Role Mining under Role-Usage Cardinality Constraint

John John, Shamik Sural, Vijayalakshmi Atluri, Jaideep Vaidya

Role Mining under Role-Usage Cardinality Constraint

John C. John[1], Shamik Sural[1], Vijayalakshmi Atluri[2], and Jaideep S. Vaidya[2]

[1]School of Information Technology
Indian Institute of Technology, Kharagpur, India
{johnj,shamik}@sit.iitkgp.ernet.in
[2]MSIS Department, Rutgers University, USA
{atluri@cimic,jsvaidya@rbs}.rutgers.edu

Abstract. *With the emergence of Role Based Access Control (RBAC) as the de facto access control model, organizations can now implement and manage many high level security policies. As a means of migration from traditional access control systems to RBAC, different role mining algorithms have been proposed in recent years for finding a minimal set of roles from existing user-permission assignments. While determining such roles, it is often required that certain security objectives are satisfied. A common goal is to enforce the role-usage cardinality constraint, which limits the maximum number of roles any user can have. In this paper, we propose two alternative approaches for role mining with an upper bound on the number of roles that can be assigned to each user, and validate their performance with benchmark data sets.*

Keywords: Role Based Access Control, Constrained Role Mining, Boolean Matrix Decomposition

1 Introduction

In traditional access control mechanisms, a user can access a resource if he has been given appropriate permission on that resource. These user permissions can be represented as a matrix, called User to Permission Assignment (UPA), where the value in each cell (1 or 0) indicates whether a user has the permission or not. In a typical organization with tens of thousands of users and hundreds of thousands of permissions, the size of the UPA matrix could be quite large, making security administration increasingly difficult.

To alleviate the burden of security administration, organizations are migrating to an alternative access control mechanism known as Role Based Access Control (RBAC) [11, 12]. In RBAC, users obtain permissions by virtue of being assigned to roles. A role may be assigned to multiple users. Since the number of roles is orders of magnitude lower than the number of users, RBAC significantly reduces security administration overhead.

For implementing RBAC in an organization currently using traditional UPA based access control, suitable roles have to be formulated. This process of formulation of roles is known as Role Engineering [8]. Generally there are two

approaches to role engineering: top down and bottom up. In the top down approach, business processes are initially divided into independent functional units called job functions. Roles are assigned to these job functions by associating the required permissions. Since the usual number of business processes, job functions and users is quite large, real life implementation of top down approach happens to be difficult, error prone and not quite cost effective. On the other hand, bottom up approaches use the existing user permission assignments. From the UPA, roles with corresponding permissions are derived and users are assigned to these roles. Two binary matrices UA (User to Role Assignment) and PA (Permission to Role Assignment) are thus formed. UA represents which users belong to which roles, and PA represents which permissions are contained in which roles. It is intuitively obvious that the bottom up approach can be conveniently automated. Such an automated procedure for deriving roles from a given UPA matrix, and thereby forming the UA and PA matrices is known as role mining [1, 13].

While there could be many possible decompositions of a given UPA matrix into UA and PA matrices, a useful notion is to obtain a decomposition that minimizes the number of roles, which is the number of columns in the UA matrix and equivalently the number of rows in the PA matrix. This is known as the basic Role Mining Problem (RMP) [1]. RMP is an optimization problem and its decision version has been shown to be NP-Complete.

An important feature of any RBAC system is the ability to impose various constraints [11, 12]. The constraints help to express the organizational security policy, thereby achieving desired security objectives. In many situations, a restriction is imposed on the maximum number of roles that can be played by any user, either due to security restrictions or due to balanced work distribution. We denote this as the *role-usage cardinality* constraint. Since the role mining process is expected to generate organizational roles and corresponding user assignments in an automated way, it is imperative that such a constraint be taken into consideration while mining the roles from a given UPA matrix. Further, from a security administration point of view also, it makes sense to let users achieve their requisite permission through lower number of roles. However, none of the existing work on role mining addresses this issue.

In this paper, we propose two alternative approaches for considering role-usage cardinality constraints while mining the roles. The first approach, called the *Role Priority based Approach* (RPA), as the name suggests, first prioritizes the roles based on their size (i.e., number of permissions within the role) and then limits the number of roles assigned to a user using this priority order. The second approach, called the *Coverage of Permissions based Approach* (CPA) chooses roles by iteratively picking the role with the largest number of permissions that are not yet assigned to that user by any other role, and then imposes the role-usage cardinality constraint. Since both these approaches use different greedy strategies, we have implemented and validated both RPA and CPA using benchmark data sets; our results show that RPA fares significantly better than CPA.

The rest of this paper is organized as follows. In Section 2, we present our proposed approaches with detailed algorithms. In Section 3, the results of running the two proposed algorithms on different datasets are presented and analyzed. In Section 4, we review the related work on role mining. Finally, Section 5 concludes this paper and provides directions for future work.

2 Role Mining with Role-usage Cardinality Constraints

In this section, we present our two proposed approaches: the Role Priority based Approach (RPA) and Coverage of Permissions based Approach (CPA), which mine the minimum set of roles from a UPA matrix under the constraint that the number of roles assigned to any user cannot exceed a specified upper bound.

We start off with some notation. First, as discussed earlier, UA, PA and UPA are all boolean matrices. Assume that there are n users and m permissions, with q candidate roles. Thus, UA is $n \times q$, PA is $q \times m$ and UPA is $n \times m$. Let c_{ij} represent the UA matrix entry for user i ($i = 1 \dots n$) and role j ($j = 1 \dots q$), and let r_{jt} represent the PA matrix entry for role j ($j = 1 \dots q$) and permission t ($t = 1 \dots m$). Finally, let x_{it} represent the UPA matrix entry for user i and permission t. Then, for any correct decomposition, the following conditions must be met [2]: $\sum_{j=1}^{q} c_{ij} r_{jt} \geq 1$ for $x_{it} = 1$ and $\sum_{j=1}^{q} c_{ij} r_{jt} = 0$ for $x_{it} = 0$.

Given the above, the role-usage cardinality constraint can be represented as $\sum_{j=1}^{q} c_{ij} \leq maxcount, for 1 \leq i \leq n$.

2.1 Role Priority based Approach (RPA)

The Role Priority based approach prioritizes roles by their size (i.e., the number of permissions in the role). Now, roles are assigned to each user by picking the roles whose permissions are a subset of the permissions required by that user, and are not redundant (i.e., the permission has not already been assigned through another prior-picked role). While any set of candidate roles could be used, in this paper, we use the candidate roles generated from the UPA by the greedy strategy for optimal boolean matrix decomposition (referred to as OBMD) proposed by Liu et al. [2]. Note that OBMD can produce redundant roles for some of the users. A role r is considered redundant (for a user i) if the permissions of r form a subset of the permissions of one or more other roles of the same user i. This is because the greedy approach only eliminates those candidate roles which are not used by any of the users. OBMD does not check the redundancy of roles for each user. Therefore, we eliminate such redundant roles in the candidate roles generated by OBMD. Also, the roles generated by Fast Miner [5], which OBMD uses as base, do not cover the permissions which are exclusively assigned to a user. To cover these permissions, separate roles are generated. The roles are then prioritized based on the number of permissions contained in each role. To restrict the number of roles of each user upto the specified upper bound, the roles are selected in the order of their priority. If all of the permissions of the user are not covered after selecting $maxcount - 1$ roles, where $maxcount$ is the specified

upper bound, the remaining uncovered permissions are formed into a separate role.

These steps are summarized in Algorithm 1. Formation of initial UA matrix and the associated PA matrix using OBMD is done in Line 1. Then, a Role Priority (RP) list is created in which roles are maintained in the descending order of the number of permissions associated with each role (Line 2). The variable *maxcount* denotes the maximum number of roles allowed for a user and *rolecount* denotes the number of roles so far assigned to a user. In lines 4-12, for each user, the roles are selected one by one as per the order in the RP list until all permissions of the user are covered or until the *rolecount* becomes $maxcount - 1$. If the permissions of the selected role form a subset of already covered permissions of the user, they are redundant and are eliminated during the process (Lines 7-8). If the *rolecount* value reaches $maxcount - 1$ and all the permissions of the user are not yet covered, a new role r is generated comprising of all the uncovered permissions (if it is not an existing role; otherwise, uses the existing role r) and includes r in the PA matrix, also setting the corresponding c_{ir} cell of the UA matrix to 1 (Lines 13-23).

Table 1. Sample input data set (in the form of UPA matrix)

	p1	p2	p3	p4	p5
u1	1	0	0	0	1
u2	0	0	1	1	0
u3	1	0	1	1	0
u4	1	1	1	1	1
u5	0	0	1	1	0
u6	1	1	0	0	0

Table 2. Initial UA and PA matrices generated by Fast Miner

	r1	r2	r3	r4	r5
u1	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}
u2	c_{21}	c_{22}	c_{23}	c_{24}	c_{25}
u3	c_{31}	c_{32}	c_{33}	c_{34}	c_{35}
u4	c_{41}	c_{42}	c_{43}	c_{44}	c_{45}
u5	c_{51}	c_{52}	c_{53}	c_{54}	c_{55}
u6	c_{61}	c_{62}	c_{63}	c_{64}	c_{65}

	p1	p2	p3	p4	p5
r1	1	0	0	0	0
r2	1	0	0	0	1
r3	0	0	1	1	0
r4	1	1	0	0	0
r5	1	0	1	1	0

Illustrative Example We now consider an illustrative example given in Table 1 for explaining both the OBMD and RPA algorithms. OBMD employs the Fast Miner algorithm to generate q number of candidate roles and to form the binary

Algorithm 1 Role Priority based Algorithm (RPA)

1: Use the OBMD algorithm to form the decomposition into UA and PA matrices
2: Prepare Role Priority (RP) list by ordering the roles in descending order of the number of permissions in the role
3: **for** each user i **do**
4: Set $rolecover = null$ and $rolecount = 0$
5: **while** $rolecover$ does not contain all permissions of i and $rolecount \neq maxcount - 1$ **do**
6: Find the next role, k, in the RP list whose permissions are a subset of the permissions of user i
7: **if** $k \subseteq rolecover$ **then**
8: Set $c_{ik} = 0$
9: **else**
10: $rolecover = rolecover \cup \{\text{permissions of } k\}$ and Set $rolecount = rolecount + 1$
11: **end if**
12: **end while**
13: Set $D =$ Permissions of i - union of permissions of all roles of i
14: **if** $D = \phi$ **then**
15: Set all remaining roles of i to be 0
16: **else**
17: **if** $D =$ existing role r in the PA matrix **then**
18: Set $c_{ir} = 1$ and Set all remaining roles of i to 0
19: **end if**
20: **else**
21: Generate a new role r with all permissions of D
22: Include r in the PA and UA matrices(set $c_{ir} = 1$) and Set all remaining roles of i to 0
23: **end if**
24: **end for**

Table 3. Final UA and PA matrices obtained from OBMD

	r1	r2	r3	r4
u1	1	1	0	0
u2	0	0	1	0
u3	1	0	1	0
u4	1	1	1	1
u5	0	0	1	0
u6	1	0	0	1

	p1	p2	p3	p4	p5
r1	1	0	0	0	0
r2	1	0	0	0	1
r3	0	0	1	1	0
r4	1	1	0	0	0

PA matrix. The initial UA and PA matrices consisting of five candidate roles generated through Fast Miner are shown in Table 2.

The UA matrix is formed by selecting appropriate roles from the PA matrix based on the count of the conditions being satisfied while selecting a role. Some of the candidate roles which are not used by any of the users can finally be rejected, thereby reducing the total number of roles. The final matrices after applying the appropriate conditions as mentioned above are given in Table 3.

Assume the value of the cardinality constraint (*maxcount*) is 2. In Step 1, the OBMD algorithm gives an output as shown in Table 3. The RP list is prepared as {$r2$, $r3$, $r4$, $r1$}. RPA then selects the appropriate roles from the list for each user. For user $u1$, role $r2$ is selected. With this role all permissions of $u1$ are covered. Similarly, role $r3$ is found for user $u2$. The first role selected for user $u3$ is role $r3$. Now the value of *rolecount* becomes 1 which is equal to $maxcount - 1$. It can be observed that the remaining set of uncovered permissions is the same as the existing role $r1$, and hence, $r1$ is selected. For user $u4$, role $r2$ is initially selected. Again the value of *rolecount* becomes equal to $maxcount - 1$. Now the uncovered permissions are not the same as any existing role and so a new role $r5$ is created with all the uncovered permissions. For user $u5$, the algorithm initially selects role $r4$. With this role all permissions of the user are covered. Thus, the algorithm finds a total of 5 roles for a *maxcount* value of 2. The final UA and PA matrices are shown in Table 4.

It may be noted that the OBMD algorithm generates four roles as shown in Table 3. But user $u4$ has four roles which exceeds *maxcount*. The decomposition done by RPA restricts the number of roles of users to the specified upper bound. The total number of roles is increased to five to satisfy this constraint.

Table 4. UA and PA matrices generated by RPA

	r1	r2	r3	r4	r5
u1	0	1	0	0	0
u2	0	0	1	0	0
u3	1	0	1	0	0
u4	0	1	0	0	1
u5	0	0	1	0	0
u6	0	0	0	1	0

	p1	p2	p3	p4	p5
r1	1	0	0	0	0
r2	1	0	0	0	1
r3	0	0	1	1	0
r4	1	1	0	0	0
r5	0	1	1	1	0

2.2 Coverage of Permissions based Approach (CPA)

The second approach, called the Coverage of Permissions based Approach (CPA), selects roles based on the number of permissions that are as yet unassigned to a user, while still enforcing the cardinality constraint. Fast Miner is employed to form the initial set of candidate roles, similar to that shown in Table 2. Effectively, it works on decompositions containing all candidate roles. In order

to limit the number of roles of each user upto the specified bound, the algorithm keeps on finding roles which contain the largest number of uncovered permissions (permissions not yet assigned through any role) for the user. If all permissions of the user are not covered after selecting $maxcount - 1$ roles, the remaining uncovered permissions are grouped into a separate role. The redundant roles are also eliminated during the process.

The steps are summarized in Algorithm 2. Fast Miner is used to generate q candidate roles and form the initial PA matrix and the UA matrix in Line 1. Next, c_{ij} values of the UA matrix are set to 0 according to the condition $\sum_{j=1}^{q} c_{ij} r_{jt} = 0$ (Line 2). For each user i, in each step, the algorithm finds a role which covers the maximum number of uncovered permissions of i from among all the roles r where $c_{ir} = 1$. The process is repeated until all the permissions of the user are covered or *rolecount* becomes $maxcount - 1$ (Lines 4-8). If the *rolecount* is $maxcount - 1$ and all the permissions of user i are not yet covered, a new role r is generated with the uncovered permissions (if it is not an existing role; otherwise, the existing role r is used). Finally, r is included in the PA matrix and the corresponding c_{ir} of the UA matrix is set to 1 (Lines 9-20).

Algorithm 2 Coverage of Permissions based Algorithm (CPA)

1: Generate candidate roles by taking intersection of the permissions of every pair of users. Form PA matrix and associated UA matrix. (Fast Miner approach)
2: Set the required cells of c_{ij} to 0 as per $\sum_{j=1}^{q} c_{ij} r_{jt} = 0$ for all $x_{it} = 0$
3: **for** each user i **do**
4: Set $rolecover = null$ and $rolecount = 0$
5: **while** *rolecover* does not contain all permissions of i and $rolecount \neq maxcount - 1$ **do**
6: Find the role k that contains the maximum number of permissions of i which are not yet contained in *rolecover*, while not adding any unallowed permissions to i
7: Set c_{ik} to be 1, Set $rolecover = rolecover \cup \{\text{permissions of } k\}$ and Set $rolecount = rolecount + 1$
8: **end while**
9: Set D = Permissions of i - union of permissions of all roles of i
10: **if** $D = \phi$ **then**
11: Set all remaining roles of i to be 0
12: **else**
13: **if** D = existing role r in the PA matrix **then**
14: Set $c_{ir} = 1$ and Set all remaining roles of i to 0
15: **end if**
16: **else**
17: Generate a new role r with all permissions of d
18: Include r in the PA and UA matrices
19: Set $c_{ir} = 1$ and Set all remaining roles of i to 0
20: **end if**
21: **end for**

Illustrative Example Consider again the same dataset given in Table 1 and assume the value of cardinality constraint to be 2. The initial decomposition is as given in Table 2. In the next step, the CPA algorithm finds the cells of UA matrix which are to be set to 0 using the condition $\sum_{j=1}^{q} c_{ij} r_{jt} = 0$. Then, for each user, it finds roles which cover maximum number of uncovered permissions. User $u1$ has two roles from which the algorithm selects role $r2$ since it covers more permissions of $u1$ as compared to role $r1$ and with that role all permissions of $u1$ are covered. So it sets the remaining cells of $u1$ in the UA matrix to 0. Similarly, user $u2$ gets role $r3$ and user $u3$ gets role $r5$. User $u4$ is assigned to role $r5$ initially. Now the *rolecount* becomes equal to $maxcount - 1$. So a new role $r6$ is formed with all uncovered permissions. The new role is included in the UA matrix and c_{61} is set to 1. Then user $u5$ gets role $r3$ and finally user $u6$ gets role $r4$. The unused role $r1$ is removed. The algorithm completes execution after mining five roles. The final decomposition matrices are shown in Table 5 (after renaming roles from $r2$ to $r6$ as $r1$ to $r5$).

Table 5. UA and PA matrices generated by CPA

	r1	r2	r3	r4	r5
u1	1	0	0	0	0
u2	0	1	0	0	0
u3	0	0	0	1	0
u4	0	0	0	1	1
u5	0	1	0	0	0
u6	0	0	1	0	0

	p1	p2	p3	p4	p5
r1	1	0	0	0	1
r2	0	0	1	1	0
r3	1	1	0	0	0
r4	1	0	1	1	0
r5	0	1	0	0	1

As is evident, from the above discussions, RPA and CPA use different greedy choices. The output decompositions in the examples are also different. But the number of roles obtained is the same in the two cases for the example dataset even though the actual roles are different. In the next section, we give a comparative performance of both these algorithms with benchmark datasets.

3 Experimental Results

Algorithms 1 (RPA) and 2(CPA) were executed on a number of datasets as listed in Table 6. These are real-world datasets used to evaluate the work reported in [9]. Table 6 also provides the sizes of these datasets in terms of the number of users, number of unique permissions as well as the permission size, which is the total number of permissions added over all users.

We plot the variation of the number of roles (y-axis) generated by RPA and CPA with varying constraint value (x-axis). The roles discovered by both algorithms satisfy the given constraint. The plots are shown in Figures 1 to 3.

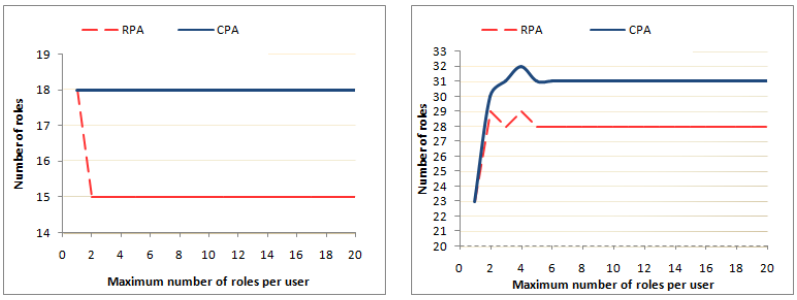
Table 6. Data sets

Dataset	Users	Permissions	Total permission size
Healthcare	46	46	1,486
Domino	79	231	730
EMEA	35	3,046	7,220
Firewall 1	365	709	31,951
Firewall 2	325	590	36,428

The number of roles generated by RPA is less than or equal to that of CPA in all datasets. By observing the output for the datasets Healthcare and Firewall 2, it can be concluded that the number of generated roles decreases when the value of the constraint increases. The number of roles generated by OBMD in the Healthcare dataset is 17. But RPA generates only 15 roles when the value of the constraint is greater than or equal to 2. When the constraint value is set to 1, it generates 18 roles. On the contrary, CPA gives 18 roles for all values of the constraint. Similarly, for Firewall 2 dataset, OBMD generates 10 roles and RPA produces the same number of roles when the constraint value remains greater than or equal to 2. It generates 11 roles for a constraint value of 1. Here also CPA generates 11 roles for all cases.

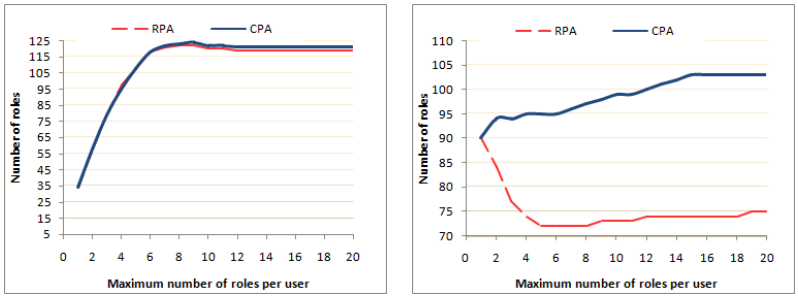
For the datasets Domino and EMEA, the reverse occurs. There the number of roles generated shows a decrease when the value of the constraint decreases. This appears to be contradictory. However, the reason is that the candidate roles generated by the Fast Miner algorithm in those datasets do not contain roles which produces optimum result. Also, presence of many exclusive permissions in the dataset, results in increased number of roles. Such exclusive roles are not part of the roles generated by Fast Miner. For the Domino dataset, OBMD produces 29 roles. However, the optimal number is 23, which is produced by both RPA and CPA algorithms for the constraint value 1. Similarly for the EMEA dataset, the number of roles generated by OBMD is 122 but the optimal number is 34 which is produced by both the proposed algorithms for the constraint value 1. Thus, OBMD does not give optimal number of roles for these datasets. In both of our approaches, when the value of the constraint is reduced and it approaches a value of 1, many of the roles get merged with other roles to form a new role. As a result, in these datasets, the algorithms give optimal result when the value of the constraint is low. An improvement in the Fast Miner algorithm would give a better candidate role generation.

For the Firewall 1 dataset, OBMD produces 75 roles. RPA gives the optimal result for constraint values greater than or equal to 4 and the minimum number of roles generated by this algorithm is 72. When the value of the constraint is less than 4, the number of roles generated by RPA increases and for a value of 1 it produces the highest number of roles, which is 90. However, for this dataset, CPA never finds an optimal result. It produces minimum number of roles (90 roles) for the constraint value 1.



(a) (b)

Fig. 1. Number of roles generated for (a) Healthcare dataset and (b) Domino dataset



(a) (b)

Fig. 2. Number of roles generated for (a) EMEA dataset and (b) Firewall 1 dataset

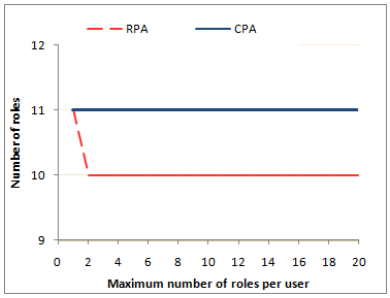


Fig. 3. Number of roles generated for Firewall 2 dataset

4 Related Work

Several attempts have been made so far for bottom-up identification of roles in RBAC. Schlegelmilch and Steffens [4] introduced a role mining tool named ORCA which forms a hierarchy of permission clusters. A subset enumeration technique is used in [5]. After clustering the users having similar permissions, it intersects all possible combinations of user's permissions and generates a set of candidate roles. These roles are then prioritized based on the number of users sharing the roles. The work in [1] maps RMP into an equivalent problem – the Largest Uncovered Tile Mining (LUTM) problem, which uses the concept of database tiling [6]. Generating minimum number of roles in RMP is equivalent to finding the minimum number of tiles in the tiling problem. Zhang et al. [8] use a graph optimization technique to form role hierarchies. In [2], Lu et al. model the problem into optimal Boolean matrix decomposition problem. In order to generate practical roles, [7] ignores some permission assignments. Authors of [9] proposed an algorithm which uses bipartite graph formulation to find approximately minimum number of roles by forming biclique cover of edges of the graph.

Little work has been done on RMP with cardinality constraints. Kumar et al. [10] considered RMP with a constraint which limits the maximum number of permissions that a role can have. They use a combination of clustering and constrained permission set mining to generate roles. Recently, Hingankar and Sural [3] have worked towards role mining with a constraint on the maximum number of users in a role. They use the biclique cover approach to arrive at the solution. However, there is no available literature on RBAC role mining with an upper bound on the number of roles a user can have.

5 Conclusions and Future Direction

In this paper, we consider the problem of constrained role mining, by enforcing the limit on the number of roles a user can have. We have presented two different approaches to enforce the role-usage cardinality constraint. Variation in the number of generated roles by varying the value of the constraint has been studied on real-world datasets for both these approaches.

Organizations may impose restriction on multiple RBAC parameters at the same time. The parameters could be upper bounds on number of roles to a user, number of permissions in a role, number of roles in a permission, etc. While we consider a single constraint in this paper, the combined effect on the mining process when many constraints are considered simultaneously needs be studied. The work can be extended to find an optimal solution to the role mining problem with such multiple constraints.

6 Acknowledgement

This work is partially supported by a research grant from the Department of Science and Technology, Government of India, under Grant No. SR/S3/EECE/082/

2007. The work of Atluri is supported in part by the National Science Foundation under grant CNS-0746943.

References

1. Jaideep Vaidya, Vijayalakshmi Atluri and Qi Guo: The Role Mining Problem: Finding a Minimal Descriptive Set of Roles. In: Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 175-184. (2007)
2. Haibing Lu, Jaideep Vaidya and Vijayalakshmi Atluri: Optimal Boolean Matrix Decomposition: Application to Role Engineering. In: Proceedings of the IEEE 24th International Conference on Data Engineering, pp. 297-306. (2008)
3. Manisha Hingankar and Shamik Sural : Towards Role Mining with Restricted User-Role Assignment. In: Proceedings of the 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), pp. 1 -5. (2011)
4. Schlegelmilch Jürgen and Steffens Ulrike: Role mining with ORCA. In: Proceedings of the tenth ACM symposium on Access control models and technologies, pp. 168–176. (2005)
5. Jaideep Vaidya, Vijayalakshmi Atluri and Janice Warner: Role Miner: Mining Roles using Subset Enumeration. In: Proceedings of the 13th ACM conference on Computer and communications security, pp. 144-153. (2006)
6. Floris Geerts, Bart Goethals and Taneli Mielikinen: Tiling Databases. In: Discovery Science, pp. 278-289. Springer (2004)
7. Dana Zhang, Ramamohanarao Kotagiri and Ebringer Tim and Yann Trevor: Permission Set Mining: Discovering Practical and Useful Roles. In: Computer Security Applications Conference, ACSAC , pp. 247-256. (2008)
8. Dana Zhang, Ramamohanarao Kotagiri and Ebringer Tim: Role Engineering using Graph Optimization. In: Proceedings of the 12th ACM symposium on Access control models and technologies, pp. 139-144. (2007)
9. Ene Alina, Horne William, Milosavljevic Nikola, Rao Prasad, Schreiber Robert and Tarjan Robert E.: Fast Exact and Heuristic Methods for Role Minimization Problems. In: Proceedings of the 13th ACM Symposium on Access control Models and Technologies, pp. 1-10. (2008)
10. Ravi Kumar, Shamik Sural and Arobinda Gupta: Mining RBAC Roles under Cardinality Constraint. In: Proceedings of the International Conference on Information Systems Security. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 6503, pp. 171-185 (2010)
11. Ravi S. Sandhu , Edward J. Coyne , Hal L. Feinstein and Charles E. Youman: Role Based Access Control Models. IEEE Computer Society Press, pp. 38-47 (1996)
12. David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn and Ramaswamy Chandramouli: Proposed NIST Standard for Role-Based Access Control. In: ACM Transactions on Information and System Security 4(3) (2001)
13. Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin B. Calo, Jorge Lobo: Mining Roles with Multiple Objectives. In: ACM Transactions on Information and System Security 13(4), 36 (2010)