Article

Access Control Role Evolution Mechanism for Open Computing Environment

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

1 Information Engineering University, Zhengzhou 450000, China; ladyexue@163.com (A.L.); twftina_w@126.com (N.W.)
2 He'nan Province Key Laboratory of Information Security, Zhengzhou 450000, China
* Correspondence: dxh37139@sina.com

Received: 16 February 2020; Accepted: 19 March 2020; Published: 21 March 2020

Abstract: Data resources in open computing environments (including big data, internet of things and cloud computing) are characterized by large scale, wide source, and strong dynamics. Therefore, the user-permission relationship of open computing environments has a huge scale and will be dynamically adjusted over time, which enables effective permission management in the role based access control (RBAC) model to become a challenging problem. In this paper, we design an evolution mechanism of access control roles for open computing environments. The mechanism utilizes the existing user-permission relationship in the current system to mine the access control role and generate the user-role and role-permission relationship. When the user-permission relationship changes, the roles are constantly tuned and evolved to provide role support for access control of open computing environments. We propose a novel genetic-based role evolution algorithm that can effectively mine and optimize roles while preserving the core permissions of the system. In addition, a role relationship aggregation algorithm is proposed to realize the clustering of roles, which provides a supplementary reference for the security administrator to give the role real semantic information. Experimental evaluations in real-world data sets show that the proposed mechanism is effective and reliable.

Keywords: data security; access control; role evolution; role mining; genetic algorithm

1. Introduction

Open computing environments (including big data [1], internet of things [2] and cloud computing [3]) provide us convenient services such as data sharing and effective computing. It has been widely used in human's production and life. By analyzing and utilizing data resources in open computing environments, we can create enormous social and economic value [4]. Furthermore, the greater amount of data and the wider sources, the more value is generated. However, open computing environments also face serious security challenges when bringing new development opportunities. Various types of security accidents occur frequently [5], such as Facebook data leakage results in the illegal access of more than 50 million users' personal data. Therefore, the unauthorized sharing of data resources will bring huge security threats to users' data. Realizing the safe and controllable sharing of data resources is the basis of application and development in open computing environments. As one of the important technologies to protect data security, access control technology [6] can enable the authorized users to access the corresponding resources in the system according to their permissions, and prohibit unauthorized users access to the data. Among them, role-based access control (RBAC) technology has become a popular and effective access control method.

RBAC (role-based access control) [7,8] uses the concept of role to establish associations between users and permissions. By establishing user-role and role-permission relationships, RBAC can achieve security protection of resources. It reduces the complexity of access control management by granting roles and revoking roles to manage user permissions. Since the role is the core of access control, how to set the correct role in the system and give it the appropriate permissions becomes the basis for implementing RBAC system. The advantages of RBAC can only be realized when the role set is suitable for the security needs of organization. To solve this problem, role engineering technology [9–11] emerged whose purpose is to get a set of roles that can accurately describe the security requirements and functional requirements of system. However, the data resources in open computing environments have features such as large volume, wide sources and strong dynamics [12,13], which can make data resources in open computing environments and traditional data resources have different application requirements and challenges in role engineering [14,15].

Current role engineering mainly includes top-down [16] mode and bottom-up mode [17,18]. The top-down mode relies on the professional knowledge of security experts to obtain a set of roles and corresponding role relationships by manual analysis. In a closed computing environment, it is safe and feasible to perform manual role management in the face of limited data resources. However, in an open computing environment, the role management of massive and dynamic data resources with the help of professional knowledge is a labor-intensive task, and its workload is huge and error-prone. It is easy to cause excessive authorization and insufficient authorization, which affects the security and availability of the system. At the same time, with the dynamic changes of data resources, the role of access control also is required to change dynamically. Therefore, it is necessary to implement the role evolution. Therefore, in an open computing environment, role management for data resources requires automatic capabilities to improve the management efficiency of dynamic permissions.

Different from the top-down mode, the bottom-up mode utilizes data mining technology to analyze the existing access control information (user-permission relationship) in the system, so as to realize the automatic generation of role set. It reduces the manual dependence of security experts, which is also known as role mining technology [17]. Essentially, role mining is a decomposition problem of a Boolean matrix. The user-permission matrix is divided into user-role matrix and role-permission matrix. It includes precise role mining and approximate role mining. To cover all user-permission relationships, traditional precise role mining technology leads to the excessive number of roles, low efficiency, and lack of dynamic adjustment capabilities. Research [19] shows that about 40% of the roles in the role set can achieve the coverage of 90% user-permission relationship. So approximate role mining can better meet the needs of open computing environment for role mining. However, current approximate role mining methods [20–22] have the risk of losing the core role in the system. It will lead to the failure of the relational business process in the system, which will affect the system's availability. In addition, current role mining methods are mostly static mining methods which are unable to meet the need of dynamic role evolution in an open computing environment. So it is difficult to balance the security and availability of the system.

Role mining is an NP-hard problem [18]. So in terms of performance, it is not acceptable to traverse all the solutions to find the optimal role set. It is very necessary to find an optimization method to get an approximate optimal solution in a finite time and to make the role have the ability of dynamical evolution. To solve the above problems, this paper proposes an access control role evolution mechanism for open computing environment, which can reduce the solving cost of space search significantly. Further, it gives the role the ability of dynamical evolution with the dynamic change of data resources. The contributions of the paper include:

(1) We propose a role evolution method based on genetic algorithm, which includes role mining and role optimization. The multi-dimensional mixed evaluation index is used to guide the role mining process, so that the security and availability of system can be considered at the same time. According to the dynamic change of the current user-permission relationship, the role set is adjusted to realize the role evolution periodically.

(2) We propose the concept of permission structure complexity (PSC) to evaluate the importance of permission and generate core permissions. A role optimization algorithm is designed to avoid the loss of core permissions in the role mining, which ensures the normal running of the business system.

(3) We design a role relationship aggregation algorithm to cluster the roles by analyzing the user and permission relationship of the roles. It can establish the semantic relationship among roles, guide the generation of roles in the real environment, and give semantic meaning to the role group.

The remainder of this paper is organized as follows. We review the related work in Section 2. Section 3 introduces preliminary knowledge and genetic algorithm. Section 4 proposes role evolution mechanism. The role evolution method based on genetic algorithm is elaborated in Section 5. The role relationship aggregation algorithm is elaborated in Section 6. Experimental evaluation of the proposed mechanism is discussed in Section 7. Finally, we summarize the paper and provide directions for future research.

2. Related Work

Kuhlmann et al. [17] firstly proposed the concept of role mining and used matrix decomposition to solve the problem. Lu et al. [23] also turned role mining problem into the optimal decomposition problem of the Boolean matrix, and used integer linear programming (ILP) to achieve role mining. Sarana et al. [24] introduced separation of duty into the role mining, using minimum biclique cover (MBC) to achieve role mining. Zhang et al. [25] designed a role mining model based on concept lattice, and found the minimum role set through the greedy algorithm of role substitution. Zhou et al. [26] proposed a semantic role mining algorithm based on formal concept analysis. It generated user's concept lattice of permission and concept lattice of attribute by calculating access control information. Then it assigned roles based on similarity analysis between concept lattices. Dong et al. [27] used bipartite networks to find roles, and proposed a method to evaluate the importance of edges in bipartite networks. It can eliminate inappropriate edges and improved the quality of generated roles. Vavilis et al. [28] studied the minimal noise role mining problem and the multiple factor optimization role mining problem to mine roles from access control logs with noise information.

To improve the efficiency of role mining, Harikat et al. [29] proposed a concurrent role mining framework to find roles under the condition of role-usage and permission-distribution cardinality constraints. For TRBAC (temporal role-based access control), Mitra et al. [30] introduced the cumulative overhead of temporal roles and permissions, using the greedy algorithm to implement the role management. Stoller et al. [31] proposed an algorithm for mining high-quality TRBAC roles from timed ACLs (Access Control Lists). The algorithm described the relationship among roles by attribute information. Narouei et al. [32] proposed a novel top-down role engineering approach that used natural language processing techniques to extract roles from documents. Kumar et al. [33] proposed a constrained role mining scheme (CRM). The scheme satisfies a cardinality condition that no role can contain more than a given number of permissions. Literature [34] proposed a prioritization method, PairCount (PC), for the role mining problem. By calculating the frequency of permissions shared by users in different roles, the priority of different roles is set to optimize the process of role mining. Vaidya et al. [35] used the subset enumeration (CM) method to design the role mining algorithm. Common permissions could exist among different roles, which met the need for overlapping permissions between different roles. Zhang et al. [36] used graph optimization (GO) theory to optimize the process of role mining to reduce the management complexity of RBAC system. Literature [37] proposed a hierarchical miner (HM). The HM is based on formal concept lattices and user-attribute information. It can balance the semantic guarantee of roles with system complexity.

In view of the difficulty in synchronous optimization for role minimization and edge concentration, Dong et al. [38] proposed a data-centric quality evaluation algorithm (DCQE), which can predict the quality of role based on the statistical characteristics of the ACL dataset. DCQE didn't need to run any role mining algorithms. Since the existing role mining method does not consider existing roles in the system, Zhai et al. [39] optimized the role mining process by calculating the similarity between the newly generated role set and the original role set. For the problem of role mining under incomplete knowledge condition, Kunz et al. [40] studied the quality criteria and feature dependence of role mining technology from 22 dimensions. Blundo et al. [41] constrained the role mining process by the number of contained permissions in the role and the number of contained in the user. Pan et al. [42] proposed a log-based role reconstruction method, which generated more targeted roles based on historical access behavior. Li et al. [43] proposed a role mining method based on fermat transformation theory and set theory for the problem of external behavior invariance and evolve-ability of legacy systems. Hachana et al. [44] studied the comparison methods among different role sets, which effectively guided security administrators to select the role set. The method can detect the misconfigured roles through the comparison.

There were also some researches [45,46] that used genetic algorithms to solve the problem of role mining, these researches only pursued the reduction of the role number. However, they ignored the consideration of other evaluation indicators. In addition, since genetic algorithm is a heuristic algorithm, the method belongs to the approximate role mining. There is a risk of losing the core permissions in the system, which will directly affect the normal running of the business system. As a result, it is difficult to directly apply those methods to the business environment. It is adapted to the actual deployment of role-based access control. In view of the shortcomings of the above methods, we use the genetic algorithm to solve the problem of role evolution and introduce the concept of permission structure complexity (PSC). When ensuring that the core permission is not lost, we optimize the performance of role mining from multiple dimensions, reduce the number of generated roles, and take into account the security and availability of the system. In addition, with the help of the role relationship aggregation algorithm, the relationship among different roles is established to provide effective support for role management.

3. Preliminaries

3.1. Terms and Definitions

This section presents preliminaries on RBAC along with the terms used in the paper.

Definition 1. *RBAC model: It is described as a quad (U, R, P, URA, PRA), U represents the user set, R represents the role set, P represents the permission set, URA represents the user-role relationship, and PRA represents the role-permission relationship.*

Definition 2. *User–permission relationship: It is described as an $f \times l$ Boolean matrix UPM, f represents the number of users, and l represents the number of permissions. If UPM(u_i,p_j) = 1, it means that the user u_i is granted the permission p_j. If UPM(u_i,p_j) = 0, it means that the user u_i is not granted the permission p_j. The UPM can be generated according to access control policy of system.*

Definition 3. *User–role relationship: It is described as an $f \times k$ Boolean matrix URM, f is the number of users and k is the number of roles. If URM(u_i,r_j) = 1, it means that the user u_i is granted the role r_j. If URM(u_i,r_j) = 0, it means that the user u_i is not granted the role r_j.*

Definition 4. *Role–permission relationship: It is described as a $k \times l$ Boolean matrix RPM, k is the number of roles and l is the number of permissions. If RPM (r_i, p_j) = 1, it means that the role r_i is granted the permission p_j. If RPM (r_i, p_j) = 0, it means that the role r_i is not granted the permission p_j.*

Definition 5. *Basic role mining problem: Given the user set U, the permission set P, and the user-permission relationship UPM, find a role set R, a user–role relationship URM, and a role-permission relation RPM, UPM = URM $\times$ RPM is satisfied, and the number of roles k is minimized.*

Definition 6. *Approximate role mining problem: Given the user set U, the permission set P, and the user-permission relationship UPM, find a role set R, a user-role relationship URM, and a role-permission relationship RPM,* $\|URM \times RPM - UPM| | \leq \delta \bullet \|UPM\|$ *is satisfied (δ is an approximation coefficient), and the number of roles k is minimized.*

Definition 7. *Dynamic role reconstruction problem: Given the role set* R_{cur}*, the user–role relationship* URM_{cur}*, the role–permission relationship* RPM_{cur}*, and new user–permission relationship* UPM_{new}*, find new role set* R_{new}*, new user-role relationship* URM_{new}*, and new role-Permission relationship* RPM_{new}*,* $\|URM \times RPM\text{-}UPM \| \leq \delta \bullet \|UPM\|$ *is satisfied, and the number of roles k is minimized.*

3.2. Genetic Algorithm

Genetic algorithm is a heuristic computational model that is influenced by natural evolutionary ideas and biogenetic mechanisms. The optimal solution of the problem is searched by simulating the natural evolution of biological population. The genetic algorithm includes many concepts similar to biology, such as individuals, populations, genes, fitness functions, selection, crossover, and mutation. Among them, the individual is an independent entity consisting of the encoded genes. The chromosomes of each individual is a candidate solution. The solution performance can be evaluated to guide the direction of genetic evolution by the fitness function. A genotype is an internal representation of a chromosome. The value of a gene is known as an allele. The phenotype is the external representation of the individual's chromosome which represents the candidate solution. Finding the correct phenotype is the key to solve a specific task and the basis for genetic optimization. In this paper, roles are represented by genes, and role model is represented by the individual.

Genetic evolution is a process in which the population gradually adapts to the living environment and the quality is continuously improved. It includes selection operators, crossover operators, and mutation operators. In the process of genetic evolution, the parent's selection operator is to select individuals in the population and is the seed of the next generation of reproduction. In general, individuals with better fitness will be more likely to be selected, thereby further enhancing the ability of the population. However, individuals with poor fitness also have the opportunity to be selected, and their genes will have the opportunity to be passed on to the next generation. This will avoid the search mechanism being too greedy and avoid falling into local optimal solutions. The crossover operator generates a new offspring by crossing the genes which are selected from the parent. The mutation operator is responsible for creating new individuals based on existing individuals in the current population, thereby discovering new search space. Every new individual is called an offspring or a new solution. The fitness function is used to determine the pros and cons of individuals in the population. The genetic algorithm flow is shown in Figure 1.

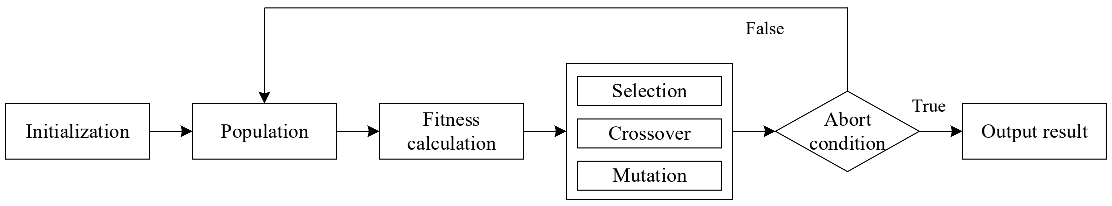

Figure 1. Genetic algorithm.

First, the initialization of population is performed. A certain number of individuals are randomly generated. Then the best individual is picked and placed in the initial population. This process is iterated until the number of individuals in the initial population reaches a predetermined scale. After that, the individual's fitness is calculated, which is specified according to the actual proximity of the problem solution. Then the next generation of populations is generated by the breeding process, which includes gene selection, crossover, and mutation. If the new generation population satisfies the abort condition, genetic algorithm is aborted. If the abort condition is not satisfied, new generation population is iteratively calculated until the abort condition is met. Finally, we output the final result.

4. Role Evolution Mechanism

The structure of role evolution mechanism is shown in Figure 2, which includes the static role mining and dynamic role reconstruction. The input of role mining is the user-permission relationship UPM. After the role is initialized, the initial role population R_POP_{sta} is obtained. The input of role reconstruction is the existing role model (U_{cur}, R_{cur}, P_{cur}, URA_{cur}, PRA_{cur}) in the system, and the current role population R_POP_{dyn} is obtained through the role encoding. R_POP_{sta} or R_POP_{dyn} is input into the role evolution process, and the new role set $RoleModel_{post}$ is obtained after the pre-processing algorithm, role evolution method based on genetic algorithm and post-processing algorithm. We input $RoleModel_{post}$ into the role application process, and implement access control on data resources after performance evaluation and role relationship assignment. The core work of this paper is the gray background part of the Figure 2.

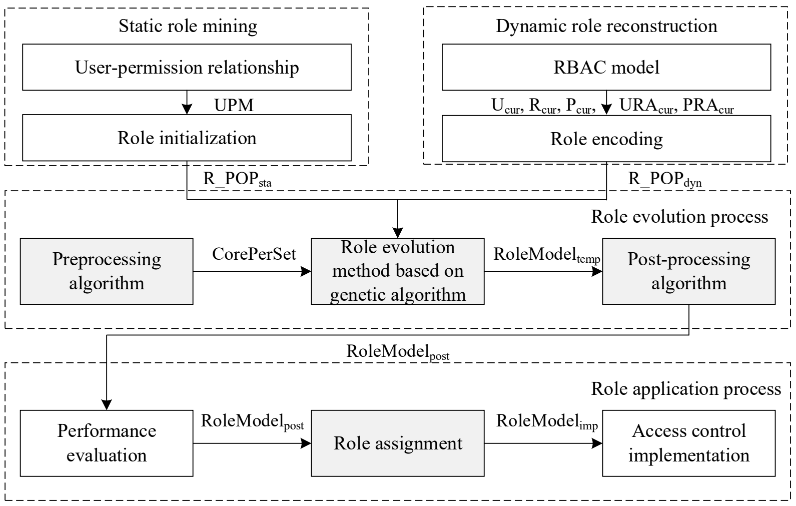

Figure 2. Structure of role evolution.

The core algorithms in role mining and role reconstruction are the same which are all role evolution method based on genetic algorithm. However, the starting point for their evolution is different. For role mining, the starting point of evolution is a randomly initialized role set. For role reconstruction, the starting point of evolution is the existing role set in the system. Therefore, role evolution is not only applied to the initialization of role model, but to the full lifecycle of role management throughout the access control run phase. The role set is periodically evolved according to the change of the current system's permission status, so that the role set is continuously optimized. There are time series relationships among role sets, as shown in Figure 3.

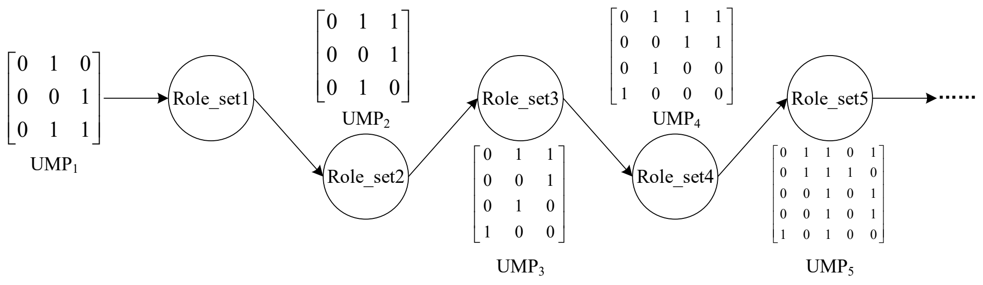

Figure 3. Temporal relationship of role evolution.

5. Role Evolution Method Based on Genetic Algorithm

This section describes the important concepts and algorithm which are involved in the role evolution method. It includes core permission evaluation, encoding and decoding of role genes, calculation operators and role optimization and so on.

5.1. Core Permission Evaluation

The lack of core permissions will prevent the proper running of the business system. The goal of role evolution is intended to cover as many user-permission relationships as possible. For a permission, during the role evolution process, if the quantity of users who have the permission is small, the possibility that the permission is discarded will be greater. In general, the importance of permissions is inversely proportional to the coverage of permissions in access control systems. The more important the permission, the fewer users will have the permission so that the permission is not misused. For the permission owned by many users, on the one hand, the significance of this permission is relatively low. On the other hand, this permission is hard to lose during role evolution. Even if one user does not get the permission, many other users do, thus ensuring the availability of the system. Therefore, the goal of the core permission evaluation is to find those permissions that cover fewer users and are more easily discarded. We evaluate the permissions by permission structure complexity (PSC) to generate the core permissions. The calculation method of PSC is shown in Equation (1), where $\mathrm{m_{i,j}}$ is the value of the i-th row and the j-th column in the user-permission relationship UPM of $f \times l$. $\alpha 1$ and $\alpha 2$ are weights, and *thd* is a complexity threshold. When the PSC of permission $\mathrm{p_j}$ is less than *thd*, we consider $\mathrm{p_j}$ to be the core permission.

$$PSC(p_j) = \sum\nolimits_{i=1}^{f} (\alpha_1 \cdot m_{i,j} + \alpha_2 \cdot \sum\nolimits_{k=1}^{l} m_{i,j} \cdot m_{i,k}), \quad (1)$$

where $\sum_{i=1}^{f} m_{i,j}$ is the number of users with permission p_j. The smaller the value, the less users have permission p_j. We think that the occurrences number of this permission is small and the more important. $\sum_{i=1}^{f} \sum_{k=1}^{l} m_{i,j} \cdot m_{i,k}$ is the number of permissions that are owned by users with permissions p_j. The smaller its value, the greater the proportion of the permission p_j, the more important. Therefore, the lower the *PSC*, we consider the permission is the more important.

5.2. Encoding and Decoding of Role Genes

Encoding is the mapping of an individual from phenotype to genotype, which transforms the external representation of an individual into a genetic feature. Decoding is the mapping of an individual from genotype to phenotype, which transforms the individual's genetic feature into an external representation. A two-dimensional array [UR, RP] is used to encode the role genes, UR is the granted user-role relationship, RP is the granted role-permission relationship. The role gene is a basic unit of the role model which represents a role, as shown in Figure 4.

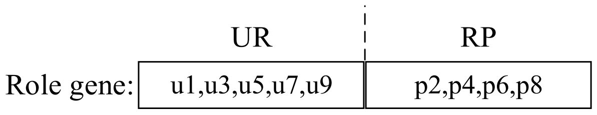

Figure 4. Role gene.

5.3. Selection, Ccrossover and Mutation of Role Genes

The operators include selection operator, crossover operator and mutation operator in the role evolution method.

(1) Selection operator: Select adaptive individuals from the population to produce the next generation. After several generations of evolution, the differences among individual chromosomes will reduce, that can make the population lose the diversity of individuals. In order to solve the problem,

we use the Roulette Wheel Selection method to randomly select the individuals to be combined. The basic idea is that the selected probability of each individual is proportional to the size of its fitness. The calculation method is shown in Equation (2).

$$P(k) = \frac{f(x_k)}{\sum_{m=1}^{n} f(x_m)}, \tag{2}$$

where $f(x_k)$ is the fitness of the k-th individual, and P(k) is the probability that the k-th individual is selected by the selection operator.

(2) Crossover operator: It includes single point crossover and multiple point crossover, as shown in Figure 5. Single point crossover randomly selects a crossover point. Then genes are exchanged between chromosomes which are located in front and back of the crossover point. Finally, the new offspring will be generated. For example, in Figure 5a, single point crossover occurs in point A. The positions of R_{14} and R_{15} are interchanged with the positions of R_{24} and R_{25} as a whole.

Multiple point crossover requires multiple crossover point to be set. The exchange of genes between chromosomes is performed front and back multiple crossover point to generate new offspring. For example, in Figure 5b, multiple point crossover occurs in point B and point C, so the sequence of corresponding gene can be changed.

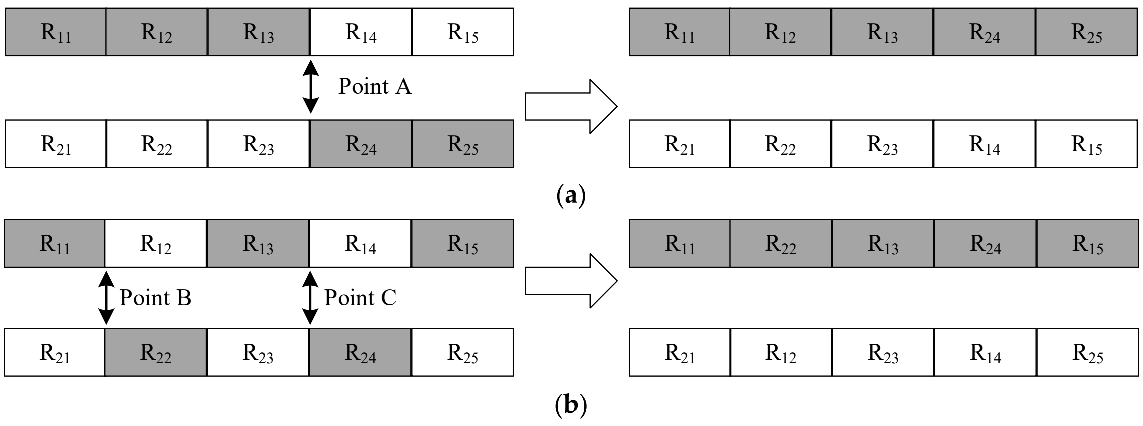

Figure 5. (**a**) Single point crossover. (**b**) Multiple point crossover.

(3) Mutation operator: It includes chromosome-level mutation and gene-level mutation. Figure 6 shows chromosome-level mutation, which includes the increase of role genes and the deletion of role genes.

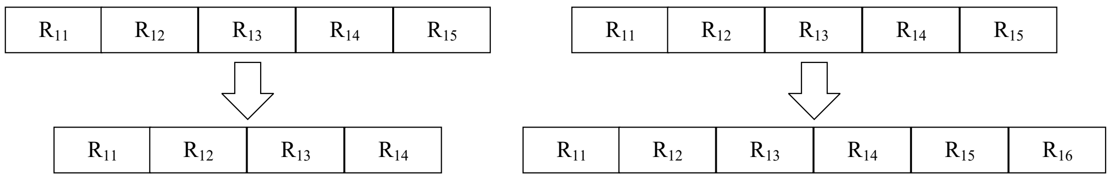

Figure 6. Chromosome-level mutation.

Figure 7 shows the gene-level mutation, which includes the increase, deletion and modification of both the user-role relationship and the role-permission relationship.

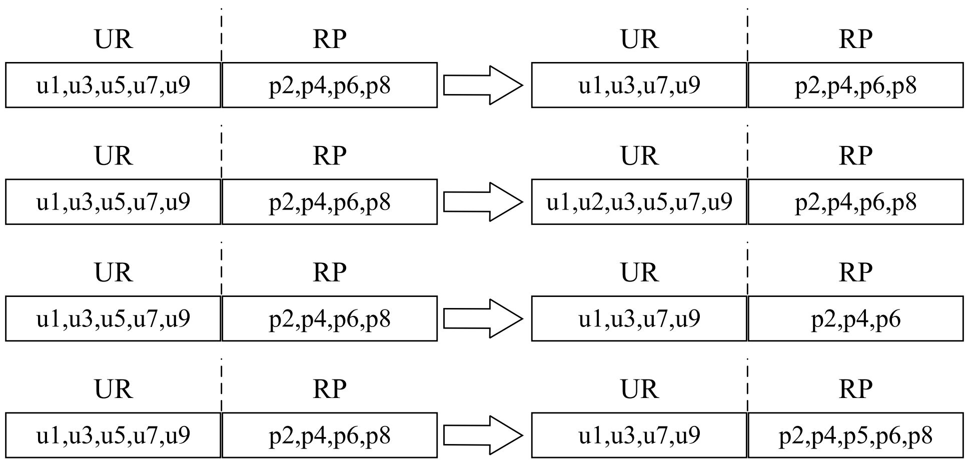

Figure 7. Gene-level mutation.

5.4. Evaluation Indicators and Fitness Calculation

The evaluation indicators include the number of obtained roles |R|, the number of user-role assignment relationships |UR|, the number of role-permission assignment relationships |RP|, the accuracy of role evolution Pe, the confidentiality indicator CI, and the availability indicator AI.

As shown in Equations (3) and (4), the accuracy of role evolution (Pe) is used to evaluate the degree of consistency between the new obtained user-permission relationship by the mapping of evolved user-role-permission relationship and the user-permission relationship before evolution.

$$EM = UP_{new} - UP_{old}, \tag{3}$$

$$Pe = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} (N_{EM(i,j)=0}), \tag{4}$$

UP_{new} is the new user-permission relationship matrix, and UP_{old} is the original user-permission relationship matrix. The bigger Pe is, the better the algorithm performance is.

The confidentiality indicator (CI) (as shown in Equation (5)) is used to determine whether a permission leak has occurred. When a permission does not exist in the original user-permission relationship, instead existing in the new user-permission relationship, we believe that the permission leak has occurred.

$$CI = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} (N_{EM(i,j)=1}), \tag{5}$$

$N_{EM(i,j)=1}$ is the number of leaked permissions. The smaller the CI is, the better the algorithm performance is.

As shown in Equation (6), the availability indicator (AI) is used to evaluate the availability of evolutionary results. When a permission exists in the original user-permission relationship, and does not exist in the new user-permission relationship, we believe that the permission is discarded. The availability of system is affected. The smaller the AI is, the better the algorithm performance is.

$$AI = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} (N_{EM(i,j)=-1}), \tag{6}$$

Different evaluation indicators can evaluate the effects of role evolution from different dimensions. In fact, role evolution problem is a multi-objective optimization problem. Based on the importance of different evaluation indicators, we use the weight coefficient method to set the weight value ω_i for each sub-goal, and the linear weighted summation of each sub-object is the fitness function. As shown in Equation (7), the multi-objective optimization problem is transformed into a single-objective optimization problem, which is a minimization problem.

$$f = \omega_1 \cdot |R| + \omega_2 \cdot |UR| + \omega_3 \cdot |PR| + \omega_4 \cdot \frac{1}{Pm} + \omega_5 \cdot CI + \omega_6 \cdot AI, \tag{7}$$

Through the adjustment of the parameters, relevant evolution parameter ω_i can be optimized according to the intention of security administrator, so that the role evolution is more targeted and the effect of the access control role management is improved.

5.5. Role Optimization

There may be redundant roles in the evolved roles. Redundant roles are deleted through role consolidation which reduces the number of roles, as shown in Equation (8).

$$R_1 = [u\_set_1, p\_set_1], R_2 = [u\_set_2, p\_set_2], \tag{8}$$

when $u\_set_1 = u\_set_2$ or $p\_set_1 = p\_set_2$, R_1 and R_2 can be combined. When $u\_set_1 \subseteq u\_set_2$ and $p\_set_1 \subseteq p\_set_2$, R_1 can be deleted.

5.6. Description of Algorithm

Role evolution process includes three core algorithms, which are preprocessing algorithm, role evolution algorithm and post-processing algorithm (as shown in Figure 2).

(1) Preprocessing algorithm is used to evaluate the importance degree of the permission, generate the core permission, and initialize the population of role model. The preprocessing algorithm consists of two core steps, namely the population initialization of the role model and the calculation of the core permission. The way to initialize the population of role model is to generate InitNum role genes by random method, and the corresponding user and permission codes of each role gene are also assigned randomly. The core permissions are calculated by calculating the PSC value (shown in Equation (1)) of the permissions. The pseudo code of the algorithm is shown in Algorithm 1, as follows.

Algorithm 1 Preprocessing algorithm

```
INPUT: UPM, Rmax (Role size), θ (Initialization index),
    threshold(Complexity threshold)
OUTPUT: InitRoleModel(Initial role model),
    CorePerSet(Core permission set)
1: InitNum = Random(θ•Rmax, Rmax), InitRoleModel = null
2: for I = 1 to InitNum do
3:     TempRole = null
4:     for j = 1 to Random (1,UPM.UserNum) do
5:         TempRole.Users.append(Random (1,UPM.UserNum))
6:     for k = 1 to Random (1,UPM.PermNum) do
7:         TempRole.Perms.append(Random (1,UPM.PermNum))
8:     InitRoleModel.append(TempRole)
9: for k = 1 to UPM.PermNum do
10:     index = PSC(k)
11:     if (index> threshold)
12:         CorePerSet.append(k)
```

(2) The role evolution algorithm uses the genetic algorithm to solve the problem. The genetic algorithm has strong global search ability, which can guarantee the high quality of the final role results. Moreover, the generated role set has the ability to dynamically evolve based on changes of user-permission relationships in an open computing environment. It can implement the role mining

and the role reconstruction. The algorithm includes the selection, crossover, mutation and individual fitness calculation of the role gene (Section 5.3 and Section 5.4). When the fitness meets the condition or reaches the upper limit of evolutionary generations, the algorithm will stop. The pseudo code for the algorithm is shown in Algorithm 2.

Algorithm 2 Role evolution algorithm

```
INPUT: n_pop(Population size), n_gen(Number of evolutions),
    $P_c$(Probability of gene crossover),
    $P_m$(Probability of gene mutation)
OUTPUT: RoleModel
1: Initialize pop, t = 0
2: do{ Calculate the fitness F(i) of each individual
3:     do{ $I_1$, $I_2$ = chose2indivial(pop)
4:         if (random(0, 1) < $P_c$):
5:             $I_{g1}$, $I_{g2}$ = crossChr($I_1$, $I_2$)
6:         else:
7:             $I_{g1}$, $I_{g2}$ = $I_1$, $I_2$
8:         if (random(0, 1) < $P_m$)
9:             $I_{g1}$, $I_{g2}$ = mutChr($I_{g1}$, $I_{g2}$)
10:         $pop_{t+1}$.append($I_{g1}$, $I_{g2}$)
11:    } while( len($pop_{t+1}$) < n_pop )
12:    pop = $pop_{t+1}$, t = t + 1
13: }while( F(pop.chr) < Fe and t < n_gen )
```

(3) The post-processing algorithm is used to implement role optimization, removes redundant roles (as shown in Equation (8)) and determines whether the role set includes the core permission. If the core permission is lost during the evolution process, the algorithm will add the core role to supplement the core permission. The core permissions are calculated by the Preprocessing algorithm. The pseudo code for the algorithm is shown in Algorithm 3.

Algorithm 3 Post-processing algorithm

```
INPUT: RoleModel, CorePerSet(Core permission set)
OUTPUT: FinalRoleModel()
1: for each $Role_i$∈RoleModel do
2:     for each $Role_j$∈RoleModel do
3:         if ($Role_i$.Users=$Role_j$.Users or $Role_i$. Perms=$Role_j$. Perms)
4:             RoleModel.merge($Role_i$, $Role_j$)
5:         else if ($Role_i$.Users ⊇ $Role_j$.Users and $Role_i$.Perms ⊇ $Role_j$. Perms)
6:             RoleModel.delete($Role_j$)
7: for each $CorePer_i$∈CorePerSet do
8:     flag = 0
9:     for each $Role_j$∈RoleModel do
10:    if($Role_j$ contains $CorePer_i$)
11:    flag = 1
12: if(flag == 0)
13: RoleModel.append($CorePer_i$)
```

6. Role Relationship Aggregation Algorithm

In order to correlate the mined role model with the real semantics in the real environment, the role information can be better applied in the access control system. Based on mean shift clustering algorithm, we cluster the roles and put similar roles in one category. Clustering is a class of roles with similar users and permissions, and there may be semantic correlation in the real working environment. Using the results of clustering, security administrators can be instructed to assign real semantics to a large number of roles based on the working environment. This will lay the foundation for assigning semantic information and subsequent applications of roles. The pseudo code for the algorithm is shown in Algorithm 4.

Algorithm 4 Role relationship aggregation algorithm

```
INPUT: RoleVecSet(Role vector set), radius
OUTPUT: RoleClu(Role clustering model)
1: Initialize RoleClu=null, countVec, tempData=null
2: for each RoleVeci∈RoleVecSet do
3:     clu_center = RoleVeci
4:     while True:
5:         for each RoleVecj∈RoleVecSet do
6:             if(RoleVecj - clu_center)<=radius
7:                 tempData.append(RoleVecj)
8:                 countVec[i] += 1
9:             new_cen = Average(tempData)
10:            if(RoleVeci.equal(new_cen))
11:                break
12:        new_clu=newCluster(countVec)
13:sameClu = False
14:for each Clui∈RoleClu do
15:    if((Clui .cen-new_cen)<=radius)
16:        combine(Clui, new_clu)
17:        sameClu = True
18:    if (has_same == Flase)
19:        RoleClu.append(new_clu)
```

The mean shift algorithm is a center-based clustering algorithm. It can be utilized to deal with the case where the number *k* of clusters is unknown. It is not necessary to set the number of clusters *k* in advance. The core idea is to calculate the average value M of the distance between a certain point A and its surrounding radius R, and calculate the direction of the next shift of the point (A = M + A). When the point no longer moves, it forms a cluster with the surrounding points, and calculates the distance between the cluster and other clusters. If the distance is less than the threshold D, they are merged into a cluster. If it is not satisfied, a new cluster is formed by itself until all the data points are selected. The results will establish the semantic relationship among roles, guide the generation of roles in the real environment, and help security administrator to give semantic meaning to the role group.

As can be seen from Section 5.2, encoding of role gene is shown in Figure 8. The role gene can be transformed into a fixed-length character vector to achieve role clustering. Role clustering is achieved by transforming role genes to fixed-length role vectors.

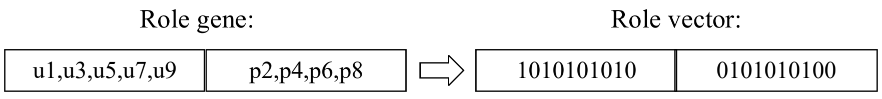

Figure 8. Role vector generation.

For a role vector v in a given vector space, the basic form of mean shift is:

$$G_r(x) = (y \big| \|y - x\|_d \le r), \tag{9}$$

$$M_r(v) = \frac{1}{k} \sum_{v_i \in G_r} (v_i - v), \tag{10}$$

G_r is a high-dimensional spherical space of radius r, defined as Equation (9). The k is the number of samples. Add all the vectors that are formed by all points and the center of the sphere in G_r and get the result of the mean shift vector $M_r(v)$, as shown in Equation (10).

7. Experimental Evaluations

7.1. Datasets and Experimental Settings

To verify the effectiveness of our method, we perform validation experiments based on six real access control data sets: Healthcare, Domino, Firewall1, Firewall2, America$_{\text{small}}$, and America$_{\text{large}}$. The dataset consists of users, permissions, and user-permission relationship. It is consistent with the access control structure that we set up in the application environment. The statistics about users and permissions in these data sets of access control are shown in Table 1 below. They are widely used in the literature [38,41,47,48] to analyze the performance of various role mining algorithms. By comparing the performance of algorithms in different data sets, the unstable evaluation results that are caused by a single data set can be effectively avoided, and the robustness of the method is tested.

Table 1. Experimental datasets.

Datasets	|U|	|P|	|UPM|	Density of |UPM|
Healthcare	46	46	1486	0.7023
Domino	79	231	730	0.0400
Firewall1	365	709	31,951	0.1235
Firewall2	325	590	36,428	0.1900
America-small	3477	1587	10,5205	0.0191
America-large	3485	10,127	18,5294	0.0053

|U| is the number of users, |P| is the number of permissions, |UPM| is the number of user-permission assignment relationship. The hardware and software environment of the experiment is as follows: the operating system is Win10 64-bit, the CPU is Intel(R) Core(TM) i7-8750H@2.21 GHz, the GPU is GeForce GTX 1050 Ti Max-Q, the memory size is 16 GB, and the Python version is 3.6.

7.2. Performance Evaluation of Role Evolution

The role evolution process has been carried out 800 generations of evolutionary training for the four policy sets of Healthcare, Domino, Firewall1, and Firewall2. The parameters $\omega1$–$\omega6$ in the fitness function (Equation (7)) are set to [1, 1, 1, 1, 1, 1]. We assume by default that these six evaluation indicators are equally important to the system. The population size n_pop is set to 100. The number of evolution generations n_gen is set to 800. Probability of gene crossover P_c is set to 0.6. Probability of gene mutation P_m is set to 0.35.

As shown in Figure 9a, the fitness of role evolution algorithm is continuously reduced, and the effectiveness of the proposed method is verified. The heuristic algorithm is effectively converged in the evolution process. At the same time, we test the average time cost per generation in different data sets. As shown in Figure 9b, the average time cost per generation of four data sets (Healthcare, Domino, Firewall1, and Firewall2) are 0.3918 s, 1.7925 s, 6.0236 s, and 4.7048 s, respectively. As the number of users and permissions increases, so does the time cost.

To evaluate the role evolution performance from multiple dimensions, our experiments evaluate the role evolution performance of different policy sets in different generations (as shown in Figure 10a–f). The evaluation criteria include six dimensions of final results: the role evolution accuracy Pe, the confidentiality indicator CI, the availability indicator AI, the number of roles |R|, the number of user-role assignment relationships |UR|, the number of role-permission assignment relationships |RP|.

To achieve better experimental convergence efficiency, we standardized these six indicators during the evolution process and mapped them to the [0, 1] intervals to track the changes of different indicators.

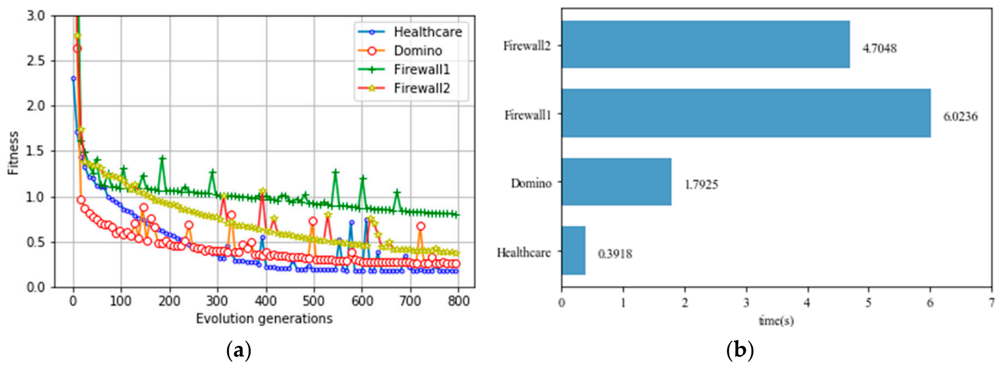

Figure 9. (**a**) Fitness of different generations. (**b**) Average time cost per generation.

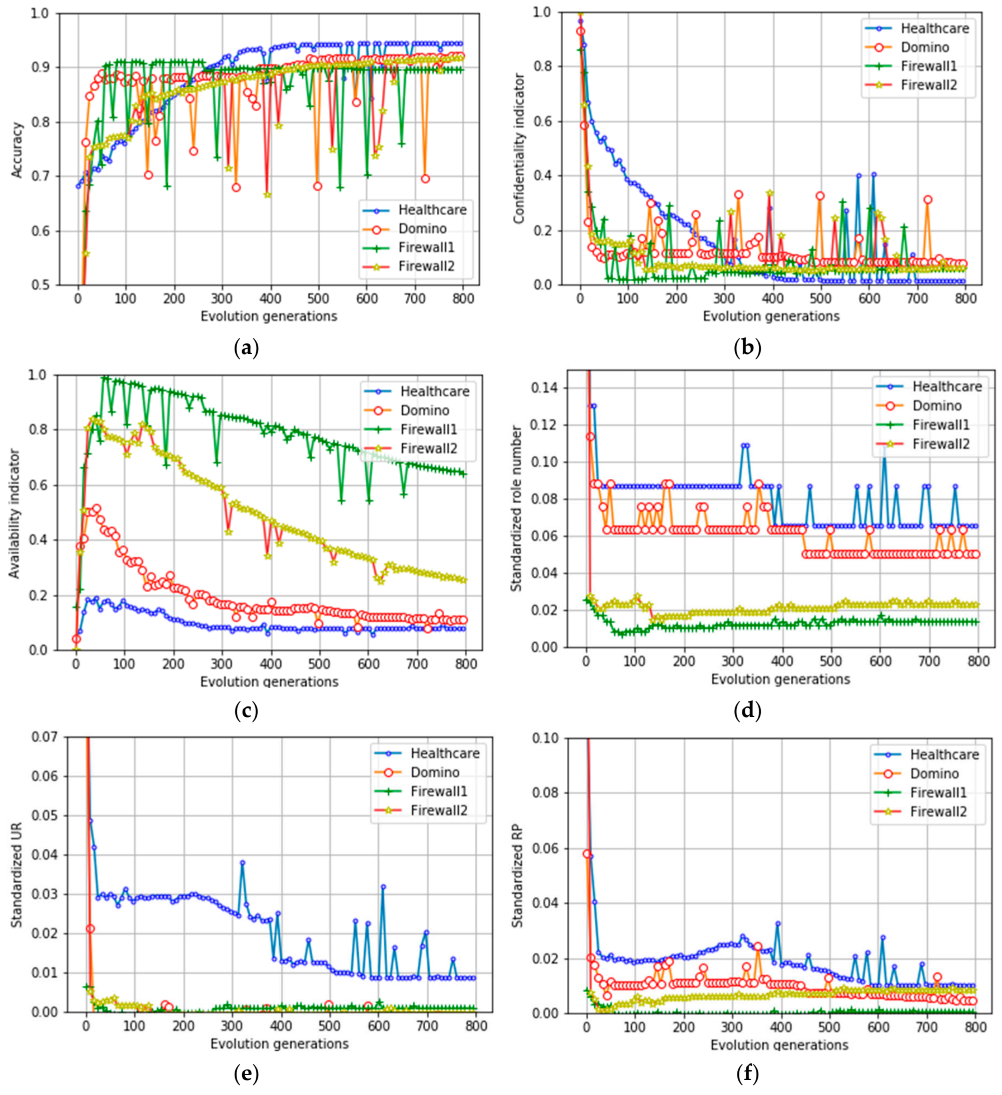

Figure 10. (**a**) Accuracy of different generations. (**b**) Confidentiality indicator of different generations. (**c**) Availability indicator of different generations. (**d**) Number of roles of different generations. (**e**) Number of user-role assignment relationships in different generations. (**f**) Number of role-permission assignment relationships in different generations.

As shown in Figure 10a, the experimental results show that the accuracy of role Evolution Pe for the four different policy sets can reach more than 90% within 800 generations. The resulting role model can cover most user-permission relationships. As shown in Figure 10b, the final confidentiality indicators in all four datasets can maintain low levels. Further, the gradual decrease of confidentiality indicator shows that the permission leakage of the role model is significantly reduced during the evolution process, and the system security is significantly increased. As shown in Figure 10c, the final availability indicators in the four datasets can maintain low level, which makes the role model of the system more available. However, availability indicators show a trend of first rising and then falling in the process of role evolution. In the early stage of role evolution process, due to the significant decrease of confidentiality indicators, the role of system will be granted too few permissions, which will lead to the increase of the permission loss in the role model (that is that the availability of the system is decreased). However, after 80 generations of evolution, the system has been able to significantly optimize the availability indicators so that the loss of permissions is reduced (that is that the availability of the system is increased). At the same time, the number of roles, user-role assignment relationships, and role-permission assignment relationships have also dropped significantly during the process (as shown in Figure 10d–f).

In addition, we compare the performance of our method with the five general algorithms of CRM [34], PC [35], CM [36], GO [37], and HM [38]. The experimental results are shown in Table 2. It can be seen from Table 2 that compared with other general methods, our method can more effectively compress the role scale under the premise of allowing certain evolution errors, and has better effects. It can effectively reduce the role management burden of security administrator and improve permission management efficiency of access control system. CRM and GO also have good performance in some data sets. This is because CRM adds some constraints. These constraints limit the number of permissions that a role can contain. GO uses graph optimization theory to avoid the backtracking search process of permission in other methods. However, these two methods only consider a single constraint index, and do not fully consider the security and availability of the system. There is a risk that the system will not work properly. In addition, when the user, permission, and user-permission relationships in the system change, it is necessary to start role mining from the beginning, which adds extra overhead and work. The old results of role mining are not used effectively.

Table 2. Number of roles of different algorithms.

Datasets	CRM	PC	CM	GO	HM	Our Method
Healthcare	14	24	31	16	17	13
Domino	20	64	62	20	27	20
Firewall1	66	248	278	71	91	61
Firewall2	10	14	21	10	10	9

7.3. Performance Evaluation of Role Relationship Aggregation

We have two role sets RoleSet1 and RoleSet2 which are generated based on the datasets America$_{small}$ and America$_{large}$ (the role size is around 3000). The two large-scale role sets are clustered by using role relationship aggregation algorithm, and the clustering results are mapped to the 2-dimensional space by using principal component analysis (PCA) [49,50]. The experimental results are shown in Figure 11a,b. The roles in RoleSet1 are clustered into five categories, and the roles in RoleSet2 are clustered into fifteen categories. We can directly understand the relationship between the roles by visualization. Since the aggregation algorithm does not need to set the clustering value, the automatic generation of the role clustering category can be realized. Further, the roles with similar users and permissions relationships can be effectively clustered. This result will assist the role to obtain real semantic information and improve the efficiency of security administrators to implement access control management.

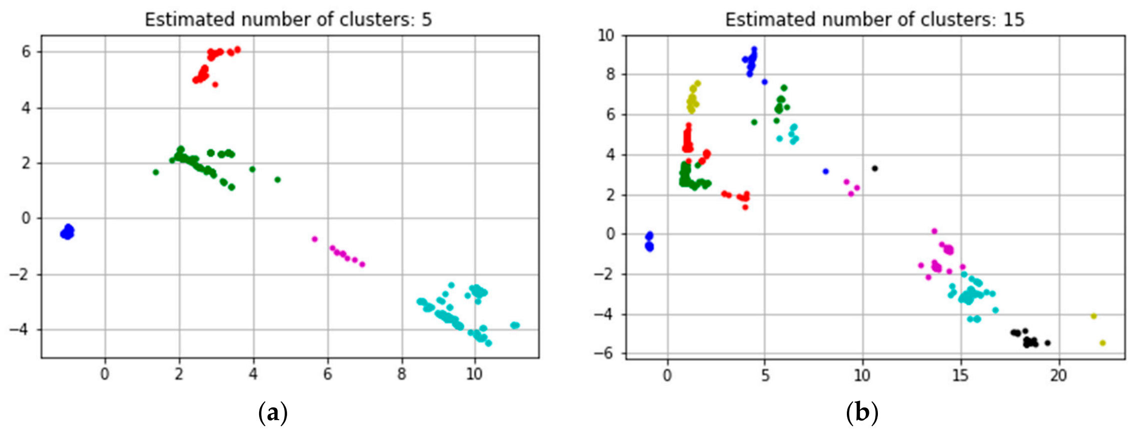

Figure 11. (**a**) Role clustering of America$_{\text{small}}$; (**b**) role clustering of America$_{\text{large}}$.

8. Conclusions

To meet the dynamic evolution needs of the access control role for open computing environment, this paper proposes a role evolution mechanism based on genetic algorithm, which can automatically implement the role mining and reconstruction. Furthermore, the mechanism can provide role support for intelligent and automated access control. The role evolution is optimized from multiple dimensions when taking into account the security and availability of resources. By introducing the evaluation of core permission, it can effectively avoid the loss of core permission in the process of role evolution. Moreover, the role relationship aggregation algorithm is used to implement the clustering of roles, which is an auxiliary means for giving real semantics to the roles. As a result, it improves the efficiency of the security administrator to implement access control management. In the future, we will optimize the performance and efficiency of role evolution to further improve accuracy and reduce the error.

Author Contributions: Conceptualization, A.L., X.D. and N.W.; methodology, A.L., X.D. and N.W.; validation, A.L., X.D.; formal analysis, A.L; investigation, A.L., X.D. and N.W.; writing—original draft preparation, A.L., X.D. and N.W.; writing—review and editing, A.L., X.D. and N.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key Research and Development Program of China (No. 2018YFB0803603, No. 2016YFB0501901), and is also supported by National Natural Science Foundation of China (No. 61802436, No. 61902447).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wu, X.; Zhu, X.; Wu, G.; Ding, W. Data mining with big data. *IEEE Trans. Knowl. Data Eng.* **2013**, *26*, 97–107.
2. Li, S.; Da Xu, L.; Zhao, S. The internet of things: A survey. *Inf. Syst. Front.* **2014**, *17*, 243–259. [CrossRef]
3. Barua, H.B.; Mondal, K.C. A Comprehensive Survey on Cloud Data Mining (CDM) Frameworks and Algorithms. *ACM Comput. Surv.* **2019**, *52*, 1–62. [CrossRef]
4. O'Halloran, K.L.; Tan, S.; Wignell, P.; Bateman, J.; Pham, S.; Grossman, M.; Moere, A.V. Interpreting Text and Image Relations in Violent Extremist Discourse: A Mixed Methods Approach for Big Data Analytics. *Terror. Politi-Violence* **2016**, *31*, 454–474. [CrossRef]
5. Data Breaches Compromised 4.5 Billion Records in First Half of 2018. Available online: https://www.gemalto.com/press/Pages/Data-Breaches-Compromised-4-5-Billion-Records-in-First-Half-of-2018.aspx (accessed on 23 October 2018).
6. Lazouski, A.; Martinelli, F.; Mori, P. Usage control in computer security: A survey. *Comput. Sci. Rev.* **2010**, *4*, 81–99. [CrossRef]
7. Power, D.; Slaymaker, M.; Simpson, A. On Formalizing and Normalizing Role-Based Access Control Systems. *Comput. J.* **2008**, *52*, 305–325. [CrossRef]
8. Wang, G.H. Role-Based Access Control. *Computer* **1996**, *29*, 38–47.

9. Vaidya, J.; Atluri, V.; Warner, J.; Guo, Q. Role Engineering via Prioritized Subset Enumeration. *IEEE Trans. Dependable Secur. Comput.* **2008**, *7*, 300–314. [CrossRef]
10. Baumgrass, A.; Strembeck, M. Bridging the gap between role mining and role engineering via migration guides. *Inf. Secur. Tech. Rep.* **2013**, *17*, 148–172. [CrossRef]
11. Coyne, E.J.; Davis, J.M. *Role Engineering for Enterprise Security Management*; Artech House: Norwood, MA, USA, 2007.
12. Fang, L.; Yin, L.H.; Guo, Y.C.; Fang, B.X. A Survey of Key Technologies in Attribute-Based Access Control Scheme. *Chin. J. Comput.* **2017**, *40*, 1680–1698.
13. Li, H.; Zhang, M.; Feng, D.G.; Hui, Z. Research on Access Control of Big Data. *Chin. J. Comput.* **2017**, *1*, 72–91.
14. Liu, A.D.; Du, X.H.; Wang, N.; Li, S.Z. A blockchain-based access control mechanism for big data. *J. Softw.* **2019**, *9*, 2636–3654.
15. Hui, Z.; Li, H.; Zhang, M.; Feng, D.G. Risk-adaptive access control model for big data in healthcare. *J. Commun.* **2015**, *36*, 190–199.
16. Strembeck, M. Scenario-Driven Role Engineering. *IEEE Secur. Priv. Mag.* **2010**, *8*, 28–35. [CrossRef]
17. Kuhlmann, M.; Shohat, D.; Schimpf, G. Role mining—Revealing business roles for security administration using data mining technology. In Proceedings of the Eighth Acm Symposium on Access Control Models & Technologies, Huhehaote, China, 10–13 October 2003.
18. Mitra, B.; Sural, S.; Vaidya, J.; Atluri, V. A Survey of Role Mining. *ACM Comput. Surv.* **2016**, *48*, 1–37. [CrossRef]
19. Vaidya, J.; Atluri, V.; Guo, Q.; Lu, H. Role Mining in the Presence of Noise. In *DBSec'10: Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy, Rome, Italy, 21–23 June 2010*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6166, pp. 97–112.
20. Vaidya, J.; Atluri, V.; Guo, Q. The role mining problem: A formal perspective. *ACM Trans. Inf. Syst. Secur.* **2010**, *13*, 27. [CrossRef]
21. Huang, H.; Shang, F.; Zhang, J. Approximation Algorithms for Minimizing the Number of Roles and Administrative Assignments in RBAC. In Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, Izmir, Turkey, 16–20 July 2012; pp. 427–432.
22. Mitra, B.; Sural, S.; Atluri, V.; Vaidya, J. The generalized temporal role mining problem. *J. Comput. Secur.* **2015**, *23*, 31–58. [CrossRef]
23. Lu, H.; Vaidya, J.; Atluri, V. An optimization framework for role mining. *J. Comput. Secur.* **2014**, *22*, 1–31. [CrossRef]
24. Sarana, P.; Roy, A.; Sural, S.; Vaidya, J.; Atluri, V. Role Mining in the Presence of Separation of Duty Constraints. In *ICISS 2015: Proceedings of the 11th International Conference on Information Systems Security, Kolkata, India, 16–20 December 2015*; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9478, pp. 98–117.
25. Zhang, L.; Zhang, H.L.; Han, D.J.; Shen, X.J. Theory and Algorithm for Roles Minization Problem in RBAC Based on Concept Lattice. *Acta Electron. Sin.* **2014**, *42*, 2371–2378.
26. Zhou, C.; Ren, Z.Y.; Wu, W.C. Semantic Roles Mining Algorithms Based on Formal Concept Analysis. *Comput. Sci.* **2018**, *45*, 118–129.
27. Dong, L.; Wang, Y.; Liu, R.; Pi, B.; Wu, L. Toward edge minability for role mining in bipartite networks. *Phys. A Stat. Mech. Its Appl.* **2016**, *462*, 274–286. [CrossRef]
28. Vavilis, S.; Egner, A.I.; Petkovic, M.; Zannone, N. Role Mining with Missing Values. In Proceedings of the 2016 11th International Conference on Availability, Reliability and Security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 167–176.
29. Harika, P.; Nagajyothi, M.; John, J.C.; Sural, S.; Vaidya, J.; Atluri, V. Meeting Cardinality Constraints in Role Mining. *IEEE Trans. Dependable Secur. Comput.* **2014**, *12*, 71–84. [CrossRef]
30. Mitra, B.; Sural, S.; Vaidya, J.; Atluri, V. Mining temporal roles using many-valued concepts. *Comput. Secur.* **2016**, *60*, 79–94. [CrossRef]
31. Stoller, S.D.; Bui, T. Mining hierarchical temporal roles with multiple metrics. *J. Comput. Secur.* **2017**, *26*, 121–142. [CrossRef]
32. Narouei, M.; Takabi, H. Towards an Automatic Top-down Role Engineering Approach Using Natural Language Processing Techniques. In *SACMAT'15: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, Vienna, Austria, 1–3 June 2015*; Association for Computing Machinery: New York, NY, USA, 2015; pp. 157–160.

33. Kumar, R.; Sural, S.; Gupta, A. Mining RBAC Roles under Cardinality Constraint. In *ICISS'10: Proceedings of the 6th International Conference on Information Systems Security, Gandhinaga, India, 15 December 2010*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6503, pp. 171–185.
34. Molloy, I.; Li, N.; Li, T.; Mao, Z.; Wang, Q.; Lobo, J. Evaluating role mining algorithms. In Proceedings of the 14th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks—PE-WASUN '17, Bodrum, Turkey, 17–18 October 2010; pp. 95–104.
35. Vaidya, J.; Atluri, V.; Warner, J. RoleMiner: Mining roles using subset enumeration. In *CCS'06: Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006*; Association for Computing Machinery: New York, NY, USA, 2006; pp. 144–153.
36. Zhang, D.; Ramamohanarao, K.; Ebringer, T. Role engineering using graph optimisation. In *SACMAT'07: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, Sophia Antipolis, France, 20–22 June, 2007*; Association for Computing Machinery: New York, NY, USA, 2007; pp. 139–144.
37. Molloy, I.; Chen, H.; Li, T.; Wang, Q.; Li, N.; Bertino, E.; Calo, S.B.; Lobo, J. Mining roles with semantic meanings. In *SACMAT'08: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, Estes Park, CO, USA, 11–13 June 2008*; Association for Computing Machinery: New York, NY, USA, 2008; pp. 21–30.
38. Dong, L.; Wu, K.; Tang, G. A Data-Centric Approach to Quality Estimation of Role Mining Results. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 2678–2692. [CrossRef]
39. Zhai, Z.G.; Wang, J.D.; Cao, Z.N.; Mao, Y.G. Hybrid Role Mining Methods with Minimal Perturbation. *J. Comput. Res. Dev.* **2013**, *50*, 951–960.
40. Kunz, M.; Fuchs, L.; Netter, M.; Pernul, G. How to Discover High-Quality Roles? A Survey and Dependency Analysis of Quality Criteria in Role Mining. *Commun. Comput. Inf. Sci.* **2015**, *576*, 49–67.
41. Blundo, C.; Cimato, S.; Siniscalchi, L. PRUCC-RM: Permission-Role-Usage Cardinality Constrained Role Mining. In Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, Italy, 4–8 July 2017; Volume 2, pp. 149–154.
42. Pan, N.; Zhu, Z.; He, L.; Sun, L. An efficiency approach for RBAC reconfiguration with minimal roles and perturbation. *Concurr. Comput. Pr. Exp.* **2017**, *30*, e4399. [CrossRef]
43. Han, L.I.; Zheng, S.; Chen, F. Research on Role Engineering of Legacy System. *J. Front. Comput. Sci. Technol.* **2017**.
44. Hachana, S.; Cuppens, F.; Cuppens-Boulahia, N.; Garcia-Alfaro, J. Semantic analysis of role mining results and shadowed roles detection. *Inf. Secur. Tech. Rep.* **2013**, *17*, 131–147. [CrossRef]
45. Saenko, I.; Kotenko, I. Administrating role-based access control by genetic algorithms. In *GECCO'17: Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany, 15–19 July, 2017*; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1463–1470.
46. Saenko, I.; Kotenko, I. Genetic algorithms for role mining in critical infrastructure data spaces. In *GECCP'18" Proceedings of the Genetic and Evolutionary Computation Conference Companion, Kyoto, Japan, 15–19 July 2018*; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1688–1695.
47. Wu, L.; Dong, L.; Wang, Y.; Zhang, F.; Lee, V.E.; Kang, X.; Liang, Q. Uniform-scale assessment of role minimization in bipartite networks and its application to access control. *Phys. A Stat. Mech. Its Appl.* **2018**, *507*, 381–397. [CrossRef]
48. Xu, Z.; Stoller, S.D. Algorithms for mining meaningful roles. In *SACMAT'12: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, Newark, NJ, USA, 20–22 June 2012*; Association for Computing Machinery: New York, NY, USA, 2012; pp. 57–66.
49. Jolliffe, I.T. *Principal Component Analysis*, 2nd ed.; Springer: New York, NY, USA, 2002.
50. Yousefi, B.; Sfarra, S.; Ibarra-Castanedo, C.; Maldague, X.P. Comparative analysis on thermal non-destructive testing imagery applying Candid Covariance-Free Incremental Principal Component Thermography (CCIPCT). *Infrared Phys. Technol.* **2017**, *85*, 163–169. [CrossRef]