International Journal of Innovative Computing, Information and Control
Volume **12**, Number **1**, February **2016**
 ISSN 1349-4198
pp. 91–101

MUTUAL EXCLUSION ROLE CONSTRAINT MINING BASED ON WEIGHT IN ROLE-BASED ACCESS CONTROL SYSTEM

Xiaopu Ma[1,*], Jianfang Wang[1], Li Zhao[1] and Ruixuan Li[2]

[1]School of Computer and Information Technology
Nanyang Normal University
No. 1638, Wolong Road, Wolong District, Nanyang 473061, P. R. China
*Corresponding author: mapxiao@nynu.edu.cn

[2]School of Computer Science and Technology
Huazhong University of Science and Technology
No. 1037, Luoyu Road, Hongshan District, Wuhan 430074, P. R. China
rxli@mail.hust.edu.cn

Received July 2015; revised November 2015

Abstract. *Role-based access control (RBAC) is the most popular access control model at present, and has become the norm in many applications. Role engineering is a way to migrate a non-RBAC system to an RBAC system. However, none of the role engineering work has employed mutual exclusion role constraint mining based on weight to our knowledge although constraint is an important aspect of RBAC, thus providing the motivation for this work. In this paper, we first define the weight of permission by considering the attributes of user and permission, and then study how to generate mutual exclusion role constraint based on weight. Finally, experiments on performance study prove the superiority of our approach.*
Keywords: RBAC, Role engineering, Weight, Mutual exclusion role constraint

1. **Introduction.** Role-based access control (RBAC) recently has been widely deployed in enterprise security management and enterprise management products [1]. The notion of role makes RBAC have several benefits than others [2]. As a solution to facilitate the process to migrate a non-RBAC system to an RBAC system, role engineering is introduced [3].

However, a key challenge that has not been adequately addressed so far is how to define the weight of permission and how to generate mutual exclusion role constraint based on weight in role engineering. In other words, most of the existing role engineering approaches did not consider the different nature and importance of each permission, or treated each permission evenly [4, 5]. However, this is not always the case. For example, the permission *read* of the patient's personal information may be more important than the permission *write* to the patient's personal information. This is because the *read* permission usually leads to more information leakage, but the standard role mining simply ignores this difference. In another case, the permission *write* to the student's achievement may be more significant than the permission *read* of the student's achievement. Furthermore, constraint is a set of imposed rules on RBAC, and they are one of the most distinctive and important features of the RBAC approach. We can get an incomplete architectural structure of RBAC if there are no constraints. A common example is that of *mutually exclusive roles*, such as purchasing manager and account payable manager. In most organizations, the same individual will not be permitted to be a member of both roles, because this creates a possibility for committing fraud. If there is no idea about this constraint in RBAC system,

there may be wrong in enforcing the principle of least privilege in RBAC system. For example, let us assume that there is a requested permission set $RQ = \{p_1, p_2, p_3\}$ (the user requires permissions p_1, p_2 and p_3 to perform the task), $ass\_perms(r_1) = \{p_1, p_2\}$, $ass\_perms(r_2) = \{p_2, p_3\}$ (permission p_1 and p_2 belong to r_1, r_2 has the permissions p_2 and p_3). In this situation, role r_1 and r_2 can enforce the principle of least privilege for the requested permission set because the goal of the principle of least privilege is to identify the minimal set of roles whose permissions exactly equal the requested permission set. However, this approach may be wrong if r_1 and r_2 are *mutually exclusive roles*.

To this aim, this research tries to define weight and mining mutual exclusion role constraint based on weight in a feasible way. We first introduce the notion of weight of permission. Our approach for the notion of weight can produce a natural importance on permission to assist the mutual exclusion role constraint mining process that most approaches currently lack. Finally, the algorithm of mutual exclusion role constraint mining based on weight is tested on simulated test data. Experiments on performance study are given to prove the superiority of our approach.

The remainder of the paper is organized as follows. We discuss related work in Section 2. The limitations in existing applications for constraint mining drive our motivation and Section 3 proposes how to define the weight of permission. Section 4 describes the algorithm of mining mutual exclusion role constraint based on weight. A summary of our experimental results on simulated data is discussed in Section 5. Finally, Section 6 provides some insight into our ongoing and future work.

2. **Related Work.** In role engineering, the top-down approach starts from the user scenario and business process in order to find out a role state that guarantees all the roles receive their necessary rights so they can perform their functions and no more than their functions [6]. However, since there are often dozens of business processes or tasks and ten thousands of users, it makes the way time consuming and human intensive. Under the bottom-up approach, roles can be generated through the user permission assignments and some corresponding attribute information. According to their outputs, the bottom-up role mining algorithms can be divided into two categories. The first class algorithm generates a set of candidate roles, and then gives every role a priority value. The higher a role's priority value is, the more likely the role can be selected by users. The representative algorithms of the first class are CompleteMiner (CM) and FastMiner (FM) [7]. As for the CM algorithm, it can get the unique intersection sets from the generated roles while the FM algorithm can only find the intersection between pairs of initial roles. The second class algorithm uses Weighted Structural Complexity (WSC) as a common quality measure to generate a complete RBAC state. There are many algorithms belonging to this catalog, such as ORCA [8] (which is a hierarchical clustering algorithm where every permission is exactly assigned only to one role), HierarchicalMiner (HM) [9] and GO [10] (where HM restructures the lattices according to the cost decreased of the RBAC with a greedy strategy from the concept lattices, and GO reduces the number of role to user and permission to role assignments by a graph optimization).

Furthermore, constraint is an essential aspect of RBAC and is sometimes argued to be the principle motivation for RBAC [11, 12]. The most frequently mentioned constraint in context of RBAC is *mutually exclusive constraint*, this constraint in terms of a many-to-many user to role assignment relationships specifies that one individual cannot be a member of both roles, and this constraint in terms of a many-to-many permission to role assignment relationships specifies that the same permission cannot be assigned to both role. Another example is *cardinality constraint*; for example, the cardinality constraint of user is defined as the maximum number of users which a role can have, the cardinality

constraint of role is defined as the maximum number of roles to which an individual user can belong, the cardinality constraint of permission is defined as the maximum number of roles to which a permission can belong and the number of permissions which a role can own should also have cardinality constraint. The third concept is *prerequisite constraints*; for example, a user can be assigned to role r_1 only if the user is already a member of role r_2. Finally, the constraint also can apply to *sessions*. Although the constraint is essential for the RBAC model, only a few researches took the constraint into account in role mining. For example, Kumar et al. consider the maximum number of users that can be assigned to each role in the mining RBAC role approach [13]. John et al. take the number of roles to which an individual user can belong into the role mining approach [14].

However, the traditional role mining approach only considers how to mine role based on different constraints without taking into account how to mine different constraints in role engineering. Furthermore, the traditional constraint mining approach assumes that permission has the same importance without taking account of their weight. To this aim, this research tries to assign weight to each permission in a feasible way. We introduce the concepts of original similarity between users, resources, operations and permissions, and then propose a reinforced similarity to represent the importance of each permission. Our focus is on how to calculate the similarity and how to define the weight of each permission based on the similarity. We also present a new mutual exclusion role constraint mining algorithm based on weight (WCM) to address the above problem. The experimental results are tested to show the effectiveness of our findings.

3. **Preliminaries.** We develop the material in this paper in the context of the NIST standard, the most widely known RBAC model [1]. This model includes RBAC0 which includes users, permissions, roles and their relationships, RBAC1 which introduces hierarchical between roles based on RBAC0, RBAC2 which adds exclusivity relations among roles with respect to user assignment based on RBAC1, RBAC3 which defines exclusivity relations with respect to roles that are activated as part of a user's session based on RBAC2. For the sake of simplicity, we do not consider sessions in this paper. The architecture of RBAC is shown in Figure 1.

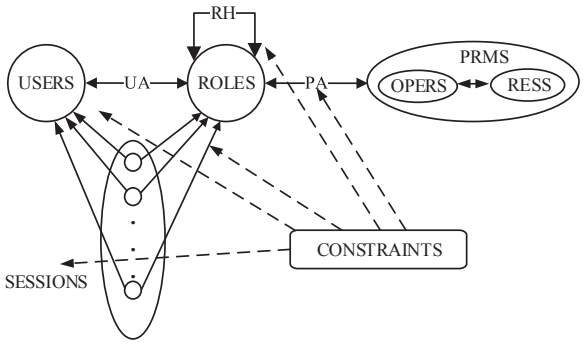

FIGURE 1. The architecture of RBAC model

Definition 3.1. *The RBAC model contains the following components:*

- *USERS, ROLES, OPERS, and RESS, users, roles, operations and resources respectively;*
- $PRMS = 2^{(OPERS \times RESS)}$*, the set of permissions;*
- $UA \subseteq USERS \times ROLES$*, a many-to-many user to role assignment relationship;*

- $ass\_users(r) = \{u \in USERS \mid (u, r) \in UA\}$, *the mapping of role* r *onto a set of users;*
- $PA \subseteq PRMS \times ROLES$, *a many-to-many role to permission assignment relationship;*
- $ass\_perms(r) = \{p \in PRMS \mid (p, r) \in PA\}$, *the mapping of role* r *onto a set of permissions;*
- $UPA \subseteq USERS \times PRMS$, *a many-to-many user to permission assignment relationship;*
- $RH \subseteq R \times R$ *is a partial order of* R *called the role hierarchy of role dominance relation, also written as* $\succeq$, *when* $r_1 \succeq r_2$ *only if all permissions of* r_2 *are also permissions of* r_1, *and all users of* r_1 *are also users of* r_2. *Formally:* $r_1 \succeq r_2 \Rightarrow ass\_perms(r_2) \subseteq ass\_perms(r_1) \wedge ass\_users(r_1) \subseteq ass\_users(r_2)$;
- $POP \subseteq PRMS \times OPERS$, *a many-to-many relationship between permissions and operations;*
- $PRE \subseteq PRMS \times RESS$, *a many-to-many relationship between permissions and resources;*
- $Op(p : PRMS) \rightarrow \{op \subseteq OPERS\}$, *the permission to operation mapping, which gives the set of operations associated with permission* p;
- $Re(p : PRMS) \rightarrow \{re \subseteq RESS\}$, *the permission to resource mapping, which gives the set of resources associated with permission* p.

We define $M = |USERS|$, $N = |PRMS|$, where M denotes the number of users, and N denotes the number of permissions. The meaning of M and N is used throughout the paper. Here, we can use an $M \times N$ binary matrix $MUPA$ to describe the assignment relationships between users and permissions before RBAC not construction. The element $MUPA_{i,j} = 1$ denotes that the ith user has the jth permission or the jth permission belongs to the ith user; otherwise, the element $MUPA_{i,j} = 0$ indicates that the ith user has not the jth permission. Since permission can be described as $2^{(OPERS \times RESS)}$, we can split permission into operations and resources where operations and resources carry the function and action information of permission. Such a split approach can provide good interpretability of a permission. For example, the permission to delete a book in a library management system can be represented as follows:

- (Permission), DeleteBook
- (Operation), Delete
- (Resource), Book

Here we can use an $N \times P$ binary matrix $MPOP$ to describe the relationships between permissions and operations (such as add, create, delete, and write). The element $MPOP_{i,j} = 1$ denotes that the ith permission has the jth operation; otherwise, the element $MPOP_{i,j} = 0$ indicates that the ith permission has not the jth operation (where P denotes the number of operations). We use another $N \times S$ binary matrix $MPRE$ to describe the relationships between permissions and resources (such as accounts, and books). The element $MPRE_{i,j} = 1$ denotes that the ith permission has the jth resource; otherwise, the element $MPRE_{i,j} = 0$ indicates that the ith permission has not the jth resource (where S denotes the number of resources). Since the basic entities in the standard RBAC have been defined, here we only define the attribute of user as follows.

Definition 3.2. *The attribute of user in RBAC is defined as:*

- $USERA$, *the set of attributes with user, such as locations, department affiliations, or task description of the user;*
- $USA \subseteq USERS \times USERA$, *a many-to-many relationship between users and attributes.*

Here we can use an $M \times K$ binary matrix $MUSA$ to describe the relationships between users and attributes. The element $MUSA_{i,j} = 1$ denotes that the ith user has the jth attribute; otherwise, the element $MUSA_{i,j} = 0$ indicates that the ith user has not the jth attribute (where K denotes the number of user's attributes). Since users are represented by the set of attributes (such as locations, department affiliations, or task description of the users that can be obtained from the top-down information), we can give a measure of original similarity between users based on this.

Definition 3.3. *The original similarity between the ith user and the jth user is defined as*

$$sim(u_i, u_j) = \frac{|MUSA_{i.} \cap MUSA_{j.}|}{|MUSA_{i.} \cup MUSA_{j.}|} \tag{1}$$

Above definition is based on a statistical similarity measure called Jaccard co-efficient between users, where 0 implies no similarity between the ith user and the jth user, and 1 represents an exact match between these two users [15].

According to the above definition, it just focuses on using a single type of relationships to calculate the similarity between users. For example, it just uses the set of attributes to calculate the similarity between users. Although this method is useful to compute the similarity between users, it is not good at using the other information to make the similarity more accurate. For example, the set of attributes between users that we call intra-relationships can affect the similarity between users on one hand; on the other hand, since users are represented by permission set, these permission sets that we call inter-relationships can also influence the similarity between users. Hence, we can use the intra-relationships and inter-relationships to reinforce the original similarity between users. The formal definition of the reinforced similarity between users is given below.

Definition 3.4. *The reinforced similarity between the ith user and the jth user is defined as*

$$resim(u_i, u_j) = wua \times sim(u_i, u_j) + wup \times \frac{|MUPA_{i.} \cap MUPA_{j.}|}{|MUPA_{i.} \cup MUPA_{j.}|} \tag{2}$$

Here $wua+wup = 1$ and wua, wup are parameters used to adjust the relative importance about the reinforced similarity between the ith user and the jth user corresponding to each relationship. If we have no prior knowledge of the initial attributes with users, we can set wua to 0. Analogously, we give the definition of original similarity between the ith resource and jth resource as follows.

Definition 3.5. *The original similarity between the ith resource and the jth resource is defined as*

$$sim(rs_i, rs_j) = \frac{\left|MURS^T_{.i} \cap MURS^T_{.j}\right|}{\left|MURS^T_{.i} \cup MURS^T_{.j}\right|} \tag{3}$$

Here $MURS = MUPA \bigotimes MPRE$, $MURS^T_{.i}$ or $MURS^T_{.j}$ is the ith or jth column transpose of the user resource relationships matrix $MURS$, which is a boolean vector representing the set of users has the ith resource or the jth resource. Similarly, we can define the original similarity between the ith operation and the jth operation as follows.

Definition 3.6. *The original similarity between the ith operation and the jth operation is defined as*

$$sim(op_i, op_j) = \frac{\left|MUOP^T_{.i} \cap MUOP^T_{.j}\right|}{\left|MUOP^T_{.i} \cup MUOP^T_{.j}\right|} \tag{4}$$

Here $MUOP = MUPA \bigotimes MPOP$, $MUOP^T_{.i}$ or $MUOP^T_{.j}$ is the ith or jth column transpose of the user operation relationships matrix $MUOP$, which is a boolean vector representing the set of users has the ith operation or the jth operation. Hence, we can formally define the reinforced similarity between permissions and the weight of permission as follows.

Definition 3.7. *The reinforced similarity between the ith permission and the jth permission is defined as*

$$\begin{aligned} resim(p_i, p_j) = wpu \times & \frac{\left|MUPA^T_{.i} \cap MUPA^T_{.j}\right|}{\left|MUPA^T_{.i} \cup MUPA^T_{.j}\right|} \\ & + wps \times \frac{\sum\limits_{rs_a \in Re(p_i)} \sum\limits_{rs_b \in Re(p_j)} sim(rs_a, rs_b)}{|MPRE_{i.}| \times |MPRE_j|} \\ & + wpo \times \frac{\sum\limits_{op_c \in Op(p_i)} \sum\limits_{op_d \in Op(p_j)} sim(op_c, op_d)}{|MPOP_{i.}| \times |MPOP_{j.}|} \end{aligned} \tag{5}$$

Here $wpu+wps+wpo = 1$ and wpu, wps, wpo are parameters used to adjust the relative importance about the reinforced similarity between the permissions corresponding to the original similarity between permissions, the original similarity between resources and the original similarity between operations. If we have no prior knowledge of the original similarity between resources and the original similarity between operations, we can set $wps = wpo = 0$.

Definition 3.8. *The weight of permission p_i is defined as*

$$\begin{aligned} w_{p_i} = 1 - \Bigg(& ww \times w_0 + wp \times \frac{\sum\limits_{j=1, j\neq i}^{N} resim(p_i, p_j)}{(N-1)} \\ & + wu \times \frac{\sum\limits_{k=1 \wedge MUPA_{ki}=1}^{M} \sum\limits_{j=1, j\neq i}^{M} resim(u_i, u_j)}{(M-1) \times |MUPA_{ki} = 1|} \Bigg) \end{aligned} \tag{6}$$

Here w_0 is the initial weight of permission p_i preset by the system based on the knowledge of comprehensive effect of all factors on permission p_i, $ww + wp + wu = 1$ and ww, wp, wu are parameters used to adjust the relative importance about the weight of permission corresponding to the initial weight, reinforced similarity between permissions and reinforced similarity between users.

Definition 3.9. *Given a set of roles $rs \subseteq ROLES$, we say $r_i \in rs$ and $r_j \in rs$ $(i \neq j)$ are mutual exclusion roles if $ass\_users(r_i) \cap ass\_users(r_j) = \emptyset$. We will use the notation $\overline{r_i \wedge r_j}$ to specify that role r_i and r_j are mutually exclusive roles.*

For example, the user cannot be a member of both accounts-manager role and purchasing-manager role. More generally, we allow to have more than one role in the notation. For example, $\overline{r_1 r_2 \wedge r_3 r_4}$ describes that the user cannot have both the role set $\{r_1, r_2\}$ and $\{r_3, r_4\}$. Analogously, we give the definition of mutual exclusion permissions as follows.

Definition 3.10. *Given a set of permissions $ps \subseteq PRMS$, we say $p_i \in ps$, $p_j \in ps$ $(i \neq j)$ are mutual exclusion permissions if $p_i \in ass\_perms(r_i)$ and $p_j \notin ass\_perms(r_i)$ for all*

$r_i \in ROLES$. We will use the notation $\overline{p_i \wedge p_j}$ to specify that permissions p_i and p_j are mutually exclusive permissions.

For example, the permission to issue checks and the permission to audit the operation should not be assigned to the same role. More generally, we allow to have more than one permission in the notation. For example, $\overline{p_1p_2 \wedge p_3p_4}$ describes that the role cannot have both the permissions set $\{p_1, p_2\}$ and $\{p_3, p_4\}$ or $\{p_1, p_2\}$ and $\{p_3, p_4\}$ cannot be given to the same role.

4. **Algorithm.** In order to mine mutual exclusion role based on weight, we assign a real number $w_{p_i} \in [0, 1]$ for each permission $p_i \in PRMS$ $(i = 1, \ldots, N)$, which we call the weight of permission that can be calculated by Definition 3.8. Since the frequent permission set is to find implications between the elements in 2^{PRMS}, not $PRMS$, we must define weight for all elements in 2^{PRMS} [16]. This can be done as follows:

For any permission set $PS \subseteq 2^{PRMS}$, suppose that there has $PS = \{p_1, p_2, \ldots, p_k\}$, where $p_i \in PRMS$ $(i = 1, \ldots, N)$. We define the weight of PS as follows.

Definition 4.1. *The weight of permission set PS is defined as*

$$w_{PS} = \sum_{i=1}^{k} w_{p_i} \tag{7}$$

According to the traditional support function and confidence used in *Apriori* algorithm [17], we define the *weighted support* and *confidence* for any permission set to generate mutual exclusion role as follows (Here we just consider how to generate mutual exclusion role from user to permission assignment relationships because the same algorithm also can generate mutual exclusion permission from permission to role assignment relationships).

Definition 4.2. *The weighted support of permission set PS is defined as*

$$wsup_{PS} = w_{PS} \times \frac{numUsers(PS)}{numUsers(All)} \tag{8}$$

Definition 4.3. *The confidence of the association rule $X \Rightarrow Y$ is the probability that Y exists given that a transaction contains X, i.e.,*

$$conf(X \Rightarrow Y) = \frac{probability(X \cup Y)}{probability(X)} = \frac{numUsers(XY)}{numUsers(X)} \tag{9}$$

Here $numUsers(PS)$ is the number of users which possess permission set PS that presented in the user to permission assignment relationships matrix, and $numUsers(All)$ is the total number of users in the matrix. Before RBAC model is implemented, there is only user to permission assignment relationship in the system; in this situation we regard permission as item, the collection of all user to permission assignment relationships as a transaction database and each user permission assignment relationship as a transaction. Then we can define a mutual exclusion role constraint based on *Apriori* algorithm as follows.

Definition 4.4. *A mutual exclusion role constraint is an implication of the form $\overline{X \Rightarrow Y}$, where $X \subseteq PRMS$, $Y \subseteq PRMS$, $X \cap Y = \emptyset$ and $MUPA_{i.} \cap X = \emptyset \vee MUPA_{i.} \cap Y = \emptyset$ for each $u_i \in USERS$ $(i = 1, \ldots, M)$.*

In reality, access control configurations in any large organization are noisy [18]. Hence, we relax the restriction that the mutual exclusion constraint between roles holds in the user permission assignment relationships if the constraint has certain user-specified *weighted minimum support* (*wminsup*) and *confidence* based on weight. Hence, there are three

cases to compute the weighted support and confidence of the mutual exclusion constraint between roles as follows.

1) $X \Rightarrow \overline{Y}$, in this situation users possess permission set X and do not contain permission set Y at the same time that presented in the user to permission assignment relationships.

$$\begin{cases} wsup = w_{X \cup \overline{Y}} \times \dfrac{numUsers\left(X\overline{Y}\right)}{numUsers(All)} \\ conf = \dfrac{numUsers\left(X\overline{Y}\right)}{numUsers(X)} \end{cases} \tag{10}$$

2) $\overline{X} \Rightarrow Y$, in this situation users possess permission set Y and do not contain permission set X at the same time that presented in the user to permission assignment relationships.

$$\begin{cases} wsup = w_{\overline{X} \cup Y} \times \dfrac{numUsers\left(\overline{X}Y\right)}{numUsers(All)} \\ conf = \dfrac{numUsers\left(\overline{X}Y\right)}{numUsers\left(\overline{X}\right)} \end{cases} \tag{11}$$

3) $\overline{X} \Rightarrow \overline{Y}$, in this situation users do not possess permission set X and Y at the same time that presented in the user to permission assignment relationships.

$$\begin{cases} wsup = w_{\overline{X} \cup \overline{Y}} \times \dfrac{numUsers\left(\overline{XY}\right)}{numUsers(All)} \\ conf = \dfrac{numUsers\left(\overline{XY}\right)}{numUsers\left(\overline{X}\right)} \end{cases} \tag{12}$$

Now we design the weighted constraint mining algorithm WCM to generate the mutual exclusion constraint between roles based on weight as follows.

1) Scan the user permission assignments matrix $MUPA$ to generate all the user to permission assignment relationships as $< UID, PermSet >$. For example, we use $< u_1$, p_1 p_2 p_3 $\overline{p_4} >$ to describe that u_1 has the permissions p_1, p_2 and p_3 while having not permission p_4 at the same time.

2) Scan all of the $< UID, PermSet >$ to generate all candidate permission sets as follows:

Let $PRMS$ be the set of all permissions. Support that Y is a q-permission set, where $q < k$. In the set of the remaining permissions $(PRMS - Y)$, let the permissions with the $(k - q)$ greatest weights $w_{p_1}, w_{p_2}, \ldots, w_{p_k}$. We can say the maximum possible weight for any k-permission containing Y is:

$$W(Y, k) = \sum_{p_i \in Y} w_{p_i} + \sum_{j=1}^{k-q} w_{p_j} \tag{13}$$

in which the first sum is the sum of the weights for the q-permission Y, and the second sum is the sum of the $(k-q)$ maximum remaining weights. Thus, we can get the minimum count for a large k-permission containing Y is given by:

$$minCount = \left\lceil \frac{wminsup}{W(Y, k)} \times numUsers(All) \right\rceil \tag{14}$$

Thus, we can get all the candidate permission sets if the number of permission is larger than the *minCount*, and we denote them as C (The idea behind this step is to generate the candidate permission sets based on weight based on the theory described in [19]).

Finally, generate all combinations of permission sets whose *weighted support* is greater than the user specified *minimum weighted support* based on the candidate permission sets, and we call them frequent permission sets and denote as F.

3) Generate mutual exclusion constraint between roles based on weight from the frequent permission sets found in Step 2 using association rules mining algorithm as *Apriori*. In this step, we eliminate the rules that do not contain $\overline{p_i}$ ($p_i \in PRMS$) in order to enhance the algorithm's efficiency.

5. **Experimental Results.** In this section, we will implement the WCM algorithm on an Intel(R) Core(TM) i5-4590 @CPU 3.3G machine with 4GB memory to evaluate our method and present the most relevant results. The running platform is Windows 8. We mainly consider four weight schemes W0: $ww = 0$, $wp = 0$, $wu = 0$; W1: $ww = 0$, $wp = 0.5$, $wu = 0.5$; W2: $ww = 0$, $wp = 2/3$, $wu = 1/3$; W3: $ww = 0$, $wp = 1/3$, $wu = 2/3$. In the three tuple W0, we do not consider the weight of each permission, and in the three tuple W1, it assumes that the similarity between users and the similarity between permissions have the same importance to the weight of permission. As for the other three tuple W2, it assumes that the similarity between permissions is more important while in the last three tuple W3, it assumes that the similarity between users is more important to the weight of permission. Furthermore, one can adjust the weight parameters to meet different requirements. Table 1 shows the test parameters.

Table 1. Parameter settings for testing WCM algorithm

	Initial Weight (ww)	Perm Re-Similarity (wp)	User Re-Similarity (wu)
W0	0	0	0
W1	0	0.5	0.5
W2	0	2/3	1/3
W3	0	1/3	2/3

To study the performance of our method, we generate the synthetic test data as follows. First, we use for loop to create the relationships between users and permissions. For each user, a random number of permissions are chosen. The value of each element in the relationships is randomly chosen as 0, indicating that the user has no such permission, or 1, indicating that the user has such permission.

Figure 2(a) shows the average number of mutual exclusion role constraint in the various user specified minimum support thresholds under the parameter W0 where there are 50 users and 20 permissions (In this situation, we cannot consider the weight of each permission). Figure 3(a) shows the average number of mutual exclusion role constraint in the various user specified minimum weighted support thresholds under the parameters W1, W2, W3 where there are 50 users and 20 permissions. From Figures 2(a) and 3(a), it can be seen that when the minimum support threshold (or minimum weighted support threshold) is low, there is a large number of identified mutual exclusion constraint between roles. As the minimum support threshold (or minimum weighted support threshold) increases, the number of mutual exclusion constraint between roles will decrease. Furthermore, we can find that the number of mutual exclusion constraint will increase when we increase the reinforced similarity between users from Figure 3(a). For example, at the same minimum weighted support threshold 0.34, the number of constraint is 28, 54 and 87 under parameters W1, W2, W3 respectively. It means that the reinforce similarity between users can

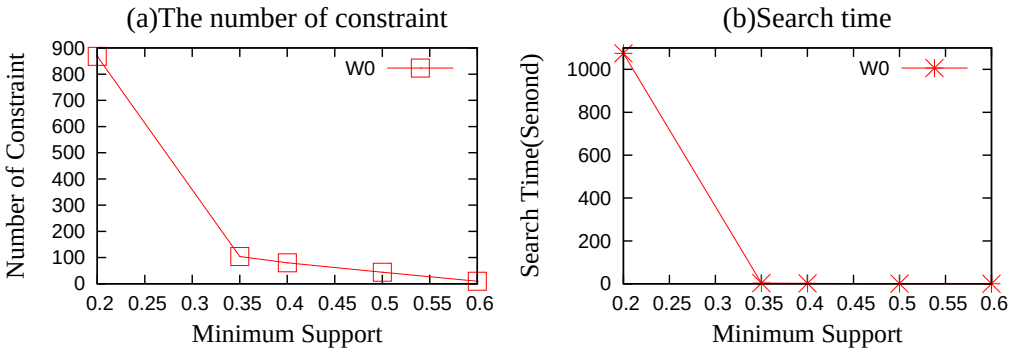

FIGURE 2. Performance analysis under 50 users and 20 permissions under parameter W0 (minimum confidence = 1)

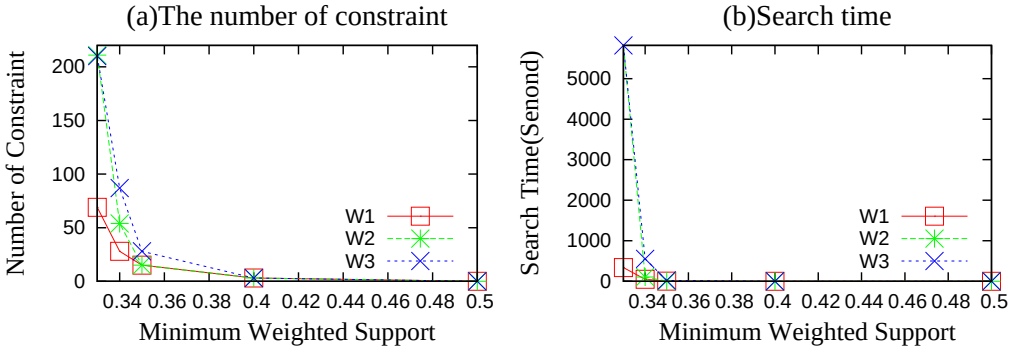

FIGURE 3. Performance analysis under 50 users and 20 permissions under parameters W1, W2, W3 (minimum confidence = 1)

impact more on the weight of each permission. Figure 2(b) shows the average search time for mutual exclusion constraint between roles in the different minimum support thresholds under the parameter W0. Figure 3(b) shows the average search time for mutual exclusion constraint between roles in the different minimum weighted support thresholds under the parameters W1, W2, W3. It can be seen that when the minimum support or weighted support threshold is larger, the searching time will decrease. However, when the minimum support or weighted support threshold is smaller, the average search time for generating mutual exclusion constraint between roles is increasing exponentially.

6. **Conclusions.** While many role mining approaches have been proposed recently, none of them considered how to mine mutual exclusion role constraint based on weight. It may fail to reflect the constraints in the security systems. In this paper, we present a mutual exclusion role constraint mining approach based on weight. We first define the weight of permission and also propose an algorithm to mine mutual exclusion role constraint based on weight. We carry out the experiments to illustrate the effectiveness of the proposed techniques. As a result, the proposed approach can generate mutual exclusion role based on weight very well. For the future work, we will try to find more meaningful information that is available to make the weight of permissions more accurate. It could be used to further refine the potential roles. Furthermore, we will find the efficient algorithm to generate the mutual exclusion role constraint based on weight.

Acknowledgements. This work is supported by the Joint Fund for National Natural Science Foundation of China and Henan Province for Fostering Talents under Grant No.

U1304619, the National Natural Science Foundation of China under Grant Nos. 61173170, 61300222, and 61433006, and the Innovation Fund of Nanyang Normal University under Grant No. ZX2013013. We thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] D. F. Ferraiolo, R. Sandhu, S. Gavrila et al., Proposed NIST standard for role-based access control, *ACM Trans. Information and System Security*, vol.4, no.3, pp.224-274, 2001.

[2] X. Ma, R. Li, Z. Lu, J. Lu and M. Dong, Specifying and enforcing the principle of least privilege in role-based access control, *Concurrency and Computation: Practice and Experience*, vol.23, no.12, pp.1313-1331, 2011.

[3] E. J. Coyne, Role engineering, *Proc. of the 1st ACM Workshop on Role-based Access Control*, p.4, 1996.

[4] X. Ma, R. Li and Z. Lu, Role mining based on weights, *Proc. of the 15th ACM Symposium on Access Control Models and Technologies*, pp.65-74, 2010.

[5] W. Zhang, Y. Chen, C. A. Gunter, D. Liebovitz and B. Malin, Evolving role definitions through permission invocation patterns, *Proc. of the 18th ACM Symposium on Access Control Models and Technologies*, pp.37-48, 2013.

[6] E. B. Fernadez and J. C. Hawkins, Determining role rights from use cases, *Proc. of the 2nd ACM Workshop on Role-based Access Control*, Fairfax, Virginia, USA, pp.121-125, 1997.

[7] A. Baumgrass, M. Strembeck and S. R. Ma, Deriving role engineering artifacts from business processes and scenario models, *Proc. of the 16th ACM Symposium on Access Control Models and Technologies*, pp.11-20, 2011.

[8] J. Schlegelmilch and U. Steffens, Role mining with ORCA, *Proc. of the 10th ACM Symposium on Access Control Models and Technologies*, Stockholm, Sweden, pp.168-176, 2005.

[9] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo and J. Lobo, Mining roles with semantic meanings, *Proc. of the 13th ACM Symposium on Access Control Models and Technologies*, Colorado, USA, pp.21-30, 2008.

[10] D. Zhang, K. Ramamohanarao and T. Ebringer, Role engineering using graph optimisation, *Proc. of the 12th ACM Symposium on Access Control Models and Technologies*, Antipoles, France, pp.139-144, 2007.

[11] X. Ma, R. Li, Z. Lu et al., Mining constraints in role based access control, *Mathematical and Computer Modelling*, vol.55, nos.1-2, pp.87-96, 2012.

[12] H. Lu, J. Vaidya, V. Atluri and Y. Hong, Constraint-aware role mining via extended Boolean matrix decomposition, *IEEE Trans. Dependable and Secure Computing*, vol.9, no.5, pp.655-669, 2012.

[13] R. Kumar, S. Sural and A. Gupta, Mining RBAC roles under cardinality constraint, *Proc. of the 6th International Conference on Information Systems Security*, pp.171-185, 2011.

[14] J. C. John, S. Sural, V. Atluri and J. S. Vaidya, Role mining under role-usage cardinality constraint, *Proc. of the 27th IFIP International Information Security and Privacy Conference*, pp.150-161, 2012.

[15] R. Li, W. Wang, X. Ma et al., Mining roles using attributes of permissions, *International Journal of Innovative Computing, Information and Control*, vol.8, no.11, pp.7909-7924, 2012.

[16] X. Chen, Z. Yi and Y. Wu, Mining association rules based on seed items and weights, *Proc. of the 2nd International Conference on Fuzzy Systems and Knowledge Discovery*, pp.603-608, 2005.

[17] F. Geerts, B. Goethals and T. Mielikainen, Tiling databases, *Proc. of the 7th International Conference Discovery Science*, pp.278-289, 2004.

[18] I. Molloy, N. Li, J. Lobo and L. Dickens, Mining roles with noisy data, *Proc. of the 15th ACM Symposium on Access Control Models and Technologies*, Pittsburgh, PA, USA, pp.45-54, 2010.

[19] C. H. Cai, W. C. Fu, C. H. Cheng and W. W. Kwong, Mining association rules with weighted items, *Proc. of the 1998 International Symposium on Database Engineering and Applications*, pp.68-77, 1998.