Towards Role Mining with Restricted User-Role Assignment

Manisha Hingankar
School of Information Technology
Indian Institute of Technology
Kharagpur, India 721302
Email: hngnkrp@sit.iitkgp.ernet.in

Shamik Sural
School of Information Technology
Indian Institute of Technology
Kharagpur, India 721302
Email: shamik@sit.iitkgp.ernet.in

***Abstract*—RBAC (Role-Based Access Control) is a widely adopted access control model. However, migrating to RBAC requires identification of a complete and correct set of roles. Many bottom-up algorithms have been proposed for mining such roles. These algorithms attempt to find a minimal set of roles that is consistent with a given input configuration. But none of these consider any form of constraint on the maximum number of users that can be assigned to each role. This is a common requirement often termed as 'cardinality constraint' which has been widely referred in early RBAC literature. In this paper, we propose three algorithms which identify roles that satisfy such type of constraint. The algorithms vary in their degree of complexity and accuracy. Comparative performance analysis has been done using benchmark datasets.**

Keywords: RBAC, Role Engineering, Role Mining, Constraint

I. Introduction

Access control mechanisms allow organizations to grant or deny access of system resources to its users. In traditional access control methods, a UPA (User-Permission Assignment) matrix is given in which rows represent users and columns represent permissions. A '1' in a particular cell denotes that the user is assigned the corresponding permission.

In Role-Based Access Control (RBAC) systems, users are assigned to roles and permissions are granted to roles. Users acquire permissions by being members of roles [2]. As the number of roles is always less than the number of users, security administration is more convenient with RBAC. It also has other advantages. For example, if within an organization a user's work profile changes, the user can be assigned to the new role and removed from the old one. In the absence of an RBAC system, all the old permissions assigned to the user would have to be revoked and new permissions granted. The process of defining roles and associating permissions with them is known as role-engineering [5]. The task of role-engineering can be carried out in two ways: top-down and bottom-up [4].

In top-down approach, business processes are analyzed and decomposed into smaller functional units [5]. A set of permissions is associated with each functional unit and a role is defined for it. But this is a time consuming process. In contrast, bottom-up approaches identify roles from a given UPA matrix. This technique is named as role mining. Two matrices namely, UA (User Assignment) and PA (Permission Assignment) are derived. The UA matrix defines user-to-role assignments while PA defines permission-to-role assignments. As the bottom-up role mining process can be automated, it is preferred over top-down approaches.

Formally, the role mining problem can be defined as follows: For an input set of users U, a set of permissions P, and a user-permission assignment relation UPA, determine the smallest set of roles R, together with a set of user role assignments UA and a set of role permission assignments PA such that user $u \in$ U has permission $p \in$ P iff there is role $r \in$ R such that both (u, r) $\in$ UA and (r, p) $\in$ PA as well as (u, p) $\in$ UPA [1], [6]. We implemented the greedy algorithm proposed in [1] and ran it on different datasets. In many cases, it was found that large number of users got assigned to individual roles. This makes RBAC administration quite difficult. Therefore, an organization may wish to restrict the maximum number of users that can be assigned to any role. Existing role mining algorithms do not address this problem. In this paper, we propose a number of algorithms which perform role mining with such restrictions on user-role assignment cardinality.

The rest of this paper is organized as follows. In Section II, we review some of the existing bottom-up role mining algorithms. In Section III, we briefly describe a graph-based greedy algorithm suggested for role mining in [1]. In Section IV, we propose three approaches that consider restriction imposed on the cardinality of user assignment to role. The results of running our algorithms on different benchmark datasets are presented in Section V. Finally, we draw conclusions in Section VI.

II. Related work

Over the last few years, many bottom-up role mining algorithms have been proposed such as RoleMiner: Mining Roles using Subset Enumeration [5], Role Mining with ORCA [3], Role Engineering using Graph Optimization [7], etc. In [5], roles are identified via subset enumeration. The algorithm computes all roles of interest. Roles are then prioritized based on counts of users who have the permission set of discovered roles. Schlegelmilch and Steffens [3] use clustering analysis to form clusters of sets of permissions. Each cluster corresponds to a role. In another approach, a graph optimization technique is proposed to define a set of roles [7]. The basic role mining problem has been mapped to the Database Tiling problem by Vaidya et al. [6]. As the tiling problem is shown to be equivalent to the basic role mining problem, the algorithms developed for Minimum Tiling can be used to find approximately minimum number of roles. In [6], first the minimum tiling for UPA matrix is determined and then the tiles are converted into roles. The algorithm 'Role Mining Via Biclique Covers' given in [1] attempts to find approximately minimum number of roles by finding biclique cover of edges of a bipartite graph.

All the above-mentioned approaches map the role mining problem to an unconstrained optimization problem. In a recent attempt, Kumar et al. [8] proposed an algorithm for mining roles with cardinality constraint on the maximum number of permissions that a role can have. However, such a constraint often overloads the user with too many roles. For example, consider that a user is granted ten permissions in the UPA matrix while the constraint defined is a maximum of two permissions per role. In such a situation, at least five roles have to be assigned to that user. The case is worse when only one permission is limited to the role which leads to assignment of ten roles to a single user in this example. This means that the number of roles assigned to the users increases as the maximum number of permissions that can be assigned to the individual roles decreases. With relatively large number of roles, RBAC administration becomes difficult in these cases. If we keep a restriction on the number of users assigned to any given role, then the set of roles identified would be different from the ones found by the above approach. But administration of the roles becomes easy and convenient if we incorporate such types of constraint. Further, such a constraint is more natural in organizations where there are often a maximum number of persons who can belong to a given role like the number of directors, number of managers, number of contract employees, etc.

Since we consider a graph based approach for our proposed constrained role mining algorithms, we explain one of the existing unconstrained role mining problems that uses biclique covers. This will enable better understanding of the new approach.

III. Role Mining Via Biclique Covers

The UPA matrix can be represented as a bipartite graph in which the users and the permissions form two vertex sets. There is an edge between any user and a permission if that user is assigned the corresponding permission in UPA. The basic role minimization problem can be mapped to the problem of finding minimum biclique cover of the edges of a bipartite graph [1]. The greedy algorithm proposed in [1] identifies and includes one biclique at a time in the cover until all edges are covered. The algorithm is briefly described below.

A. Greedy Algorithm

In this algorithm, bicliques are constructed as follows.

- Take a vertex v with minimum number of uncovered edges.
- If v is user, find all permissions assigned to him and then find all other users who also have been assigned all of these permissions.
- If v is permission, find all users who have been assigned this permission and then all permissions that all of these users have been assigned.

IV. Role Mining with constraint

It is observed that the roles found out by using the above greedy algorithm contain large number of users assigned to them. This makes role administration more difficult. In this section, we propose three algorithms that limit the number of users assigned to any role. All of these algorithms find biclique cover of the edges of a bipartite graph. But they use different strategies to identify biclique.

A. Algorithm 1

In Algorithm 1, bicliques are identified using the following strategy.

- User with minimum number of uncovered incident edges is selected.
- All permissions assigned to him are determined.
- Next find all other users who also have been assigned all of these permissions.
- Arrange all of these users in descending order of their uncovered incident edges.
- If the constraint defined is r then
  - If the users found out in the above step are less than or equal to r then select all users.
  - Else, select first r users.

B. Algorithm 2

We have modified the greedy algorithm described in Section III so as to satisfy the given constraint. The modified algorithm finds bicliques in the following way.

- A vertex with minimum number of uncovered edges is selected.
- If the selected vertex is user, we find all his permissions and then find the users one by one who have all of these permissions till we get the number of users defined by constraint.
- If the selected vertex is permission, we find the number of users so as not to exceed the limit specified by constraint and then find permissions that all of these users have.

C. Algorithm 3

The third algorithm finds bicliques in the following manner.

- Permission with minimum number of uncovered incident edges is selected.
- Find all users that have this permission. Let this number be n.
- If n is less than the number defined by constraint then find all permissions that all of these users have.
- Else, find all combinations of r users out of n users.

- Find other permissions that are possesed by all *r* users for each combination.
- Select the combination of *r* users so that maximum number of permissions are covered.

D. An Example

TABLE I
SAMPLE DATASET

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

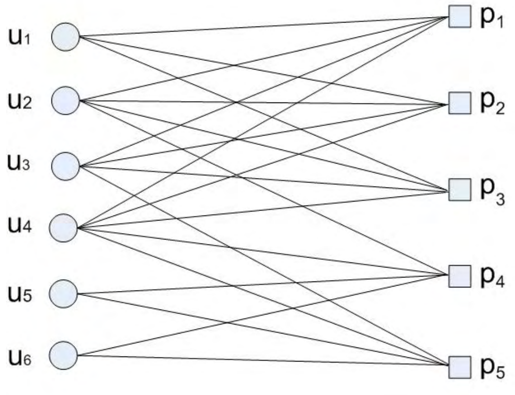

Fig. 1. Bipartite Graph representation of Sample Dataset

TABLE II
USERS AND PERMISSIONS WITH THEIR NUMBER OF UNCOVERED INCIDENT EDGES

user	perm-count	perm	user-count
u_1	3	p_1	4
u_2	4	p_2	4
u_3	4	p_3	4
u_4	5	p_4	4
u_5	2	p_5	4
u_6	2	-	-

We consider the sample dataset given in Table I to explain the above three algorithms in a step by step manner. u_1, u_2, u_3, u_4, u_5, u_6 represent users. p_1, p_2, p_3, p_4, p_5 represent permissions. '1' in i^{th} row and j^{th} column means user *i* is assigned permission *j*. Fig. 1 shows bipartite graph representation of the given sample dataset. When executed on this dataset, all three of the proposed algorithms first find a vertex with minimum number of uncovered incident edges. So we have shown the number of uncovered incident edges of each user and permission in Table II. Number of uncovered incident edges is represented by perm-count for the users and by user-count for the permissions. All iterations of the three algorithms are shown in Tables III, IV and V. Here we consider the value of cardinality constraint as 2.

TABLE III
ITERATIONS OF ALGORITHM 1

Algorithm 1	
1^{st} iteration	1. Found user u_6 with minimum number of uncovered incident edges. 2. Permissions p_4, p_5 assigned to u_6 are selected. 3. Users u_4, u_5 having the same permissions as u_6 are found. 4. We selected one user u_4 having maximum number of uncovered incident edges as constraint defined is 2. 5. A role is defined with users u_4, u_6 and permissions p_4, p_5.
2^{nd} iteration	1. Found user u_5 with minimum number of uncovered incident edges. 2. Permissions p_4, p_5 assigned to u_5 are selected. 3. No user has the same permissions as u_5 has. 4. A role is defined with user u_5 and permissions p_4, p_5.
3^{rd} iteration	1. Found user u_4 with minimum number of uncovered incident edges. 2. Permissions p_1, p_2, p_3 assigned to u_4 are selected. 3. Users u_1, u_2, u_3 having the same permissions as u_4 are found. 4. User u_2 is selected. 5. A role is defined with users u_2, u_4 and permissions p_1, p_2, p_3.
4^{th} iteration	1. Found user u_2 with minimum number of uncovered incident edges. 2. Permission p_4 assigned to u_2 is selected. 3. No user has the same permission as u_2 has. 4. A role is defined with user u_2 and permission p_4.
5^{th} iteration	1. Found user u_1 with minimum number of uncovered incident edges. 2. Permissions p_1, p_2, p_3 assigned to u_1 are selected. 3. User u_3 having the same permissions as u_1 is found. 4. A role is defined with users u_1, u_3 and permissions p_1, p_2, p_3.
6^{th} iteration	1. Found user u_3 with minimum number of uncovered incident edges. 2. Permission p_5 assigned to u_3 is selected. 3. No user has the same permission as u_3 has. 4. A role is defined with user u_3 and permission p_5.

In this example, we get six roles with Algorithm 1 and five with Algorithm 2, while Algorithm 3 gives four roles, which is the least among all three algorithms proposed in this paper. However, since we find the best combination of users so that maximum number of permissions are covered, this algorithm takes more time as compared to the first two algorithms. Consider a situation in which a particular permission with minimum number of uncovered incident edges is selected according to Algorithm 3. Let the constraint defined be 3. Suppose there are ten users having this permission then we get ${}^{10}C_3$ possible combinations of users. Hence the time required is high. Due to this reason, only the first two algorithms are of practical use.

TABLE IV
ITERATIONS OF ALGORITHM 2

Algorithm 2	
1st iteration	1. Found user u_5 with minimum number of uncovered incident edges. 2. Permissions p_4, p_5 assigned to u_5 are selected. 3. User u_4 having the same permissions as u_5 is found. 4. No further search for user is done as the constraint defined is 2. 5. A role is defined with users u_4, u_5 and permissions p_4, p_5.
2nd iteration	1. Found user u_6 with minimum number of uncovered incident edges. 2. Permissions p_4, p_5 assigned to u_6 are selected. 3. No user has the same permissions as u_6 has. 4. A role is defined with user u_6 and permissions p_4, p_5.
3rd iteration	1. Found permission p_4 with minimum number of uncovered incident edges. 2. User u_2 having permission p_4 is selected. 3. Permissions p_1, p_2, p_3 assigned to u_2 are selected. 4. A role is defined with user u_2 and permissions p_1, p_2, p_3, p_4.
4th iteration	1. Found permission p_5 with minimum number of uncovered incident edges. 2. User u_3 having permission p_5 is selected. 3. Permissions p_1, p_2, p_3 assigned to u_3 are selected. 4. A role is defined with user u_3 and permissions p_1, p_2, p_3, p_5.
5th iteration	1. Found permission p_1 with minimum number of uncovered incident edges. 2. Users u_1, u_4 having permission p_1 are selected. 3. Permissions p_2, p_3 assigned to u_1, u_4 are selected. 4. A role is defined with users u_1, u_4 and permissions p_1, p_2, p_3.

TABLE V
ITERATIONS OF ALGORITHM 3

Algorithm 3	
1st iteration	1. Found permission p_1 with minimum number of uncovered edges. 2. Users u_1, u_2, u_3, u_4 are found. 3. Possible combinations of users are (u_1, u_2), (u_1, u_3), (u_1, u_4), (u_2, u_3), (u_2, u_4), (u_3, u_4) as constraint defined is 2. 4. Combination (u_3, u_4) is selected as it covers maximum permissions. 5. Permissions p_2, p_3, p_5 are selected. 6. A role is defined with users u_3, u_4 and permissions p_1, p_2, p_3, p_5.
2nd iteration	1. Found permission p_1 with minimum number of uncovered edges. 2. Users u_1, u_2 are selected. 3. Permissions p_2, p_3 are selected. 4. A role is defined with users u_1, u_2 and permissions p_1, p_2, p_3.
3rd iteration	1. Found permission p_5 with minimum number of uncovered edges. 2. Users u_5, u_6 are selected. 3. Permission p_4 is selected. 4. A role is defined with users u_5, u_6 and permissions p_4, p_5.
4th iteration	1. Found permission p_4 with minimum number of uncovered edges. 2. Users u_2, u_4 are selected. 3. A role is defined with users u_2, u_4 and permission p_4.

V. Experimental Results

We ran our algorithms on different real-world benchmark datasets. Table VI gives the sizes of these datasets [1]. In this table, *Users* represent total number of users. *Permissions* represent total number of permissions available. *Edges* represent total number of permissions assigned to all users, i.e., the size of the UPA matrix.

TABLE VI
REAL-WORLD DATASETS

Dataset	Users	Permissions	Edges
Healthcare	46	46	1,486
APJ	2,044	1,164	6,841
EMEA	35	3,046	7,220
Domino	79	231	730
Firewall 1	365	709	31,951
Firewall 2	325	590	36,428
Americas	3,477	1,587	1,05,205

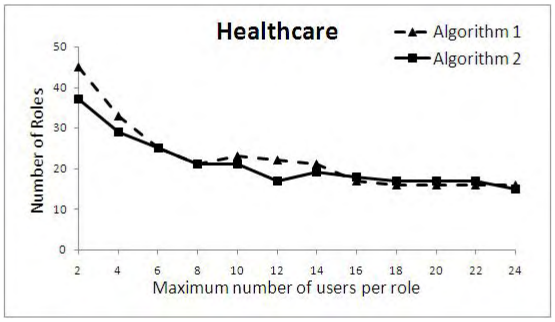

Fig. 2. Number of Roles generated for Healthcare Dataset

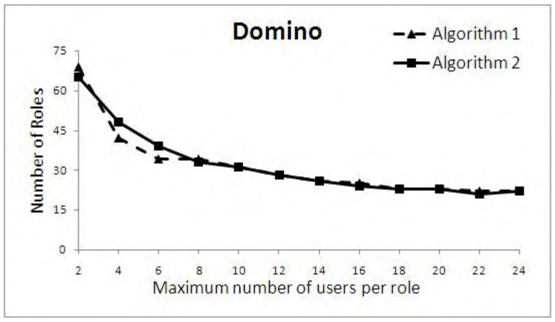

Fig. 3. Number of Roles generated for Domino Dataset

We present the variation of number of roles (y-axis) with varying constraint value (x-axis) for Algorithm 1 as well as Algorithm 2. For each constraint value, all the roles identified by both algorithms satisfy the given constraint. The results show that in most cases, the number of roles generated by Algorithm 2 is less than the number of roles generated by Algorithm 1. It is also observed that in most cases as we increase the number of users belonging to the role, corresponding number of roles decreases for both the algorithms.

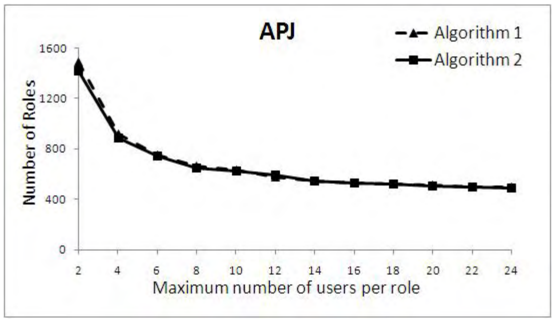

Fig. 4. Number of Roles generated for APJ Dataset

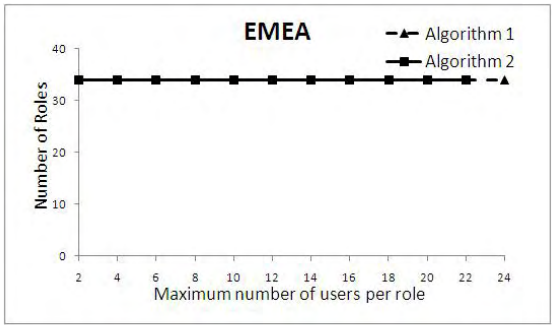

Fig. 5. Number of Roles generated for EMEA Dataset

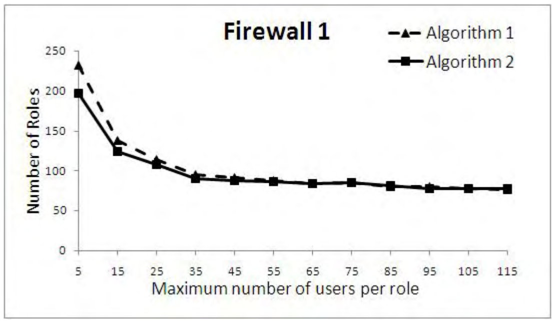

Fig. 6. Number of Roles generated for Firewall 1 Dataset

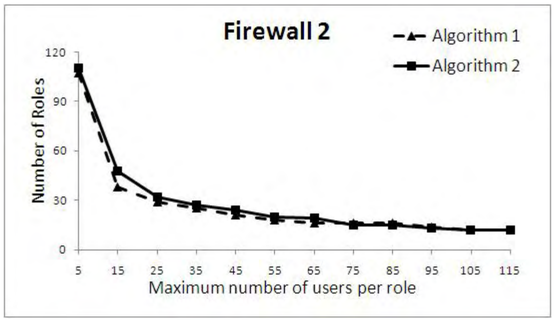

Fig. 7. Number of Roles generated for Firewall 2 Dataset

VI. Conclusion

We have presented three algorithms that consider a constraint for limiting the number of users assigned to any role in the RBAC role mining problem. Variation in the number of mined roles by varying the maximum number of users that can be assigned to any role is studied for the first two algorithms. Both of these algorithms find a set of roles for all the datasets given in Table VI. Also, all of these roles satisfy the constraint. Algorithm 3 being computationally expensive, works only for small datasets. It also gives the least number of roles as demonstrated for our sample dataset in Section IV-D. When the input size is large, time required for Algorithm 3 is quite high. So it is not useful for large organizations. Our future plan is to modify Algorithm 3 in such a way that one gets the least number of roles within a reasonable amount of time.

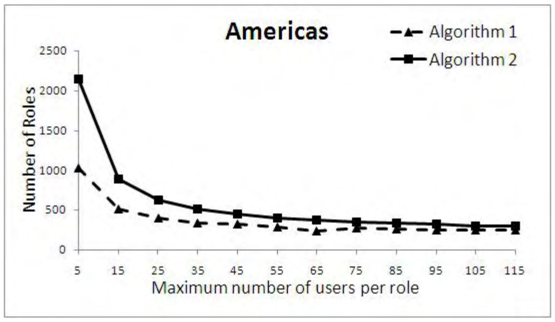

Fig. 8. Number of Roles generated for Americas Dataset

Acknowledgement

This work is partially supported by a grant from the Department of Science and Technology, Govt. of India, under Grant No. SR/S3/EECE/ 082/2007.

References

[1] Alina Ene, William Horne, Nikola Milosavljevic, Prasad Rao, Robert Schreiber, and Robert E. Tarjan. “Fast Exact and Heuristic Methods for Role Minimization Problems”. In SACMAT ’08: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, Estes Park, CO, USA, pages 1 - 10, 2008.

[2] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. “Proposed NIST Standard for Role-Based Access Control”. ACM Transactions on Information and System Security, Volume 4, No. 3, pages 224 - 274, 2001.

[3] Jürgen Schlegelmilch and Ulrike Steffens. “Role Mining with ORCA”. In SACMAT ’05: Proceedings of the tenth ACM Symposium on Access Control Models and Technologies, Sweden, pages 168 - 176, 2005.

[4] Edward J. Coyne. “Role Engineering”. In Proceedings of the 1st ACM Workshop on Role Based Access Control, New York, USA, pages 15 - 16, 1996.

[5] Jaideep Vaidya and Vijayalakshmi Atluri. “Roleminer: Mining Roles using Subset Enumeration”. In CCS 06: Proceedings of the 13th ACM Conference on Computer and Communications Security, USA, pages 144 - 153, 2006.

[6] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. “The Role Mining Problem: Finding a Minimal Descriptive Set of Roles”. In Symposium on Access Control Models and Technologies (SACMAT), France, pages 175 - 184, 2007.

[7] Dana Zhang, Kotagiri Ramamohanarao, and Tim Ebringer. “Role Engineering using Graph Optimisation”. In SACMAT ’07: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, France, pages 139 - 144, 2007.

[8] Ravi Kumar, Shamik Sural, and Arobinda Gupta. “Mining RBAC Roles under Cardinality Constraint”. In ICISS ’10: Proceedings of the 6th International Conference on Information Systems Security, India, Lecture Notes in Computer Science, Springer Verlag, 2010.