

# Graph Based Strategies to Role Engineering

Dana Zhang  
The University of Melbourne  
Melbourne, Australia  
zhangd@csse.unimelb.edu.au

Kotagiri  
Ramamohanarao  
The University of Melbourne  
Melbourne, Australia  
rao@csse.unimelb.edu.au

Steven Versteeg  
CA Inc  
Melbourne, Australia  
steven.versteeg@ca.com

Rui Zhang  
The University of Melbourne  
Melbourne, Australia  
rui@csse.unimelb.edu.au

## ABSTRACT

Role Engineering for Role Based Access Control (RBAC) has emerged as a challenging area of research, both in industry and academia. The problem originates from the practical need to create a set of roles that accurately reflects the internal functionalities of an enterprise. Existing approaches that have used data mining techniques for this problem often generate too many candidate roles and do not consider the effect of a given combination of roles on the overall configuration.

Identification of an ideal RBAC solution is only possible with a clear and concise evaluation of the RBAC configuration goals. To address this issue, we discuss use of the graph model for the Role Engineering problem and show how effective this approach is in the search for a role engineering solution. We evaluate and formalise the problem of identifying the minimum number of descriptive roles for RBAC using a graph model and propose how its variations can be represented. Finally, we introduce novel strategies using the proposed models for future innovation and perform experimentation on both real and synthetically generated data.

## Categories and Subject Descriptors

K.6.5 [Management of Computing and Information Systems]: Security and protection; D.4.6 [Operating Systems]: Security and Protection—*Access Control*

## General Terms

Security, Graph Modeling

## Keywords

Role-based access control, role engineering, role mining, graph modeling

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CSIIRW '10 April 21-23, Oak Ridge, Tennessee, USA  
Copyright 2010 ACM 978-1-4503-0017-9 ...\$5.00.

## 1. INTRODUCTION

Role Based Access Control (RBAC) is an established security mechanism for management of user permission assignments; instead of assigning permissions to users individually, permissions are assigned as a collection through roles [4, 5, 6]. The use of roles reduces the administration required on both users and permissions and allows for improved adherence to enterprise policies.

While the benefits of RBAC are well understood, implementation and maintenance requires Role Engineering, the process of identifying a set of roles that are appropriate, complete and efficient [2]. Without these roles, the full benefits of RBAC cannot be realised.

However, automated role engineering is a challenging task. The problem of finding the minimum number of roles to represent the system has been shown to be NP-complete and existing data mining heuristics often generate too many candidate roles [1, 7, 8, 9]. While weighted pruning can reduce the size of the candidate set, the roles are often weighted individually and there is no guarantee that the chosen set of roles in combination will create the most administratively beneficial RBAC configuration.

A clear and concise cost evaluation of the limiting factors of the overall RBAC configuration goals is required; identification of an ideal solution is only possible if the metrics on which to optimise are identified. To address this issue, graph based role engineering can be used to model both roles and their relationships in relation to the overall RBAC configuration. Doing so allows for evaluation of individual roles, their effect on other roles and their effect on the global configuration.

Recent heuristic approaches have initiated research in the area of graph based role engineering. Zhang et al. maps users, roles and permissions to nodes and their relationships to edges [10]. A cost model and an iterative heuristic to minimise roles and edges is proposed. Ene et al. maps users and permissions to nodes and their relationships to edges in a bi-partite graph and uses biclique cover of edges to identify roles [3]. Mapping of the role engineering problem allows for a more comprehensive evaluation of the roles and their relationship with other roles, users and permissions. In this research, we formalise the graph model and show how it can be used to effectively evaluate an RBAC configuration with respect to the administration cost of roles and their

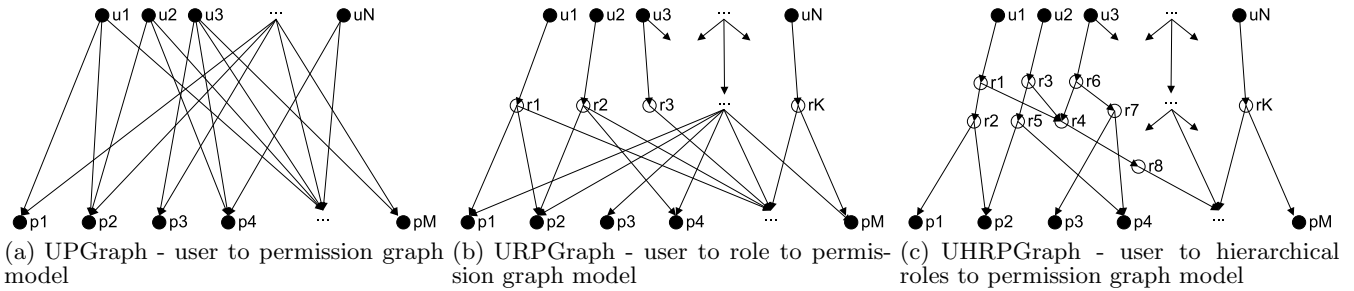


Figure 1: Role engineering graph models

assignments. We present a cost model that can be used to address the problem of role minimisation, edge minimisation and both edge and role minimisation.

The rest of the paper is organised as follows. We introduce a graph model in section 2 with a cost model that is capable of evaluating both roles and their relationships in the RBAC configuration. In Section 3, we show how our graph strategy can represent different role engineering problems and their constraints. We also propose techniques for solving these problems and perform preliminary experimentation in Section 4. Finally, Section 5 summarises the impact of the graph model strategy for the role engineering problem and gives directions for future work.

## 2. ROLE BASED ACCESS CONTROL GRAPH MODELS

We use the following concepts proposed by the National Institute of Standards and Technology (NIST) for RBAC [5]:

- *USERS, ROLES and PRMS*, the set of users  $U$ , roles  $R$  and permissions  $P$  respectively where permissions represent allowable operations on objects within the system.
- $UA \subseteq USERS \times ROLES$ , a many to many mapping of user-to-role assignments.
- $PA \subseteq PRMS \times ROLES$ , a many to many mapping of permission-to-role assignments.
- $RH \subseteq ROLES \times ROLES$ , a partial order on *ROLES* called the inheritance relation, written as  $\succeq$ , where  $r_1 \succeq r_2$  only if all permissions of  $r_2$  are also permissions of  $r_1$ .

We will also discuss  $UP \subseteq USERS \times PRMS$ , the underlying many to many mapping of user-to-permission assignments. In this paper, we do not consider sessions or separation of duty constraints. However, they can be incorporated into our framework.

These RBAC concepts can be modeled using the directed acyclic graph (DAG), represented as  $G = (V, E)$  where  $V = V(G)$  is the set of vertices and  $E = E(G)$  is the set of edges.  $E$  is a set of ordered pairs; when mapped to RBAC components the direction of the edges are user-to-permission  $u \rightarrow p$ , user-to-role  $u \rightarrow r$ , role-to-permission  $r \rightarrow p$  and role-to-role  $r_1 \rightarrow r_2$  when  $r_1 \succeq r_2$ .

We formalise the role engineering graph models as UPGraph, URPGraph and UHRPGraph, shown in Figure 1. In

the absence of RBAC, all fundamental access control can be represented by the UPGraph (Figure 1(a)). Users and permissions are mapped to nodes and direct user-to-permission assignments are mapped to edges. Figure 1(b) shows the URPGraph to add an additional layer of abstraction; roles are added as nodes into the graph and edges now represent user-to-role assignments and role-to-permission assignments. Finally, the UHRPGraph in Figure 1(c) allows for partial orderings on roles to be represented. While nodes still represent users, roles and permissions, edges now represent role-to-role assignments as well as user-to-role and role-to-permission assignments.

The benefit of the RBAC graph models is that each RBAC component that requires administration can be represented and evaluated using the following cost model.

$$\text{cost}(G) = c_1|V_R| + c_2|E| \quad (1)$$

where

$c_1$  is the administration or maintenance cost perceived by the enterprise in association with a role

$|V_R|$  is the number of role nodes in graph  $G$

$c_2$  is the cost of a user-to-role, role-to-role (if  $G$  represents a UHRPGraph) and role-to-permission assignment perceived by the enterprise

$|E|$  is the number of user-to-role, role-to-role (if  $G$  represents a UHRPGraph) and role-to-permission assignments in  $G$

This cost model evaluates roles and their relationships in the RBAC graph model. While permission and user nodes are part of the graph model, they are not modified during the role engineering process, as a result they do not need to be evaluated with this cost model.

Another benefit of the RBAC graph models is that the relationship between each of the RBAC components are effectively captured and can be maintained easily. For example, the assignment of  $r_1$  to  $u_1$  where  $r_1 \succ r_2$  makes the assignment of  $r_2$  to  $u_1$  unnecessary. In the graph models, a direct edge from  $r_2$  to  $u_1$  would be removed if  $r_1$  is assigned to  $u_1$ . As a result, the effect of the addition or removal of a role can be easily reflected by the models. This is currently not considered in existing data mining approaches where identification of candidate roles is a separate process to the assignment of users to roles.

### 3. ROLE ENGINEERING GRAPH PROBLEMS

In mapping the role engineering problems to graph representation, reducing the complexity of the graph is equivalent to reducing the complexity of the RBAC configuration. The RBAC graph models can be reduced using three constraints: role minimisation, edge minimisation and both role and edge minimisation [8, 10]. In the basic problem, role minimisation attempts to identify the minimum set of descriptive roles given user-to-permission assignments, that is the minimum number of roles that can represent all user permission assignments. In edge minimisation, role engineering attempts to identify the roles that produce the minimum number edges. This is effective as the edges represent an assignment that requires administration. This assignment could be a user-to-role, role-to-role or role-to-permission assignment. Finally, role and edge minimisation attempts to minimise both roles and edges.

Minimisation of both roles and edges is an interesting problem. Reducing the number of roles does not always reduce the number edges. For example, a role that is assigned to three users and contains two permissions. This role has five edges, three for each of the users and two for each of the permissions. Removing this role would remove five edges but six edges would need to be added as each of the three users still need to be assigned the two required permissions as process of role engineering should not remove existing permissions or add spurious assignments.

These constraints can all be measured using the cost model in Equation 1.  $\text{cost}(G) = c_1|V_R| + c_2|E|$  can be minimised to solve the role and edge minimisation role engineering problem. For the role minimisation, the following cost metric should be minimized.

$$\text{cost}(G) = c_1|V_R| \quad (2)$$

This is produced by setting  $c_2 = 0$  in Equation 1.

For edge minimisation, the following cost metric should be minimized.

$$\text{cost}(G) = c_2|E| \quad (3)$$

This is produced by setting  $c_1 = 0$  in Equation 1.

All of these problems have been shown to be NP-Complete and hard to approximate [3, 8]. As a result, we propose a heuristic technique to address all three variations of the problem in Algorithm 1.

In this algorithm, the initial graph  $G$  is created based on the underlying user permission assignments of the enterprise (Line 2).  $G$  starts as an UPGraph and turns into a URP-Graph when roles are inserted based on each user's permission assignments with edges updated to reflect the insertion of the new roles (Line 3). If multiple users have the same set of permissions, only one role is added to  $G$  and the users are all assigned to the same role.

To greedily reduce the cost of the graph, roles are then inserted or deleted in Lines 4–12. If a role is to be deleted, a role that is costly is removed. If a role is added, the role that would reduce graph cost is added. If these operations create roles with a partial ordering,  $G$  becomes a UHRPGraph.

Since this heuristic algorithm is cost model independent, identification of the most effective role to delete or create can be based on Equation 1, Equation 2 or Equation 3 in Line 7 and Line 10.

---

#### Algorithm 1: Heuristic Strategy for Graph Based Role Engineering

---

**Require:**  
UP - user permission assignments  
**Result:**  
 $G$  - optimised RBAC configuration

```

1 begin
2   create  $G$  - add each user and permission to a node
   in  $G$ , add each permission assignment as an edge in
    $G$ 
3   insert initial roles into  $G$  - create a role based on
   each user's permission set and insert into  $G$ , update
   edges to reflect assignment of new roles
4   repeat
5     choose operation - delete or create a role
6     if  $operation == delete$  then
7       identify costly role in  $G$ 
8       remove role from  $G$  and update edges
9     if  $operation == create$  then
10      identify beneficial role not in  $G$ 
11      add role to  $G$  and update edges
12    until no more cost reducing operations ;
13 end

```

---

Finally, Line 8 and Line 11 updates the edges of  $G$  after the delete or create operation so no additional permission assignments are granted and no existing permission assignments are removed. Relationships between roles that are inherited that no longer need to be directly assigned are also updated.

A similar Iterative Graph Optimisation method was proposed by Zhang et al. [10]. In their method, pairwise roles are compared and operations are performed based on three scenarios: 1) if the two roles are the same, they are merged into the same role 2) if one role is a superset of the other role, a partial ordering relationship is placed between the roles 3) if there is an intersect between the two roles, a new role is created based on the intersect.

In our heuristic strategy, the create and remove operations are performed randomly instead of in ordered pairwise matchings. The performed operations are also different. Instead of merge, intersect and partial ordering operations, Algorithm 1 has only two operations: create and delete. Merge and intersect operations are not required as only unique roles are added to  $G$  and edge relationships are updated during each operation. The delete operation is included to remove non-beneficial roles, allowing the create role operation to be reversible. In Iterative Graph Optimisation, a role cannot be removed after it is created, even if the later iterations of the optimisation identify the role to be undesirable.

### 4. EXPERIMENTAL RESULTS

To demonstrate the effectiveness of both the proposed model and heuristic, we perform testing on synthetically generated user permission assignments using Zhang et al.'s data generator for role engineering testing [11]. The data generator randomly assigns permissions to roles and roles to users using a Gaussian distribution given the number of users, roles and permissions, and the average and standard deviation for number of permissions per role and roles per user. As an example, we show results for a data set with 500 users, 50 roles and 500 permissions; the average and standard deviation of roles to users was 5 and 2 respectively,

the average and standard deviation of permissions to roles was 10 and 5 respectively. The roles were removed and the heuristic algorithm proposed in Algorithm 1 was run using only user-to-permission data to recover the original roles.

Algorithm 1 was also run on real access control data from the Department of Computer Science and Software Engineering at The University of Melbourne. These permissions were taken from a unix environment where each group is considered an assignable permission. RBAC does not exist in this environment. The dataset contains 598 users, 306 permissions and 1630 permission assignments.

Two different cost models were used: Equation 1 and Equation 3 with  $c_1 = 1$  and  $c_2 = 1$ . All tests were run using a single core on a 2.38GHz Dell Xeon E5440 Server. Each test was repeated 10 times and the average results are shown in Table 1 and Table 2. While it is possible to perform the heuristic strategy using Equation 2, additional measures to ensure permission can only be assigned through roles must be enforced. We are currently ensuring these measures are enforced into our graph operations.

**Table 1: Results of the Heuristic Strategy for Graph Based Role Engineering using different cost models on synthetically generated data with 500 users, 50 roles and 500 permissions**

Cost metric	$c_1 V_R  + c_2 E $	$c_2 E $
Average number of roles recovered	44.4	44.7
Graph cost without roles	26225.0	26225.0
Average graph cost of result with roles	4199.6	3914.1
Average processing time in seconds	94.9	98.1

**Table 2: Results of the Heuristic Strategy for Graph Based Role Engineering using different cost models on data from the Department of Computer Science at the University of Melbourne**

Cost metric	$c_1 V_R  + c_2 E $	$c_2 E $
Average number of roles identified	46.8	75.8
Graph cost without roles	1630.0	1630.0
Average graph cost of result with roles	1442.1	1383.4
Average processing time in seconds	11.4	12.8

From Table 1 and Table 2, it can be seen that the Heuristic Strategy for Graph Based Role Engineering can identify solutions with significantly reduced cost evaluations quickly. On average, just less than 45 out of 50 roles can be recovered exactly in less than 100 seconds in the synthetically generated data. In the real data set instance, reducing both the number of roles and edges produced less roles in comparison with reducing only edges.

## 5. CONCLUSION AND FUTURE WORK

Role engineering is a fundamental process for migration to and maintenance of RBAC. Creation of a set of appropriate, complete and efficient roles requires a clear and concise evaluation of the RBAC configuration. In this paper, we propose strategies for modeling and evaluating RBAC components and relationships according to three minimisation constraints.

We discuss and introduce graph and cost models for identifying the minimum number of descriptive roles, a set of roles that produces the minimum number of administration requirements and a combination of both. Using graphs allows for representation of the full RBAC configuration, their components and their relationships with each other. We also propose a heuristic strategy for graph based role engineering that is cost model independent and perform experimental analysis on both real and synthetically generated data.

Future work in this research involves further experimentation on synthetic and real data to verify the proposed heuristic algorithm and analysis of results given the three different cost models. Alternative optimisation strategies based on the proposed graph models are also being investigated. Finally, sessions and separation of duty constraints are also being analysed for incorporation with graph strategies for future innovation.

## 6. REFERENCES

- [1] A. Colantonio, R. D. Pietro, and A. Ocello. Leveraging lattices to improve role mining. In *Proceedings of The Ifip Tc 11 23rd International Information Security Conference (SEC'08)*, pages 333–347, Boston, 2008. Springer.
- [2] E. J. Coyne. Role engineering. In *RBAC '95: Proceedings of the first ACM Workshop on Role-based access control*, pages 4–5, New York, NY, USA, 1995. ACM Press.
- [3] A. Ene, W. Horne, M. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan. Fast exact and heuristic methods for role minimization problems. In *SACMAT'08: Proceedings of the thirteenth ACM symposium on Access control models and technologies*, Estes Park, Colorado, June 2008.
- [4] D. F. Ferraiolo and D. R. Kuhn. Role-Based Access Control. In *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, pages 554–563, Baltimore, Maryland, USA, 1992.
- [5] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.
- [6] A. N. S. I. Inc. Role Based Access Control. ANSI INCITS 359-2004, 2004.
- [7] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining roles with semantic meanings. In *SACMAT'08: Proceedings of the thirteenth ACM symposium on Access control models and technologies*, Estes Park, Colorado, June 2008.
- [8] J. Vaidya. The role mining problem: Finding a minimal descriptive set of roles. In *SACMAT '07: Proceedings of the twelfth ACM symposium on Access control models and technologies*, pages 175–184, New York, NY, USA, 2007. ACM Press.
- [9] J. Vaidya, V. Atluri, and J. Warner. Roleminer: Mining roles using subset enumeration. In *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2006. ACM Press.
- [10] D. Zhang, K. Ramamohanarao, and T. Ebringer. Role engineering using graph optimisation. In *SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 139–144, New York, NY, USA, 2007. ACM.
- [11] D. Zhang, K. Ramamohanarao, and R. Zhang. Synthetic data generation for study of role engineering. <http://www.cs.mu.oz.au/~zhangd/roledata>, 2008.