Role-Engineering Optimization with User-Oriented Cardinality Constraints in Role-Based Access Control

Wei Sun, Xiaoya Yuan, and Hui Su

(Corresponding author: Wei Sun)

Center of Network Information and Computing, Xinyang Normal University

237 Nanhu Road, Shihe District, Xinyang, Henan 464000, China

Email: sunny810715@xynu.edu.cn

(Received Dec. 4, 2020; Revised and Accepted May 6, 2021; First Online Aug. 14, 2021)

Abstract

Role-based access control (RBAC) is a popular security mechanism used by many organizations because it provides various constraint policies, such as cardinality constraints and separation-of-duty constraints. Role-engineering technology is an effective method for constructing RBAC systems. However, the mining scales are large, and the management burdens of systems are weighty. Furthermore, conventional role-engineering methods do not consider cardinality constraints. To address these issues, this paper proposes a novel method, called role-engineering optimization, with user-oriented cardinality constraints on roles (REO_UCCR). First, to reduce the mining scales and alleviate the management burdens of systems, we convert basic role mining into a clustering problem using the Hamming distance technique and four tuples of clusters. Then, we implement role mining while constructing an unconstrained role engineering system. Second, to verify whether the given cardinality constraints can be satisfied in the constructed system, we present a role optimization algorithm to reconstruct a constrained RBAC system. The experiments using synthetic and real datasets demonstrate the effectiveness of the proposed method and show encouraging results.

Keywords: Role Engineering; Role Mining; Role Optimization; User-Oriented Cardinality Constraints

1 Introduction

With the rapid development and comprehensive application of network information technology, a large amount of information storage and exchanges are required in large-scale and complex information-management systems [11, 22]. An increasing number of enterprises and organizations have adopted role-based access control (RBAC) as their main access-control mechanism over the last three decades, as it makes security administration more flexible and manageable [2, 7, 16, 17]. With the successful implementation of RBAC systems, devising an accurate and effective set of roles and constructing a good RBAC system, which can satisfy actual application requirements, have become critical tasks. The bottom-up role-engineering technology [1, 6, 14] aims to migrate from non-RBAC systems to RBAC systems. It starts from the original user-permission assignments and aggregates them into roles by applying data mining techniques, which is also known as role mining and has gained considerable attention in recent years.

In fact, role mining is the task of clustering users with identical or similar permissions and constructing different roles with these permissions [19]. Roles containing several identical permissions are frequently assigned to users. Usable roles can frequently facilitate the management and maintenance of the system and decompose the set of users into clusters of users with different attribute properties. To enhance the interpretability of role mining, it is indeed necessary to cluster users with the same attribute properties. However, due to the diversity of the attribute properties of entities and the variability of accesses, the mining scales are large, and the management burdens of systems are very heavy using conventional role-mining methods.

A key characteristic of RBAC is that it allows for the specification and enforcement of various types of security policies [15,18], such as separation-of-duty constraints and cardinality constraints, which reflect the security requirements of organizations and can ensure the security of RBAC systems. There are four different types of cardinality constraints among users, roles, and permissions, and they limit the maximum number of roles related to users or to permissions, the maximum number of permissions a role can have, or the maximum number of users to which a role can be assigned [12]. For example, the general-manager role in a company must be assigned to only one person, and ordinary users should not have too many roles; otherwise, there is the possibility that users will abuse their privileges. In terms of the approaches to the construction of role engineering, however, most of the existing methods cannot determine whether the given cardinality constraints are satisfied in a constructed RBAC system.

To address the abovementioned issues, this paper proposes a novel method, called role-engineering optimization, with user-oriented cardinality constraints on roles (REO_UCCR). In summary, the main contributions of this work are as follows:

1) To reduce the mining scales and alleviate the management burdens of systems, we adopt the Hamming distance technique to rearrange an original access matrix, generate user clusters, and then implement role mining, while constructing an unconstrained role engineering system.

2) To verify whether the given cardinality constraints can be satisfied in the constructed system, we first present the definition of the role-engineering optimization problem and then propose a role optimization algorithm to reconstruct a constrained RBAC system.

The rest of the paper is organized as follows. In Section 2, we discuss the related work and present some necessary preliminaries. In Section 3, we propose a novel research method and present several algorithms and running examples. We show the experimental evaluations in Section 4. Section 5 concludes the paper and discusses future work.

2 Related Work and Preliminaries

2.1 Methods of Role Engineering

To discover interesting roles in existing permission assignment relationships, two algorithms, called the Complete Miner and Fast Miner, were proposed [19]. Both the two algorithms use subset enumeration and allow for overlapping roles. While the first algorithm enumerates all potential roles, its computational complexity is exponential. The second algorithm improves the mining process, and its computational complexity is remarkably reduced. Vaidya *et al.* [20] converted role mining into a matrix decomposition problem and presented a definition of a basic role mining problem (basic RMP). The basic RMP has been proven to be NP-complete, and several existing studies have already been conducted to find efficient solutions. To avoid an abuse of privileges, Blundo *et al.* [3] proposed a heuristic capable of returning a complete set of roles, which limited the number of permissions assigned to a role. John *et al.* proposed two alternative approaches for restricting the number of roles assigned to a user: The role priority-based approach (RPA) and the coverage of permissions-based approach (CPA). The RPA prioritizes roles based on the number of permissions and assigns optimal roles to users, according to the priority order. The CPA chooses roles by iteratively picking the role with the largest number of permissions that are yet to be uncovered and then ensures that no user is assigned more than a given number of roles [9]. To simultaneously limit the maximum number of roles assigned to a user and a related permission, Harika *et al.* proposed two role-optimization methods: Post processing and concurrent processing. In the first method, roles are initially mined without taking the constraints into account. The user-role and role-permission assignments are then checked for constraint violation in the optimization process and appropriately re-assigned, if necessary [8]. The concurrent processing method implements optimization with double constraints during the process of role mining. Wang *et al.* [21] proposed two kinds of role mining algorithms in order to satisfy the permission cardinality constraints. The first algorithm discovered roles by decomposing a sorted access control matrix, and the second intersected the permissions of adjacent users in the access control matrix to generate candidate roles. Blundo *et al.* [4] focused on cardinality constraints, defined the constrained role mining problem for each constraint type, and presented efficient heuristics for these problems. In addition, to satisfy separation-of-duty constraints and ensure authorization security, we proposed a method, called role-mining optimization, with separation-of-duty constraints and security detection for authorizations [13].

Two main limitations are apparent in the existing studies. The first limitation is that the role-mining scales are very large, and the management burdens of systems are very heavy. The second limitation is that most conventional role-mining methods do not consider whether or not the number of roles related to a user is restricted. If the number of roles assigned to a user exceeds a particular value, then there is the possibility of an abuse of privileges, and the system is not secure. Hence, we propose a novel role-engineering optimization method in order to alleviate the management burdens, while ensuring the system security. We also evaluate the performance of the proposed method on the synthetic and real datasets.

2.2 Preliminaries

2.2.1 Basic Components of Role Engineering

According to the NIST standard of RBAC, conventional role engineering for RBAC consists of the following basic components:

1) U, P, and R are the basic elements of RBAC, which represent the sets of users, permissions, and roles, respectively;

2) $UPA \subseteq U \times P$ represents a many-to-many mapping relationship of user-permission assignments;

3) $URA \subseteq U \times R$ represents a many-to-many mapping relationship of user-role assignments;

4) $RPA \subseteq R \times P$ represents a many-to-many mapping relationship of role-permission assignments;

5) $user\_roles(u) = \{r|\exists r \in R : ((u, r) \in URA)\}$, which represents a set of roles assigned to user u;

6) $user\_perms(u) = \{p|\exists p \in P, \exists r \in R : ((u, r) \in URA) \wedge ((r, p) \in RPA)\}$, which represents a set of permissions assigned to user u.

2.2.2 The Basic RMP Problem and Fast Miner Method

The basic RMP [20] can be formally represented as follows:

$$\begin{cases} \min |R| \\ URA \otimes RPA = UPA. \end{cases} \tag{1}$$

For convenience, the UPA, URA, and RPA are used to represent their respective assignment relationships, as well as the corresponding matrices. The Fast Miner method [19] mainly consists of the following steps:

1) According to the hash mapping rule, a given access control matrix is converted into the user-permission assignment relationship;

2) To reduce the size of the original data set, different users who have the same permissions in the permission assignments are grouped together, and an initial set of roles is constructed;

3) All the potentially interesting roles are identified by implementing intersections between any pair of the initial roles. New roles are generated, and the number of users associated with any new role is counted.

2.2.3 Hamming Distance

Since the access control matrix, UPA, is a Boolean matrix, each row (or each column) can be regarded as a binary vector of the same length. The well-known technique of Hamming distance [5] is widely used to measure the distance between two different equal-length vectors. It states that given two equal-length Boolean vectors, x and y, the distance between x and y, denoted as $Dis(x, y)$, is the number of positions, where the vectors take different values for the same column position.

For instance, given two row vectors, $x =$ "100110" and $y =$ "110011", $Dis(x, y) = 3$. Clearly, the distance between any two rows in UPA increases as the number of column positions taking different values increases.

2.2.4 User-Oriented Cardinality Constraints on Roles (UCCR)

The UCCR [12] states that, given set U of users, set R of roles, and threshold MRC_{user}, the number of roles assigned to any user should not exceed MRC_{user}. This can be formalized as follows:

$$\forall u \in U : |user\_roles(u) \cap R| \leq MRC_{user} \tag{2}$$

In addition, there are another three cardinality constraints in RBAC, and they are not discussed in this work.

3 Proposed Method

In this section, we propose a novel research method, named REO_UCCR, which includes three aspects: 1) The generation of user clusters, 2) construction of unconstrained role engineering, and 3) role-engineering optimization with UCCR. Specifically, we adopt the Hamming distance technique to rearrange an original access matrix and generate user clusters in the preprocessing stage. Subsequently, we construct an unconstrained role engineering system in the role mining stage. Last, to verify whether the given cardinality constraints can be satisfied in the constructed system, we present a role optimization algorithm to reconstruct a constrained RBAC system. An overall view of the proposed framework is shown in Figure 1.

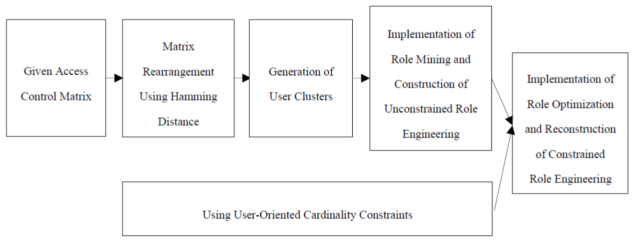

Given Access Control Matrix
Matrix Rearrangement Using Hamming Distance
Generation of User Clusters
Implementation of Role Mining and Construction of Unconstrained Role Engineering
Implementation of Role Optimization and Reconstruction of Constrained Role Engineering
Using User-Oriented Cardinality Constraints

Figure 1: Overview of the proposed role-engineering optimization framework

3.1 Generation of User Clusters

To intuitively represent the matrix, UPA, we use the Hamming distance to rearrange it, as defined below.

Definition 1. *(Matrix rearrangement problem with Hamming distance) Given an original matrix UPA, and a Hamming distance list D between any two rows of UPA, find a rearranged matrix UPA', such that the sum of distances between the adjacent rows of UPA' is minimal, which can be formalized as follows:*

$$\min(\sum_i Dis(UPA'[i], UPA'[i+1])),$$
$$\forall Dis(UPA'[i], UPA'[i+1]) \in D. \tag{3}$$

According to Definition 1, we present the process of matrix rearrangement in Algorithm 1.

```
Algorithm 1 Matrix rearrangement.
Input: original matrix $UPA$
Output: rearranged matrix $UPA'$
1: Initialize $UPA' = UPA$;
2: Represent $UPA'$ as a list of row vectors: $UPA'[1]$, $UPA'[2]$, ...;
3: Identify matrix $D_r$ of the Hamming distances between any two row vectors, such that
   $\forall i,j$: $D_r[i][j] = Dis(UPA'[i], UPA'[j])$;
4: for each $UPA'[i]$ in $UPA'$ do
5:   if ($\exists UPA'[j]$: $D_r[i][j] < D_r[i][i+1]$) then
6:     $swap(UPA'[i+1], UPA'[j])$;
7:   end if
8: end for
```

It can be seen, from Algorithm 1, that different users with the same permissions are grouped together, which can be regarded as a user group. To reduce the mining scale, we represent the groups as different user clusters and adopt four tuples to store them, as well as other properties. This is easy to implement, and we present its definition as follows.

Definition 2. *(Four tuples of user clusters) The users with the same permissions, as well as their properties, are group, which is denoted as a four-tuple form <c, user_perms(c), count_users(c), count_unperms(c)>, where c is a user cluster, C is a set of different clusters, user_perms(c) is the permission set associated with c, count_users(c) is the number of users included in c, and count_unperms(c) is the number of permissions uncovered recently in c.*

It is apparent that count_unperms(c) is equal to the value of $|user\_perms(c)|$, before role mining.

3.2 Construction of Unconstrained Role Engineering

To alleviate the management burdens of RBAC systems, it is necessary to make the cluster–permission assignments relationship as sparse as possible. Based on Definition 2, we choose the user cluster that involves the maximum number of users and regard it and its whole permission set as the candidate cluster and role, respectively. The construction process is presented in Algorithm 2.

In Algorithm 2, we first create and initialize several variables in Lines 1-4, including C', Cand_Roles, CRA, maxcount_users, and cand_cluster. For each cluster in C', we calculate the number of users derived from the cluster hierarchies and then identify the candidate cluster and its maximum number of users in Lines 5-15. Lines 16-18 update the Cand_Roles, CRA, and remove the candidate cluster. Next, for each cluster, we remove the permissions assigned to cand_cluster, which are covered by other clusters, and remove the clusters, when all of the permissions of those clusters have been covered by C' (Lines 19-28).

Algorithm 2. Construction of unconstrained role engineering.

```
Input: set C of the preprocessed results
Output: immediate cluster set C', candidate role set Cand_Roles, and cluster-role assignments CRA
1: Create and initialize C'= C;
2: Create and initialize Cand_Roles= Φ, and CRA= Φ;
3: Create and initialize maxcount_users=0, which represents the maximum number of
   users included in a cluster;
4: Create and initialize cand_cluster=Φ, which represents the cluster involving the
   maximum number of users;
5: for each c_i in C' do
6:    for each c_j in C' \ {c_i} do
7:       if user_perms(c_i) ⊆ user_perms(c_j) then
8:          count_users(c_i)+= count_users(c_j);
9:       end if
10:      if maxcount_users< count_users(c_i) then
11:         maxcount_users= count_users(c_i);
12:         cand_cluster= c_i;
13:      end if
14:   end for
15: end for
16: Cand_Roles= Cand_Roles ∪ {user_perms(cand_cluster)};
17: CRA= CRA ∪ {(cand_cluster, user_perms(cand_cluster))};
18: C'=C' \ {cand_cluster};
19: for (each c_k in C') ∧ (user_perms(cand_cluster) ⊆ user_perms(c_k)) do
20:   for each p in user_perms(cand_cluster) do
21:         user_perms(c_k)=user_perms(c_k) \ {p};
22:         count_unperms(c)- -;
23:   end for
24:   CRA= CRA ∪ {c_k, user_perms(cand_cluster)};
25:   if count_unperms(c_k)= =0 then
26:      C'=C' \ {c_k};
27:   end if
28: end for
```

3.3 Role-Engineering Optimization with UCCR

To further satisfy the constraint requirements for user clusters in RBAC systems, while enhancing the interpretation of the mining results, the UCCR should be taken into consideration in the role engineering. Specifically, the unconstrained mining results are checked to identify whether they violate the given cardinality constraints on roles. If there are no constraint violations, they are regarded as efficient solutions. First, we define the role-engineering optimization problem as follows.

Definition 3. *(Role-engineering optimization problem) Given a cluster-permission assignment matrix CPA, and a particular constraint threshold MRC_{user}, find a set Optim_Roles of roles and the corresponding decomposed matrices CRA and RPA, such that the CRA and RPA are consistent with the CPA, the number of roles assigned to any user is less than or equal to MRC_{user}, and the number of the optimal roles is minimized. This process can be formalized as follows:*

$$\begin{cases} \min |Optim\_Roles| \\ CRA \otimes RPA = CPA \\ |user\_roles(c) \cap Optim\_Roles| \leq MRC_{user}.\forall c \in C. \end{cases} \quad (4)$$

According to the mining results from Algorithm 2, we present the optimization process in Algorithm 3.

In Algorithm 3, the unconstrained mining results, Cand_Roles, CRA and C', are considered as inputs, and we output the optimized results, including Optim_Roles and the updated CRA. We make some initializations in the first lines. Next, Lines 5 and 6 indicate that, if the number of roles in c_i equals MRC_{user}−1, and there exist other permissions that are uncovered in c_i, then a new role is generated. If another cluster, c_j, includes role temp, while satisfying the cardinality constraint, then Optim_Roles and CRA are updated in Lines 7-9; otherwise, only the role, $\{user\_roles(c_i) \cup temp\}$, is assigned to c_i in Lines 10-12. In addition, we call Algorithm 2 again in order to revise C' and the relationship of its assignments in Line 15.

Algorithm 3. Role-engineering optimization.

```
Input: unconstrained mining results Cand_Roles, CRA, and $C'$ and the threshold $MRC_{user}$
Output: constrained role set Optim_Roles and updated CRA
1: Create and initialize Optim_Roles= Cand_Roles;
2: Create temporary role temp;
3: while $C' != \Phi$ do
4:    $\forall c \in C'$: user_roles(c)={r ∈ Optim_Roles | (c,r) ∈ CRA};
5:    if $\exists c_i \in C'$: (|user_roles($c_i$)| == $MRC_{user}$ - 1) ∧ (count_unperms($c_i$) != 0) then
6:       temp= user_perms($c_i$);
7:       if $\exists c_j \in \{C' \setminus c_i\}$: (|user_roles($c_j$)| < $MRC_{user}$) ∧ (temp ⊆ user_perms($c_j$)) then
8:          Optim_Roles= Optim_Roles ∪ {temp};
9:          CRA= CRA ∪ {($c_i$,temp),($c_j$,temp)};
10:      else
11:         Optim_Roles= Optim_Roles ∪ {user_roles($c_i$) ∪ temp};
12:         $CRA = (CRA \setminus \cup_{r \in user\_roles(c_i)} \{(c_i, r)\}) \cup \{c_i, user\_roles(c_i) \cup temp\}$;
13:      end if
14:   end if
15:   call Algorithm 2;
16: end while
```

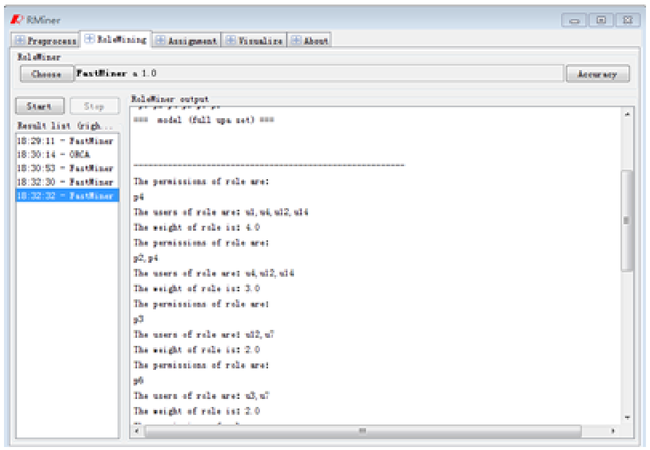
3.4 Running Examples

In this subsection, we present an illustrative example to demonstrate the effectiveness of the REO_UCCR in the following.

Example 1. *Consider the matrix UPA of an original assignment, which is comprised of 15 users and 4 permissions, as shown in Table 1, and $MRC_{user} = 2$.*

In the preprocessing stage, we first identify the distance matrix, D_r, for the original matrix, as shown in Table 2, where both the rows and columns correspond to the row vectors, and the values of the cells are the Hamming distances between any two rows. It is seen that $D_r[2][4] == D_r[2][5] == D_r[2][13] == D_r[2][14] == 0$, $D_r[3][8] == D_r[3][9] == 0$, $D_r[6][7] == D_r[6][15] == 0$, and $D_r[10][11] == 0$. According to Algorithm 1, the same (or similar) row vectors are clustered by choosing the minimal distances and swapping different rows. Similarly, we can also cluster the same (or similar) column vectors in order to further rearrange the matrix, and the result is shown in Table 3. Subsequently, in the generation of user clusters, we can identify the cluster-permission assignments, CPA, and cluster tuples, as shown in Tables 4 and 5, respectively. It is apparent that the compressed CPA is easier to use than the original assignments UPA, which can reduce the mining scale. Indeed, it is convenient and feasible to analyze and handle the compressed cluster set.

Then, we repeatedly call Algorithm 2 and Algorithm 3 in the role mining and optimization stages, and Table 6 presents the optimization process, which does not stop until C' is empty. It is seen from the table that, after the second step in the role mining, only user cluster c_1 remains in C', while the permissions, p_1 and p_2, which are uncovered, are included. Obviously, the number of roles in c_1 is less than 2. However, if we regard $\{p_1, p_2\}$ as a candidate role, then it can be only assigned to c_1 and is not associated with any other cluster, which increases the engineering cost. Thus, we remove the role $\{p_4\}$ that has been assigned to c_1 and assign a new role $\{p_1, p_2, p_4\}$ to c_1 in order to reduce the management burden. In addition, we load and implement our method in the regular mining tool, RMiner [10], as shown in Figure 2, and compare its performance with that of the existing mining methods, as shown in Table 7. It is seen from the table that, however, the user clusters, c_1 and c_2, using the enumeration method [19], violate the given constraint.



```
=== model (full upa set) ===

-------------------------------------------------------------
The permissions of role are:
p4
The users of role are: u1, u6, u12, u14
The weight of role is: 4.0
The permissions of role are:
p2, p4
The users of role are: u4, u12, u14
The weight of role is: 3.0
The permissions of role are:
p3
The users of role are: u12, u7
The weight of role is: 2.0
The permissions of role are:
p6
The users of role are: u3, u7
The weight of role is: 2.0
```

Figure 2: The mining tool, RMiner

Table 1: Original matrix UPA

	p_1	p_2	p_3	p_4
u_1	0	0	0	0
u_2	1	1	0	1
u_3	0	1	1	0
u_4	1	1	0	1
u_5	1	1	0	1
u_6	0	1	1	1
u_7	0	1	1	1
u_8	0	1	1	0
u_9	0	1	1	0
u_{10}	0	0	0	1
u_{11}	0	0	0	1
u_{12}	0	0	0	0
u_{13}	1	1	0	1
u_{14}	1	1	0	1
u_{15}	0	1	1	1

4 Experiments and Analyses

In this section, we perform two groups of experiments with REO_UCCR. The first group of experiments is used to evaluate its performance with respect to different values of constraints. The second group is to compare its performance with the existing methods. We consider four

Table 2: Distance matrix Dr

	UPA'[2]	UPA'[3]	UPA'[4]	UPA'[5]	UPA'[6]	UPA'[7]	UPA'[8]	UPA'[9]	UPA'[10]	UPA'[11]	UPA'[13]	UPA'[14]	UPA'[15]
UPA'[2]	0	3	0	0	2	2	3	3	2	2	0	0	2
UPA'[3]	3	0	3	3	1	1	0	0	3	3	3	3	1
UPA'[4]	0	3	0	0	2	2	3	3	2	2	0	0	2
UPA'[5]	0	3	0	0	2	2	3	3	2	2	0	0	2
UPA'[6]	2	1	2	2	0	0	1	1	2	2	2	2	0
UPA'[7]	2	1	2	2	0	0	1	1	2	2	2	2	0
UPA'[8]	3	0	3	3	1	1	0	0	3	3	3	3	1
UPA'[9]	3	0	3	3	1	1	0	0	3	3	3	3	1
UPA'[10]	2	3	2	2	1	2	3	3	0	0	2	2	2
UPA'[11]	2	3	2	2	2	2	3	3	0	0	2	2	2
UPA'[13]	0	3	0	0	2	2	3	3	2	2	0	0	2
UPA'[14]	0	3	0	0	2	2	3	3	2	2	0	0	2
UPA'[15]	2	1	2	2	0	0	1	1	2	2	2	2	0

Table 3: Rearranged matrix UPA'

	p_1	p_2	p_4	p_3
u_2	1	1	1	0
u_4	1	1	1	0
u_5	1	1	1	0
u_{13}	1	1	1	0
u_{14}	1	1	1	0
u_6	0	1	1	1
u_7	0	1	1	1
u_{15}	0	1	1	1
u_3	0	1	0	1
u_8	0	1	0	1
u_9	0	1	0	1
u_{10}	0	0	1	0
u_{11}	0	0	1	0

real datasets from the work in [12] and adopt the mining tool, RMiner, to evaluate the performance of the unconstrained role mining. The original datasets also include the density of each dataset, the number of the candidate role sets, Cand_Roles, and the execution time, as shown in Table 8. All experiments are implemented on a standard desktop PC, with an Intel i5–7400 CPU, 4 GB RAM, and 160 GB hard disks, running a 64-bit Windows 7 operating system. All simulations are compiled and executed under the Java environment.

Table 4: CPA

	p_1	p_2	p_4	p_3
c_1	1	1	1	0
c_2	0	1	1	1
c_3	0	1	0	1
c_4	0	0	1	0

4.1 Performance Evaluations of the REO_UCCR

To evaluate the effectiveness of our method in the role optimization process, we consider the number of the optimized roles, Optim_Roles, and the size of the assignments, URA, as measures.

Taking the dataset, Firewall 1, as an example for implementing the experiments, the value of the constraint, MRC_{user}, varies from 2 to 8, with a step of 2. We implement the experiments 5 times and take their average values. The results are shown in Figures 3 and 4. In Figure 3, the lateral axis represents MRC_{user}, and the vertical axis represents the number of Optim_Roles. In Figure 4, the lateral axis represents MRC_{user}, and the vertical axis represents the size of the URA.

Figure 3 shows that the number of roles tends to decrease slightly as the value of MRC_{user} increases. When the number of roles is considered as a unique measure, the value of MRC_{user} is greater, and the redundancies of the mining results are fewer. Figure 4 shows that, however, the size of the URA tends to increase remarkably as MRC_{user} increases, which is contrary to the variation tendency in Figure 3. This is because the greater the value of MRC_{user}, the weaker the constraint, and the greater the number of roles assigned to users. The value of $|URA|$ is up to 1516, particularly when MRC_{user} equals 8, which increases the burdens of the system management. On the contrary, the smaller the MRC_{user}, the stronger the constraint will be. Furthermore, the results of the same experiments using the datasets, Firewall 2, Domino, and Healthcare are shown in Figures 5 to 10. It is observed from the tables that, for Firewall 2, the number of roles decreases from 11 to 10 with the increasing value of MRC_{user}, while the value of $|URA|$ is up to 877 when MRC_{user} equals 8. For Domino, the number

Table 5: Four tuples of clusters

User Cluster (c)	Original Permissions (user_permissions (c))	Original User Number (count_users (c))	Uncovered Permission Number (count_unperms (c))
$c_1 = \{u_2, u_4, u_5, u_{13}, u_{14}\}$	$\{p_1, p_2, p_4\}$	5	3
$c_2 = \{u_6, u_7, u_{15}\}$	$\{p_2, p_3, p_4\}$	3	3
$c_3 = \{u_3, u_8, u_9\}$	$\{p_2, p_3\}$	3	2
$c_4 = \{u_{10}, u_{11}\}$	$\{p_4\}$	2	1

Table 6: The optimization process

Step	Optim_Roles	CRA	Updated C'	count_unperms(c)
1	$\{\{p_4\}\}$	$\{(c_4, \{p_4\}), (c_1, \{p_4\}), (c_2, \{p_4\})\}$	$\{c_1, c_2, c_3\}$	$\{p_1, p_2, p_3\}$
2	$\{\{p_4\}, \{p_2, p_3\}\}$	$\{(c_4, \{p_4\}), (c_1, \{p_4\}), (c_2, \{p_4\}), (c_2, \{p_2, p_3\}), (c_3, \{p_2, p_3\})\}$	$\{c_1\}$	$\{p_1, p_2\}$
3 (finish)	$\{\{p_4\}, \{p_2, p_3\}, \{p_1, p_2, p_4\}\}$	$\{(c_4, \{p_4\}), (c_2, \{p_4\}), (c_2, \{p_2, p_3\}), (c_3, \{p_2, p_3\}), (c_1, \{p_1, p_2, p_4\})\}$	ϕ	ϕ

Table 7: Comparison of mining results

User Cluster	Enumeration Method [19]	Blundo [3]	REO_UCCR
c_1	$\{p_1, p_2, p_4\}, \{p_2, p_4\}, \{p_2\}, \{p_4\}$	$\{p_4\}, \{p_1, p_2\}$	$\{p_1, p_2, p_4\}$
c_2	$\{p_2, p_3, p_4\}, \{p_2, p_3\}, \{p_2, p_4\}, \{p_2\}, \{p_4\}$	$\{p_4\}, \{p_2, p_3\}$	$\{p_4\}, \{p_2, p_3\}$
c_3	$\{p_2, p_3\}, \{p_2\}$	$\{p_2, p_3\}$	$\{p_2, p_3\}$
c_4	$\{p_4\}$	$\{p_4\}$	$\{p_4\}$

of roles decreases from 23 to 22 as the value of MRC_{user} increases, while the value of $|URA|$ is up to 169. For Healthcare, the number of roles decreases from 18 to 17 as the value of MRC_{user} increases, while the value of $|URA|$ is up to 143 when MRC_{user} equals 8.

According to the above analyses, we present the optimized results for different datasets when MRC_{user} equals 2, as shown in Table 9. It can be seen that the value of $|URA|$ is less than that of the enumeration method. Therefore, the optimized results using our method not only satisfy the security requirements, but also alleviate the burdens of the system management.

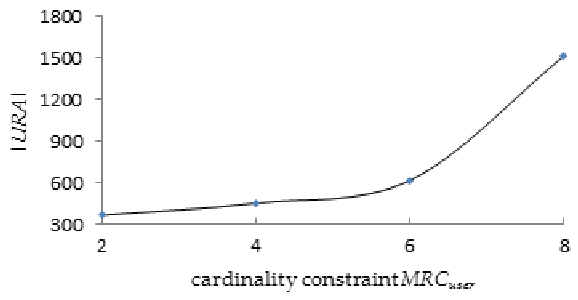


Figure 4: Results of the user-role assignments using Firewall 1

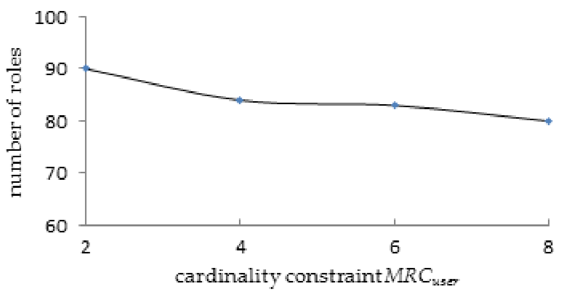


Figure 3: Results of the optimized roles using Firewall 1

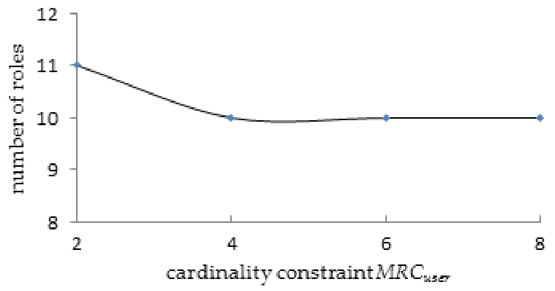


Figure 5: Results of the optimized roles using Firewall 2

Table 8: Original datasets

Dataset	|U|	|P|	|UPA|	Density	|Cand_Roles|	Execution Time(s)
Domino	79	231	730	4%	20	0.01
Healthcare	46	46	1,486	70%	15	0.01
Firewall 1	365	709	31,951	12.3%	69	0.11
Firewall 2	325	590	36,428	19%	10	0.15

Table 9: Comparison of the mining results

Dataset	Enumeration Method [19]		REO_UCCR	
	$\|R\|$	$\|URA\|$	|Optim_Roles|	$\|URA\|$
Domino	20	110	23	79
Healthcare	15	106	18	46
Firewall 1	69	874	90	36
5 Firewall 2	10	434	11	325

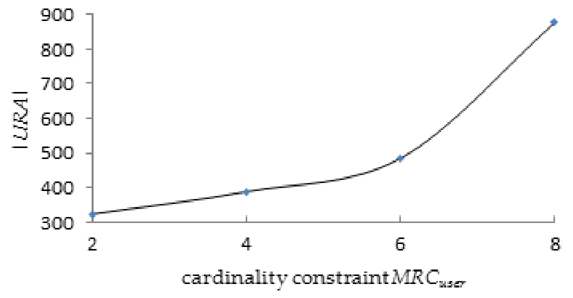


Figure 6: Results of the user-role assignments using Firewall 2

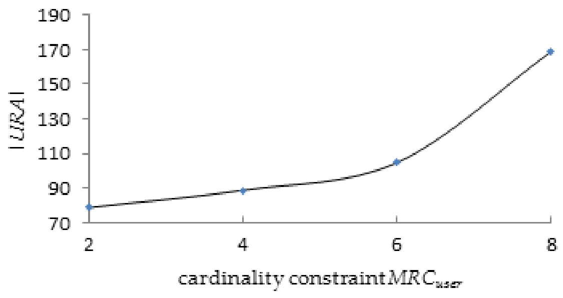


Figure 8: Results of the user-role assignments using Domino

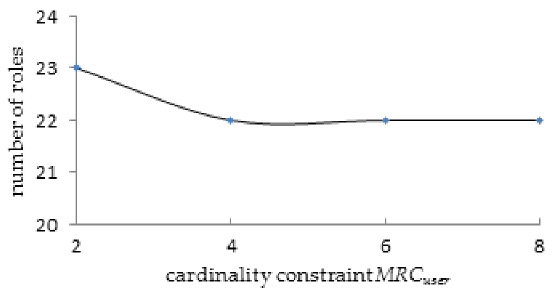


Figure 7: Results of the optimized roles using Domino

4.2 Performance Comparisons with the Existing Methods

To evaluate the efficiency of the REO_UCCR, we implement experiments with the datasets, Domino and Healthcare, as shown in Table 8, and compare its performance with the results of the representative methods, RPA and CPA [9], which are shown in Figures 11 and 12, respectively, where the lateral axis represents MRC_{user}, and the vertical axis represents the number of the optimized roles.

It can be observed, from Figure 11, that the number of roles decreases as MRC_{user} increases for the REO_UCCR, which tends to be stable as MRC_{user} increases to a certain value. Specifically, the number of roles does not obviously vary and remains close to 20 when the value of MRC_{user} exceeds 8. A further observation is that the number of roles first varies slightly and then increases significantly as MRC_{user} decreases. This is because the greater the value of MRC_{user}, the weaker the constraint, and the more roles assigned to any user. In other words, with a greater value of MRC_{user}, regular roles are more applicable and can be utilized more frequently. Thus, fewer irregular roles need to be created, and the number of roles does not vary considerably. For the RPA and CPA, however, the number of roles tends to increase as MRC_{user} increases from 1 to 4. This is because the Domino dataset contains exclusive permissions and produces exclusive roles in the presence of constraints. As shown in the figure, the maximum number of roles is close to 30 when MRC_{user} equals 4, while the minimum number of roles is 23 when MRC_{user} equals 1. Therefore, our method outperforms the RPA and CPA for the dataset, Domino. Similarly, it can be observed, from Figure 12, that the number of roles also decreases as MRC_{user} increases for the REO_UCCR, which tends to be stable and remains close to 15 when MRC_{user} increases to a certain value. However, the variations of the results of both the RPA and CPA are simple. The RPA generates 15 roles that remain unchanged when MRC_{user} exceeds 1, while the number of roles is 18 when MRC_{user} equals 1; and the CPA generates 18 roles that remain unchanged as MRC_{user} varies. Therefore, our method only outperforms the CPA for the dataset, Healthcare.

4.3 Advantages and Shortcomings of the REO_UCCR

From the above analyses, we find the REO_UCCR has the following main advantages:

1) In the initial role-engineering construction, it can reduce the mining scales and alleviate the burdens of system management by using the Hamming distance technique and cluster tuples;

2) In the role-engineering optimization, it can restrict the maximum number of roles assigned to any user and ensure system security by reconstructing a constrained RBAC system, based on the previous mining results.

Compared with the existing studies, the security characteristics of the proposed method are shown in Table 10, where a tick V indicates that the characteristic is available. It can be seen from table that our proposed method still has shortcomings: It does not satisfy the other three cardinality constraints or the separation of duty constraint.

5 Conclusions

A novel role-engineering method, called REO_UCCR, was proposed in this paper. We first converted the basic role mining problem into a clustering problem based on the Hamming distance technique and four tuples of clusters and implemented role mining, while constructing an unconstrained role engineering system. Then, we implemented the role optimization algorithm to reconstruct a constrained RBAC system in order to verify whether the given cardinality constraints can be satisfied in the constructed system. The experiments demonstrated that the

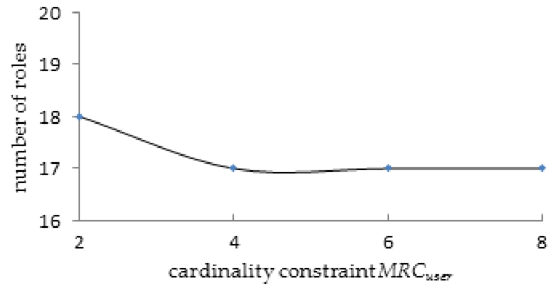


Figure 9: Results of the optimized roles using Healthcare

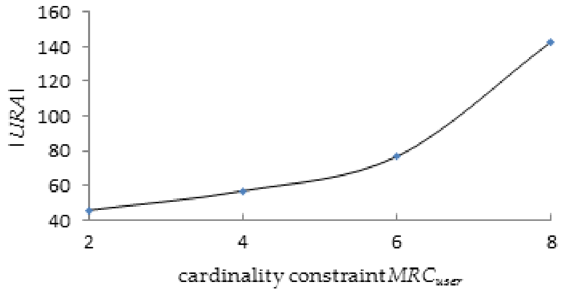


Figure 10: Results of the user-role assignments using Healthcare

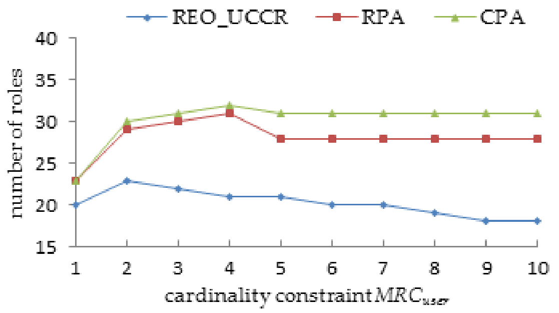


Figure 11: Performance comparison using Domino

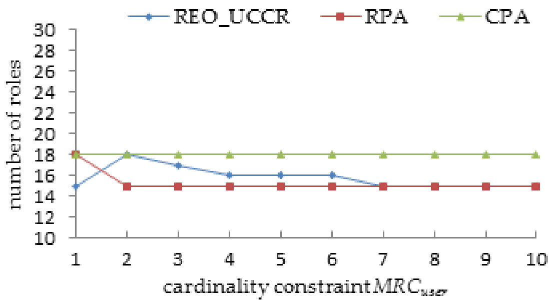


Figure 12: Performance comparison using Healthcare

Table 10: Comparison of security characteristics

Characteristic	Blundo *et al.* [3]	John *et al.* [9]	Harika *et al.* [8]	Wang *et al.* [21]	Blundo *et al.* [4]	Sun *et al.* [13]	Proposed Method
UCCR		V	V		V		V
Other cardinality constraints	V		V	V	V		
Separation of duty constraint						V	
Reducing the mining scales						V	V

proposed method not only alleviates management burdens, but also ensures system security. However, a few interesting issues remain to be solved. To further enhance the interpretability of mining results, one issue for future work is how to implement the other cardinality constraints for role-engineering optimization.

Acknowledgments

This work was partially supported by the Natural Science Foundation of China (61501393), the Natural Science Foundation of Henan Province of China (182300410145, 182102210132), and the Foundation of Henan Educational Committee, under Contract No. 20B520031.

References

[1] W. Bai, Z. Pan, S. Guo, and Z. Chen, "RMMDI: A novel framework for role mining based on the multi-domain information," *Security and Communication Network*, 2019. (https://doi.org/10.1155/2019/8085303)

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

[3] C. Blundo, and S. Cimato, "Constrained role mining," in *Proceedings of the Security and Trust Management-8th International Workshop*, pp. 289–304, Sep. 2012.

[4] C. Blundo, S. Cimato, and L. Siniscalchi, "Managing constraints in role based access control," *IEEE Access*, vol. 8, pp. 140497–140511, 2020.

[5] J. Ernvall, J. Katajainen, and M. Penttonen, "NP-completeness of the Hamming salesman problem," *BIT Numerical Mathematics*, vol. 25, no. 1, pp. 289–292, 1985.

[6] N. Gal-Oz, Y. Gonen, and E. Gudes, "Mining meaningful and rare roles from web application usage patterns," *Computers & Security*, vol. 82, pp. 296–313, 2019.

[7] M. Ghafoorian, D. Abbasinezhad-Mood, and H. Shakeri, "A thorough trust and reputation based RBAC model for secure data storage in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 778–788, 2018.

[8] P. Harika, M. Nagajyothi, J. C. John, S. Sural, J. Vaidya, and V. Atluri, "Meeting cardinality constraints in role mining," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1,pp. 71–84, 2015.

[9] J. C. John, S. Sural, V. Atluri, and J. Vaidya, "Role mining under role-usage cardinality constraint," in *Proceedings of the 27th IFIP TC 11 Information Security and Privacy Conference on Information Security and Privacy Research*, pp. 150–161, June 2012.

[10] R. Li, H. Li, W. Wei, X. Ma, and X. Gu, "RMiner: A tool set for role mining," in *Proceedings of the 18th ACM Symposium on Access Control Models and Technologies*, pp. 193–196, June 2013.

[11] L. Liu, Z. Cao, and C. Mao, "A note on one outsourcing scheme for big data access control in cloud," *International Journal of Electronics and Information Engineering*, vol. 9, no. 1, pp. 29–35, 2018.

[12] B. Mitra, S. Sural, J. Vaidya, and V. Atluri, "A survey of role mining," *ACM Computing Surveys*, vol. 48, no. 4, pp. 1–37, 2016.

[13] W. Sun, S. Wei, H. Guo, and H. Liu, "Role-mining optimization with separation-of-duty constraints and security detections for authorizations," *Future Internet*, vol. 11, no. 9, pp. 201, 2019.

[14] S. D. Stoller, and T. Bui, "Mining hierarchical temporal roles with multiple metrics," *Journal of Computer Security*, vol. 26, no. 1, pp. 121–142, 2018.

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

[16] A. Thakare, E. Lee, A. Kumar, V. B. Nikam, and Y. G. Kim, "PARBAC: Priority-attribute-based RBAC model for azure IoT cloud," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2890–2900, 2020.

[17] Y. Tian, Y. Peng, G. Gao, and X. Peng, "Role-based access control for body area networks using attribute-based encryption in cloud storage," *International Journal of Network Security*, vol. 19, no. 5, pp. 720–726, 2017.

[18] J. D. Ultra, and S. Pancho-Festin, "A simple model of separation of duty for access control models," *Computers & Security*, vol. 68, pp. 69–80, 2017.

[19] J. Vaidya, V. Atluri, and Q. Guo, "The role mining problem: A formal perspective," *ACM Transactions on Information and System Security*, vol. 13, no. 3, pp. 1–31, 2010.

[20] J. Vaidya, V. Atluri, and Q. Guo, "The role mining problem: Finding a minimal descriptive set of roles,"

in *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*, pp. 175–184, June 2007.

[21] J. Wang, J. Dong, and Y. Tan, "Role mining algorithms satisfied the permission cardinality constraint," *International Journal of Network Security*, vol. 22, no. 3, pp. 371–380, 2020.

[22] J. Wang, J. Liu, and H. Zhang, "Access control based resource allocation in cloud computing environment," *International Journal of Network Security*, vol. 19, no. 2, pp. 236–243, 2017.

Biography

Wei Sun received his B.S. and M.S. degrees from the School of Information Engineering, Zhengzhou University, China, in 2003 and 2008, respectively. He is currently working in the Center of Network Information and Computing, Xinyang Normal University. His current research interests include access control and system security.

Xiaoya Yuan received her B.S. and M.S. degrees from the School of Computer and Information Technology, Xinyang Normal University, China, in 2003 and 2011, respectively. She is currently working in the Center of Network Information and Computing, Xinyang Normal University. Her current research interests include image processing and pattern recognition.

Hui Su received his B.S. and M.S. degrees from the School of Computer and Information Technology, Xinyang Normal University, China, in 1992 and 2008, respectively. He is currently working in the Center of Network Information and Computing, Xinyang Normal University. His current research interests include image processing and pattern recognition.