ICIC Express Letters
Part B: Applications
Volume **4**, Number **2(tentative)**, April **2013**

ICIC International ©2013 ISSN 2185-2766
pp. 1–ELB12-0914

MINING ROLE BASED ON RANKS

Xiaopu Ma[1], Yan Tian[1], Li Zhao[2] and Ruixuan Li[3]

[1]School of Computer and Information Technology
Nanyang Normal University
No. 1638, Wolong Road, Wolong District, Nanyang 473061, P. R. China
mapxiao@163.com; ttianyan@126.com

[2]Academy of Exploration and Development
Henan Oilfield of Sinopec
Nanyang 473132, P. R. China
zs9950883@163.com

[3]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 September 2012; accepted December 2012

Abstract. *The definition of a good set of roles in order to match the security requirements of a company is a problem partially solved by role engineering techniques, which aids the process to migrate a non-RBAC system to an RBAC system. Unfortunately, all permissions are treated evenly in most previous approaches, thus providing the motivation for this work. In this paper, we generalize this to the case where permissions are given rank to reflect their importance to the system. We further employ a role mining algorithm to generate roles based on ranks which just scans the user permission assignments only once. Experiments on performance study prove the superiority of the algorithm.*
Keywords: Role-based access control, Role engineering, Role mining, Rank

1. **Introduction.** Role-based access control (RBAC) [1] is the most popular access control model at present and widely implemented successfully in a variety of commercial systems. In this security model, permissions are no longer assigned individually to users, but as a set of permissions through roles. This access control model often reduces the complexity of access control because the number of roles in an organization is significantly smaller than that of users [2].

However, in the approach to migrating a non-RBAC system to an RBAC system, a key challenge that has not been adequately addressed so far is how to identify the importance of permission and how to generate roles based on it. In other words, most of the existing bottom-up role engineering approaches treated the permission evenly that did not consider the different nature and importance of the each permission. Hence, the un-frequent roles cannot be found even if they represent the administrative operations. As we know, most of the administrative operations are more sensitive and important than the ordinary ones of general users. The permissions of the administrative operations may be assigned to only a small number of administrators. In this situation, the more important roles will be lost because they only are given to a small number of users although they are very important to the security systems.

The remainder of the paper is organized as follows. We discuss related work in Section 2. The limitations in existing applications for bottom-up role engineering approaches drive our motivation and Section 3 proposes our notation about the original rank between permissions and how to define the reinforced rank between permissions based on the original

rank matrix between permissions. Furthermore, we apply a role mining algorithm based on the reinforced ranks to generate roles that just scans the user-permission assignments only once to study the superiority of our idea. A summary of our experimental results on simulated data is discussed in Section 4. Finally, Section 5 provides some insight into our ongoing and future work.

2. **Related Work.** Role engineering is introduced in 1995 by Coyne which can be divided into two approaches: the top-down and the bottom-up [3]. Under the top-down approach, roles are derived by carefully analyzing each particular business function and then assigning the needed permissions to create roles for each particular business function [4]. These approaches guarantee that all the roles receive their necessary rights so they can perform their functions and no more than their functions. However, since there are often dozens of business processes or tasks and ten thousands of users, it makes top-down approach impracticable in medium to large size enterprises. As a result, the researchers have changed the focus to the bottom-up approach that utilizes the existing user-permission assignments to formulate roles to construct the RBAC system.

Under the bottom-up approach, roles can be generated through user permission assignments. According to their outputs, the bottom-up role mining algorithms can be divided into two categories [5]. The first class algorithms generate a set of candidate roles, and then give 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 algorithm of the first class is CompleteMiner (CM) and FastMiner (FM) [6]. The second class algorithms use Weighted Structural Complexity (WSC) as a common quality measure to generate a complete RBAC state. There are many algorithms belonging to this class, such as ORCA [7], HierarchicalMiner (HM) [8], and GO [9].

However, the traditional role mining approach assumes that the each permission has the same importance without taking account of their rank or weight within user permission assignments. Hence, if a permission set that represents important roles is only given together to a small number of users, it may not be identified by the traditional role mining techniques although these roles are very important.

To this aim, this research tries to assign rank to each permission in order to describe the importance of it in a feasible way. Our focus is on how to calculate the rank and how to generate roles based on the rank. We also employ a role mining algorithm to address the above problem. The experimental results are tested to show the effectiveness of our findings.

3. Mining Role Based on Ranks.

3.1. Preliminaries.

We develop the material in this paper in the context of the NIST standard, the most widely known RBAC model [10]. For the sake of simplicity, we do not consider sessions or separation of duties constraints in this paper.

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

- $USERS, ROLES, PERMS$, *the set of users, roles and permissions respectively;*
- $PA \subseteq ROLES \times PERMS$, *a many-to-many mapping of permission to role assignments;*
- $UA \subseteq USERS \times ROLES$, *a many-to-many mapping of user to role assignment relationships;*
- $UPA \subseteq USERS \times PERMS$, *a many-to-many mapping of user to permission assignment relationships;*
- $auth\_perm(ROLES) = \{p \in PERMS \,|\, (p, ROLES) \in PA\}$, *the mapping of role ROLES onto a set of permissions;*

- *$RH \subseteq ROLES \times ROLES$ is a partial order of $ROLES$ called the role hierarchy or role dominance relation, also written as $\succcurlyeq$, where $r_1 \succcurlyeq r_2$ only if all permissions of r_2 are also permissions of r_1, and all users of r_1 are also users of r_1, i.e., $r_1 \succcurlyeq r_2 \Rightarrow auth\_perm(r_2) \subseteq auth\_perm(r_1)$.*

Here we can use an $m \times n$ binary matrix $MUPA$ to describe the relationships between users and permissions where m is the number of users and n is the number of permissions. 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. We use u_i $(i = 1, \ldots, m)$ to indicate that the ith user and $UserPerms(u_i)$ $(i = 1, \ldots, m)$ to indicate the set of permissions assigned to the ith user, p_j $(j = 1, \ldots, n)$ to indicate that the jth permission and $PermUsers(p_j)$ $(j = 1, \ldots, n)$ to indicate the set of users that possess permission p_j. Since users are represented by permission sets in RBAC, we can give a measure of the original rank between permissions based on this.

Definition 3.2. *Let $UserPerms(u_i)$ $(i = 1, \ldots, m)$ and $UserPerms(u_j)$ $(j = 1, \ldots, m)$ be two sets of permissions. There are three cases to define the original rank between permissions as follows:*

1. $UserPerms(u_i) \subset UserPerms(u_j)$, in which case, the set of permissions assigned to the ith user also belongs to the set of permissions assigned to the jth user; hence, we say that the original rank of permissions in $UserPerms(u_j) - UserPerms(u_i)$ is at least as big as the original rank of permissions in $UserPerms(u_j) \bigcap UserPerms(u_i)$. According to this situation, we will use the notion $p_m \rightarrow p_n(p_m \in UserPerms(u_j) \cap UserPerms(u_i), p_n \in UserPerms(u_j) - UserPerms(u_i))$ to specify that the original rank of the permission p_n is at least as big as the original rank of the permission p_m;

2. $UserPerms(u_i) \cap UserPerms(u_j) = \varnothing$, in which case, there are no same permissions between the set of permissions assigned to the ith user and the set of permissions assigned to the jth user. Based on the opinion of the user u_i, we say that the original rank of the permissions in $UserPerms(u_i)$ is at least as big as the original rank of the permissions in $UserPerms(u_j)$. Otherwise, based on the opinion of the user u_j, we say that the original rank of the permissions in $UserPerms(u_j)$ is at least as big as the original rank of permissions in $UserPerms(u_i)$. We will use the notion $p_m \leftrightarrow p_n(p_m \in UserPerms(u_i), p_n \in UserPerms(u_j))$ to specify the original rank of the permission between p_n and p_m;

3. $UserPerms(u_i) \cap UserPerms(u_j) \neq \varnothing \wedge UserPerms(u_i) \nsubseteq UserPerms(u_j)$, in this situation, there are some permissions assigned to the ith user that also belong to the jth user. We use the notion $p_m \rightarrow p_n(p_m \in UserPerms(u_j) \cap UserPerms(u_i), p_n \in UserPerms(u_j) \cup UserPerms(u_i) - UserPerms(u_j) \cap UserPerms(u_i))$ to specify that the original rank of p_n is at least as big as the original rank of the permission p_m, $p_x \leftrightarrow p_y(p_x \in UserPerms(u_i) - UserPerms(u_j) \cap UserPerms(u_i), p_y \in UserPerms(u_j) - UserPerms(u_j) \cap UserPerms(u_i))$ to specify the original rank of the permission between p_x and p_y.

Before the RBAC system is constructed, the fundamental concepts in access control security are users, permissions and permission to user assignments. Given a set of m users and n permissions and a set of user permission assignments $UPA \subseteq USERS \times PERMS$, based on Definition 3.2, a graph representation of the permissions' original ranks can be shown in Figure 1. For example, the link from p_4 to p_2 means the original rank of the permission p_2 is at least as big as the original rank of permission p_4. Then we can construct the original rank matrix between permissions as follows.

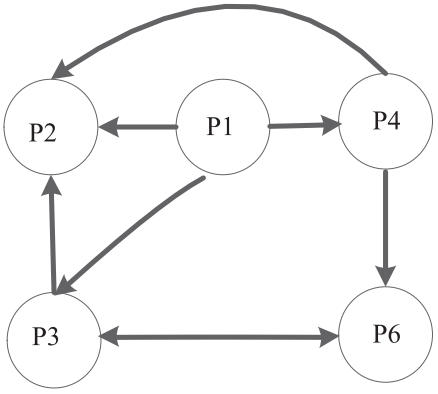

FIGURE 1. Graph representation of the original rank between permissions

Definition 3.3. *The original rank matrix between permissions is defined as*

$$\begin{array}{c} \\ p_1 \\ p_2 \\ \vdots \\ p_n \end{array} \begin{array}{c} \begin{array}{ccc} p_1 & \cdots & p_n \end{array} \\ \begin{pmatrix} edges(p_1,p_1) & \dots & edges(p_1,p_n) \\ edges(p_2,p_1) & \dots & edges(p_2,p_n) \\ \vdots & \ddots & \vdots \\ edges(p_n,p_1) & \dots & edges(p_n,p_n) \end{pmatrix} \end{array}$$

The each element $edges(p_i, p_j)$ $(i = 1, \ldots, n;\ j = 1, \ldots, n)$ in the original rank matrix indicates the sum number of edges from permission p_i to permission p_j (especially when $i = j$, we set $edges(p_i, p_j) = 0$ respectively). Based on this original rank matrix, we can use the techniques that have been used in many real-word domains [11] to compute the reinforced rank rr_{p_i} $(i = 1, \ldots, n)$ over the original rank matrix to improve the quality and utility of the original rank between permissions as follows.

Definition 3.4. *Let p_i be a permission. Then let F_{p_i} be the set of permissions that permission p_i points to and B_{p_i} be the set of permissions that point to p_i. Let $N_{p_i} = |F_{p_i}|$ be the number of links from permission p_i and let c be a factor used for normalization. We define a reinforced rank of p_i as follows.*

$$rr_{p_i} = c \sum_{v \in B_{p_i}} rr_v / N_v$$

Furthermore, for any permission set $PS \in 2^P$, suppose that there are $PS = p_1, p_2, \ldots, p_k$, where $p_i \in P$ $(i = 1, \ldots, k)$. We define the reinforced rank of permission set PS as follows:

Definition 3.5. *The reinforced rank of permission set PS is defined as*

$$rr_{PS} = \sum_{i=1}^{k} rr_{p_i}$$

3.2. **Algorithms.** Fundamentally, almost role mining approaches use an agglomerative technique to find inherent roles through given assigned permissions. In the scope of data mining, many efficient algorithms such as FP-growth, *Apriori* and *Eclat* have been developed to solve this problem on databases containing transactions [12]. However the traditional algorithm *Apriori* depends on the downward closure property which governs that subsets of a large itemset are also large. If we consider the rank of the each permission, this situation is not always true.

According to this disadvantage, we use the candidate permission set to describe the candidate permissions that may be the large in order to generate the frequent permission

Algorithm 1 *RRM* (Rank Role Mining)

Require: $D \equiv (M, N, UP_{M\times N}, rminsup, F, C)$
Require: M, the number of users
Require: N, the number of permissions
Require: $UP_{M\times N}$ represents user-permission assignments
Require: *rminsup*, the ranked support threshold
Require: F represents all the frequent permission sets
Require: C represents all the candidate permission sets
{The Main Procedure}
{Initialization}
Compute original rank between permissions
Compute refined rank between permissions
{Generate the frequent 1-permission sets}
for $(i = 1; i \leq N; i++)$ **do**
  $rsf(p_i) = rr_{p_i} \times numUsers(p_i)/M$
  if $rsf(p_i) \geq rminsup$ **then**
    insert $p_i, rr_{p_i}, PermUsers(p_i)$ into F_1, C_1
  else
{Generate the candidate permission sets}
    $Remove(S, rr_{p_i})$
    $Sort(S)$
    for $(j = 2; j \leq maxSize; j++)$ **do**
      $maxRank = rr_{p_i} + Sum(S, j-1)$
      $minR = \lceil rminsup \times N/maxRank \rceil$
      if $numUsers(p_i) \geq minR$ **then**
        insert $p_i, rr_{p_i}, PermUsers(p_i)$ into C_1
        break
      end if
    end for
  end if
end for
{Generate the frequent k-permission sets}
for $(k = 2; C_k \neq \varnothing, k{+}{+})$ **do**
  $F_k = FPG(C_{k-1}, rminsup)$
end for
$F = \cup_k F_k$
$\{FRG(C_{k-1}, rminsup)\}$
for X and Y are in C_{k-1} **do**
  if first k-2 permissions of X and Y are the same **then**
    $PermUsers(X \cup Y) = PermUsers(X) \cap PermUsers(Y)$
    $rsf(X \cup Y) = (w_X + w_Y) \times numUsers(PermUsers(X \cup Y))/M$
    if $rsf(X \cup Y) \geq rminsup$ **then**
      insert $X \cup Y, rr_{X\cup Y}, PermUsers(X \cup Y)$ into F_k, C_k
    else
      Compute $minR$ and insert the right candidate permission sets into C_k
    end if
  end if
end for
Return F

set. Furthermore, there are many algorithms that can mine association rules based on weights, such as MINWAL (O) and MINWAL (W) [13]. However, these algorithms are time costly because they need scan the database many times to compute the support of each itemsets in order to generate the frequent itemsets. Hence, we employ the algorithm to generate permission set as roles that scans the user permission assignments only once [14]. Algorithm 1 gives the detailed steps about how to find frequent permission sets based on ranks.

The algorithm consists of three phases. The first phase consists of lines 3 and 4. We first compute the original rank between permissions respectively, and then compute the reinforced rank between permissions.

Phase 2 consists of lines 6-23. The for loop in line 6 iterates over all the permissions in the user permission assignments and generates the 1-frequent permission sets and 1-candidate permission sets. Line 7 calculates the ranked support of p_i. Line 9 inserts the 1-frequent permission set into F_1 and C_1. Line 12 to line 19 find the candidate permission set which has not 1-frequent permission. Line 12 removes the rank rr_{p_i} from all the permissions' rank and line 13 sorts the rank sequence in a decreasing order. Line 14 to line 21 find the 1-candidate permission set based on the theory described in [13].

Phase 3 consists of lines 25-27. The for loop in line 25 iterates over the set of all $(k-1)$-candidate permission sets. The frequent permission set generation procedure returns the set of all k-frequent permission sets and k-candidate permission sets. The procedure of FPG consists of lines 30-40. The for loop in line 30 iterates over all the $(k-1)$-candidate permission sets to generate the k-frequent permission sets and k-candidate permission sets. We assume that the permissions in a permission set are ordered by the subscript. Line 32 finds the users containing $X \cup Y$. Line 33 computes the number of users containing $X \cup Y$ and then gets the support rsf. Line 34 to line 38 finds the k-frequent permission set that satisfy the minimum ranked support threshold and also generates the k-candidate permission sets.

4. **Experimental Results.** In this section, we will implement the proposed algorithm of mining roles based on rank on a Pentium (R) Dual-Core CPU 2.5G machine with 2GB memory to evaluate how well our algorithm performs using different metrics like in [15].

Figure 2(a) shows the average number of roles under the various minimum ranked support thresholds. From the figure we can see our algorithm generates different number of roles in the different minimum ranked support thresholds while the $ORCA$ only generates the fixed number of roles because we consider the support (the ratio of the number of users that contains the permissions or permission sets to the number of users) and rank (the importance of the permissions or permission sets) at the same time. Figure 2(b) shows the average search time under different number of permissions. We also can see that our algorithm costs less time. If the number of permissions and users in the user permission assignments is larger, the advantage of our algorithm will be more obvious. Figure 2(c)

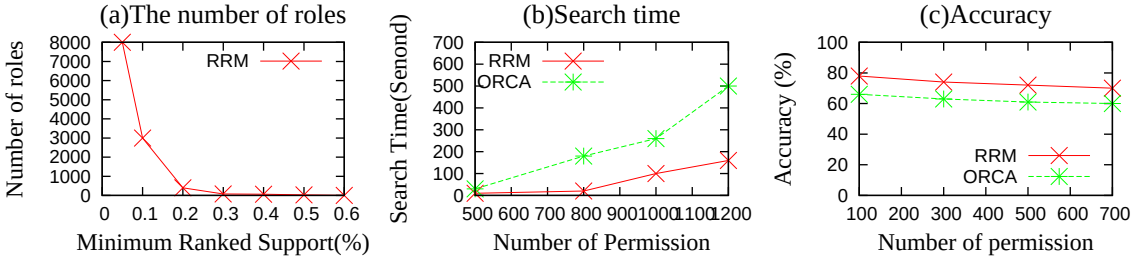

FIGURE 2. Performance and accuracy comparison under the varying number of minimum ranked support and permissions

shows the accuracy of the different algorithms under the different number of permissions. From the figure, we can see that, when the value of the number of permission is low, there is a high accuracy. As the value of the number of permission increases, the accuracy of the results will decrease. We can also see that, the accuracy of the algorithm is quite good, with the largest number of permissions getting approximately to 60%. Hence, it can decrease the administrator's workload when the administrator implements this method to generate roles.

5. **Conclusions and Future Work.** RBAC has become the norm in many applications. Hence, how to create a comprehensive framework for defining the architectural structure of RBAC has become a challenging task before all the benefits of RBAC can be realized. However, if a permission set that represents important roles is only given together to a small number of users, it may not be identified by the traditional role mining techniques although these roles are very important. Hence, in this paper, we present a role mining algorithm to generate roles based on ranks which just scans the user permission assignments only once. We carry out the experiments to illustrate the effectiveness of the proposed techniques. As a result, the proposed approach has superior performance to traditional methods in both speed and generating relevant roles. For the future work, we will evaluate our technique with others especially the role mining algorithm based on weights. Moreover, we will try to introduce the semantic information to further refine the potential roles.

Acknowledgment. This work is supported by National Natural Science Foundation of China under grants 61173170, 60873225 and 70771043, National High Technology Research and Development Program of China under grant 2007AA01Z403, Natural Science Foundation of Hubei Province under grant 2009CDB298, Innovation Fund of Huazhong University of Science and Technology under grants 2011TS135 and 2010MS068, and CCF Opening Project of Chinese Information Processing. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] M. Liu, X. Wang and H. Zhao, Research on illegal information flow in role-based access control model based on petri net, *ICIC Express Letters*, vol.6, no.1, pp.139-144, 2012.

[2] D. Zhang, K. Ramamohanarao, T. Ebringer and T. Yann, Permission set mining: Discovering practical and useful roles, *Proc. of the Annual Computer Security Applications Conference*, Anaheim, CA, USA, pp.247-256, 2008.

[3] E. J. Coyne, Role engineering, *Proc. of the 1st ACM Workshop on Role-Based Access Control*, 1995.

[4] M. Frank, J. M. Buhmann and D. Basin, On the definition of role mining, *Proc. of the 15th ACM Symposium on Access Control Models and Technologies*, Pittsburgh, PA, USA, pp.35-44, 2010.

[5] R. Li, W. Wang and X. Ma, Mining roles using attributes of permissions, *International Journal of Innovative Computing, Information and Control*, vol.8, no.11, pp.7909-7923, 2012.

[6] J. Vaidya, V. Atluri and J. Warner, Roleminer: Mining roles using subset enumeration, *Proc. of the 13th ACM Conference on Computer and Communications Security*, Alexandria, VA, USA, pp.144-153, 2006.

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

[8] 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*, Estes Park, CO, USA, pp.21-30, 2008.

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

[10] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn and R. Chandramouli, Proposed nist standard for role-based access control, *ACM Transactions on Information and System Security*, vol.4, no.3, pp.224-274, 2001.

[11] C. Ding, X. He, P. Husbands, H. Zha and H. Simon, PageRank, HITS and a unified framework for link analysis, *Proc. of ACM SIGIR Conference on Research and Develop in Information Retrieval*, New York, USA, pp.353-354, 2002.

[12] X. Deng, C. Jin, Y. Higuchi and J. C. Han, A novel method for mining frequent itemsets using transaction matrix, *ICIC Express Letters, Part B: Applications*, vol.3, no.5, pp.1147-1155, 2012.

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

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

[15] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang and J. Lobo, Evaluating role mining algorithm, *Proc. of the 14th ACM Symposium on Access Control Models and Technologies*, Stresa, Italy, pp.95-104, 2009.