Role engineering using graph optimisation

Conference Paper · January 2007

DOI: 10.1145/1266840.1266862 · Source: DBLP

CITATIONS
112

READS
579

3 authors, including:

Dana Zhang

6 PUBLICATIONS **185** CITATIONS

SEE PROFILE

Kotagiri Ramamohanarao

University of Melbourne

544 PUBLICATIONS **12,047** CITATIONS

SEE PROFILE

Role Engineering using Graph Optimisation*

Dana Zhang
Department of Computer Science and Software Engineering
The University of Melbourne
zhangd@csse.unimelb.edu.au

Kotagiri Ramamohanarao
Department of Computer Science and Software Engineering
The University of Melbourne
rao@csse.unimelb.edu.au

Tim Ebringer
CA Labs
Richmond, Australia
tim.ebringer@ca.com

ABSTRACT

Role engineering is one of the fundamental phases for migrating existing enterprises to Role Based Access Control. In organisations with a large number of users and permissions, this task can be time consuming and costly if a top down approach is used. Existing bottom up approaches are not sufficient in producing a comprehensive set of roles for hierarchical Role Based Access Control. In this research, we propose a predominately bottom up approach that uses Graph Optimisation to identify appropriate role hierarchies. Additional partial role specifications can be incorporated to produce a hybrid approach. Using rules that reduce administration requirements, roles and their hierarchies are automatically extracted from large numbers of permission assignments. The results of the Graph Optimisation approach are hierarchical Role Based Access Control infrastructures that offer improved access control administration for the system.

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, Algorithms

Keywords

Role-based access control, role engineering, role hierarchy, role definition, graph optimisation

1. INTRODUCTION

The incorporation of roles for Role Based Access Control(RBAC) [3] into large enterprises reduces administration costs of user to object permissions. Additional placement of these roles into a hierarchy reduces management costs of the roles. This reduction of administration requirements is one of the main benefits for the RBAC infrastructure. While these benefits are significant, implementation of roles into existing environments requires the generation of a set of roles that accurately reflect the functionalities of that environment.

This creation of a set of roles is called role engineering [1]. Role engineering involves the extraction of roles from an existing infrastructure and is the first phase of the role lifecycle [4]. This phase is difficult as identifying roles from natural organisational functionalities is not intuitive, requiring many human hours of analysis and elicitation if a top down approach is deployed [5].

An alternative approach is bottom up role engineering, using existing user permission assignments to derive candidate roles. This approach attempts to automate the process of role discovery to create a more cost effective solution to role engineering. Existing approaches have used role mining [5], the application of data mining techniques on permission to user assignments to extract roles and hierarchies. However, role mining suffers from the inherent problems of their chosen mining technique, causing conceptual errors in the way roles are automatically extracted.

To bypass the data mining issues, this paper proposes an automated bottom up role engineering approach that uses an alternative method for creating a role structure. The proposed approach uses Graph Optimisation to derive the most suitable roles for optimal administration benefits. This approach does not suffer the same draw backs present in data mining methods and uses graphs and matrices to represent users and their permissions. Role to user assignments and permission to role assignments are represented as decompositions of the user permission access control matrix. The set of roles given a set of users and permissions is derivable by finding the decomposition of the user permission matrix. For our approach, the measures used to create the decomposition relies on the reduced permission management benefits that RBAC offers. The resultant infrastructure provides improved administration benefits for permission management.

Partial role information can also be added to produce a hybrid approach. If information exists on certain roles that are required in the final role hierarchy, they can be added before the engineering process.

This research aims to discover a stable role hierarchy using Graph Optimisation through a set of users and their permissions. The approach attempts to address previous issues with role engineering as discussed in Section 2. This section will give an overview of previous approaches as well as some

*This research is supported and funded by CA Labs

SACMAT'07, June 20-22, 2007, Sophia Antipolis, France.

background and motivation for our research. Section 3 will explain our methodology and how Graph Optimisation can be used to find a stable role hierarchy for RBAC. We test and discuss this approach in Section 4 and finally conclude our findings with future work possibilities in Section 5.

2. BACKGROUND

This section will discuss existing approaches for role engineering and their limitations. Our motivation was driven by the downfalls of current techniques for role engineering. Our aim is to identify a set of roles that creates the most administration benefits for RBAC.

Role engineering originated from the need to create a set of roles that encapsulates the functionalities and responsibilities of a working enterprise [1]. The intuitive method of identifying roles is by elicitation; extraction from job functionalities, scenarios and processes [6, 7]. Initial approaches used use cases to group actions performed in a particular procedure into a role [2]. Different variations on role requirement elicitations were analysed but there is one major problem that is unavoidable in all top down approaches: top down role engineering is a manual process.

Manual role engineering is a time consuming and costly phase of RBAC development. Results derived using top down approaches are also prone to errors as operations performed together in a particular transaction may not necessarily imply a role. Implementation of semi-automated approaches can potentially reduce costs by 60% [5]. Top down approaches also disregard the information that already exist within the enterprise.

To solve the issue of manual requirement elicitation for role extraction, more recent approaches have turned to automated bottom up processes for role engineering. Due to the amount of data that needs to be processed and analysed, the most common approach to bottom up role engineering is *role mining*, the application of data mining techniques to extract roles. Role mining also utilizes user permission assignments that are present within the established organisation. Current approaches have used clustering analysis to group sets of permissions together to form roles [5, 9, 10]. In these approaches, each identified cluster is a candidate role.

Hierarchical clustering, clustering on users and clustering on permissions have been explored. These semi and fully automated approaches, while requiring significantly less user input than top down approaches, suffer from the drawbacks of their chosen clustering algorithm. Hierarchical clustering creates roles in a possible hierarchy, however, permissions may only exist in one role inheritance path. This limitation is not a realistic assumption in practice. Other clustering drawbacks that have been faced include requiring the number of role-clusters to be predefined before role mining can proceed and determining of weightings for best candidate role-clusters after they have been identified. Roles identified in the latter approaches also have no concept of a hierarchy.

For our work, we wanted to create a bottom up approach that uses existing user permissions to offer insight into the internal functionalities that can otherwise be neglected by top up approaches. We wanted to create an automated process without the problems faced by clustering role engineering techniques. To bypass the caveats of existing role mining techinques, Graph Optimisation of role to user and permission to role relationships were used to derive the best role hierarchy for efficient management and implementation.

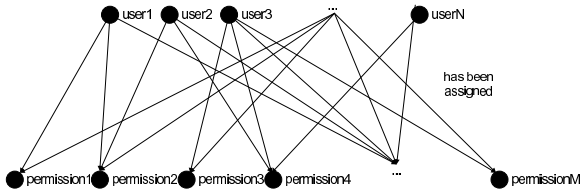


Figure 1: Graph representation of permission to user assignments

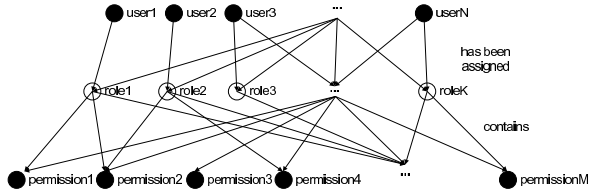


Figure 2: Graph representation of role to users and role to permission assignments

To allow for flexibility, the approach can be modified to become a hybrid approach if preferred roles are specified. With or without partial role specification, the approach is capable of identifying roles for efficient administration management.

3. ROLE HIERARCHY GRAPHING

3.1 Graph and Matrix Representation

The fundamental concepts in access control security are users u, permissions p and permission to user assignments $u \rightarrow p$. Permissions can be assigned to multiple users, users can be assigned multiple permissions. Given a set of N users and M permissions and a set of user permission assignments $\{u \rightarrow p\}$, a graph representation of their basic relationships can be shown in Figure 1.

This relationship between users and permissions can also be represented as an access control matrix A, a sparse matrix that shows permission assignments of every user [8]. While rarely being implemented due to memory constraints, all fundamental security access can be represented as $A =$ access control matrix $= [a_{ij}]$.

In this matrix, $a_{ij} = 1$ implies user i has permission j and $a_{ij} = 0$ implies user i does not have permission j. Each row represents all the permissions assigned to a particular user and each column represents the users a particular permission has been assigned to.

Both the graph and matrix represent direct permission to user assignments in the system. In RBAC, the intermediary concept of a role r is added. Conceptually, this incorporation of roles inserts an addition layer of abstraction between permissions and users. Roles are assigned users $u \rightarrow r$ and permissions are then assigned roles $r \rightarrow p$ instead of direct permission to user assignments $u \rightarrow p$. Figure 2 shows the relationship between users, roles and permissions.

This addition of roles into the graph can also be shown as a decomposition of A into two matrices B and C where $B =$ user-role matrix $= [b_{ij}]$ and $C =$ role-access matrix $= [c_{ij}]$. In these decomposition matrices, B represents role to user assignments $\{u \rightarrow r\}$ and C to represent permission to

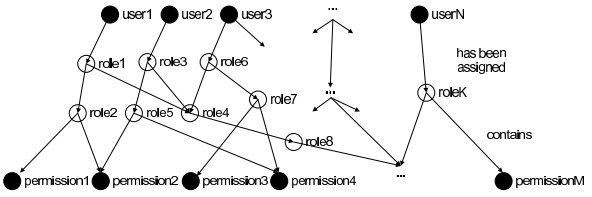


Figure 3: Graph representation of roles to users in a hierarchy and role to permission assignments

role assignments $\{r \rightarrow p\}$. That is, $B = [b_{ij}]$ where $b_{ij} = 1$ implies user i has role j and $b_{ij} = 0$ implies user i does not have has role j. Matrix $C = [c_{ij}]$ where $c_{ij} = 1$ implies role i been given permission j and $c_{ij} = 0$ implies role i has not been granted permission j. The operation on values of B and C to form the values of A can be described using operators $\wedge$ and $\bigvee$.

$$a_{ij} = \bigvee_k b_{ik} \wedge c_{kj} \tag{1}$$

In this equation, $\wedge$ represents the *and* operator, meaning $b_{ik} \wedge c_{kj} = 1$ if and only if both $b_{ik} = 1$ and $c_{kj} = 1$ where i represents a user, k represents a role and j represents the permission. However, since user i can access permission j through any role k, the *or* operator $\bigvee$ is used for all relevant $k \wedge$ operations.

A more generic form to creating an operation for matrix elements is to create the equality for matrices A, B and C. Using $\circledast$ as the composition operator, we now have equation:

$$A = B \circledast C \tag{2}$$

Equations 1 and 2 are logically equivalent, operating on the elements of the matrix and the matrices as a whole respectively. A concrete example of the $\circledast$ operator is the natural join operator $\bowtie$ in relational algebra. If B is considered the relation between users and roles and C considered the relation between roles and permissions, a natural join on B and C, $B \bowtie C$ would produce A, a relation containing users and their assigned permissions.

This decomposition lets us represent a single layer of roles between users and permissions. For hierarchical RBAC, a partial ordering on roles needs to be represented (Figure 3). This partial ordering of roles r such that $r \rightarrow r'$ implies permissions that exist within r' also exist in r.

$$r \rightarrow r' : \forall p \in r', p \in r \tag{3}$$

One solution is to add an additional matrix decomposition to represent relationships between roles. The decomposition of A with a partial ordering hierarchy could be represented as $A = B \circledast B' \circledast C$ where B' represents role to role assignments. That is, $B' = [b'_{ij}]$ where $b'_{ij} = 1$ implies role $i \rightarrow j$ or $\forall p \in j, p \in i$. Matrix B allows any user to be assigned any role and matrix B' allows the roles within B to interact with each other. This means while B' enforces a partial ordering on roles to form layers, a user can be directly assigned to any role at any level of the ordering.

To avoid the addition of an extra matrix, C can be a self referencing matrix, enabling roles to reference other roles to form the partial ordering. This alteration in C now allows any role to be assigned any other roles as well as permissions. Matrix B can now associate any user to a role at any level of the partial ordering.

Allowing C to be self referencing matrix or using B' in the decomposition would produce Figure 3, creating more than one conceptual layer of roles to lie between users and permissions. This is the role infrastructure for hierarchical RBAC.

3.2 Problem Description

To create an RBAC infrastructure, we add roles and their corresponding relations with other roles, users and permissions. We describe the problem using our representations of A, B and C and work towards finding the best B and C decompositions of A. Discovering the best decompositions would identify the most effective role infrastructure from the given user permission assignments.

PROBLEM 1. *Given A, find decompositions B and C such that the identified roles provide the best* RBAC *solution for the system.*

This problem statement requires a definition for what the best solution means. Given A, there are numerous compositions that comply with Equation 2. In order to find the optimal solution, we introduce some constraints on the decomposition that correspond with observations of RBAC. For our research, the constraints chosen were optimality measures that can be used to create a role infrastructure that offers the most improved administration benefits. One of the major benefits of RBAC is the reduction in management costs of access control permissions and the derived infrastructure attempts to maximise this reduction.

One observation that can be made is the addition of roles between users and permissions should not create additional or remove existing user permissions. The addition of roles should have no effect on final permission to user assignments, only the way permissions are assigned and thus what needs to be managed by security administrators.

The addition of roles means management now deals with role to user and permission to role assignments instead permissions to user assignments. The best role solution for increased management efficiency is one that requires the addition of the least number of role to user and permission to user relationships.

OBSERVATION 1. *Reducing the number of role to user and permission to role assignments, reduces the administration required for role related assignments.*

This first observation deals with role related relationships. These role related relationships are represented as edges in our graph. To reduce the number of role to user and permission to role assignments, the number of edges in the role based graph should be reduced. Since the role infrastructure of user permissions A is constructed from B and C, edges represented in B and C must be minimal satisfying Equation 2. Let the notation $|M|$ represent the number of edges represented by a matrix M. That is, the number of elements in the matrix set to 1, $m_{ij} = 1$ where each $m_{ij} = 1$ represents an edge or a relationship between i and j. Using the notation, we have:

PROBLEM 2. *Given A, find $A = B \circledast C$ such that $|B|+|C|$ is minimum.*

These edges or role related relationships with users and permissions are added with roles for the implementation of RBAC. While role relationships should be reduced for improved management of these assignments, reducing the number of roles also reduces the number of objects that administration has to manage. The same principle applies for roles. In the presence of roles, adding the least number of roles while ensuring correct permission to user assignments allows for reduced management of roles.

OBSERVATION 2. *Reducing the number of roles reduces the administration required on roles.*

This second observation deals with the number of roles present. The decomposition of A into B and C should be such that the minimum number of roles is represented. Based on this observation, given A, we need to find decomposition B and C such that the number of roles is minimum satisfying Equation 2. Let the notation $[M]$ represent the number of roles in a role access matrix M. That is, the number of rows in the matrix. Using the notation, we have:

PROBLEM 3. *Given A, find $A = B \circledast C$ such that $[C]$ is minimum.*

The derivation of roles from existing user permission assignments must consider Problems 2 and 3. Doing so provides an approach for finding a role infrastructure that optimises administration and enterprise security management.

However, the relationship between roles and edges from roles is not always linear. The removal of one role may increase the number of edges for other roles and the removal of an edge from a role may produce the need for more roles. We want both the minimal number of edges and the minimal number of roles.

OBSERVATION 3. *Reducing the number of roles and edges reduces administration required in a* RBAC *infrastructure.*

Using the sum of the number of roles and edges as the optimisation measurement, this observation can be applied to our problem statement to find the optimal decomposition for increased administration benefits satisfying Equation 2.

PROBLEM 4. *Given A, find $A = B \circledast C$ such that $|B| + |C| + [C]$ is minimum.*

3.2.1 Problem extension for a hybrid approach

Using the given definitions, it is possible to search for a role infrastructure if a set of roles or partial roles definitions are specified. In some enterprises, there may be some limited understanding on what roles are required. If partial roles can be specified, then these roles should be protected and retained during the role discovery process.

When role specifications are made, the following observation is made in relation to the Graph Optimisation approach to role engineering:

OBSERVATION 4. *Reducing the number of roles and edges while preserving partially defined roles reduces administration and ensures specified correctness of a* RBAC *infrastructure.*

In our notation, partially defined roles means a partially defined C. Given A and certain predefined roles (that is a partially defined C), find B and C such that the number of edges and the number of roles in the graph formed by B and C is minimal satisfying Equation 2

PROBLEM 5. *Given A and a partially defined C, find $A = B \circledast C$ such that $|B| + |C| + [C]$ is minimum.*

Problem 5 is a more general form of Problem 4 and can be solved to address more practical problems when there are notions of how the role infrastructure is to exist. Solving Problem 5 produces a hybrid approach to role engineering, permitting administrators to specify roles that must exist the hierarchy.

3.3 Graph Optimisation Solution

To find the optimal graphing solution for a set of permission assignments, Problem 4 needs to be addressed. In this case, optimal refers to the best implementation for improved and efficient management of access control permissions. The desired solution should offer a reduction in roles and their relationships, ensuring the reduction of managed objects and the increase in efficiency of management.

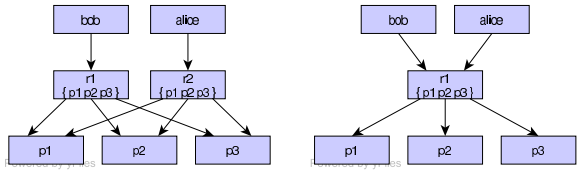
Addressing Problem 4 can produce solutions for Problem 5. The latter problem is a simplification of the initial problem, having certain roles and their relationships defined, or various graph nodes and edges predetermined in the infrastructure. To deal with the first problem, $|B| + |C| + [C]$ must be calculated and optimised for B and C given A.

Algorithm 1 Graph Optimisation

Require: users u, permissions p and permission to user assignments $u \rightarrow p$
1: identify each user permission sets as possible role
2: calculate optimisation metric = number of edges + number of roles
3: {*merge roles until stable*}
4: **while** there can be more merges **do**
5: identify two roles
6: split or merge depending on permission set overlap
7: calculate new optimisation metric = number of edges + number of roles
8: **if** new optimisation metric > optimisation metric **then**
9: undo operation
10: **else**
11: old optimisation metric = new optimisation metric
12: **end if**
13: **end while**

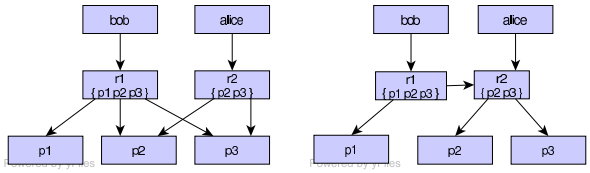
In this research, Algorithm 1 is proposed as a Graph Optimisation solution to the problem. A is reverse engineered to find B and C using the optimality metric, $|B| + |C| + [C]$. Users, permissions and permission assignments from A are required and Step 1 of the algorithm forms starting roles from each user's permission set. This step creates the initial graph that the Graph Optimisation algorithm manipulates. Step 2 of the algorithm calculates the optimisation metric as $|B| + |C| + [C]$.

In steps 4 to 13, the algorithm identifies pairs of roles to analyse and performs merge or split operations if they improve the optimisation metric of the graph. Pairs of analysed roles can be separated into four scenarios depending on the permission set overlap of the two roles: (1) the roles' permission sets are equal (2) one role's permission set is a subset of the other role's permission set (3) there is overlap between the two role's permission sets and one permission set is not the subset of the other permission set and (4) the two roles' permission sets are distinct.



(a) before merge (b) after merge

Figure 4: Scenario 1: merging of two roles that have the same permission sets



(a) before merge (b) after merge

Figure 5: Scenario 2: one role's permission set is a superset of another role's permissions

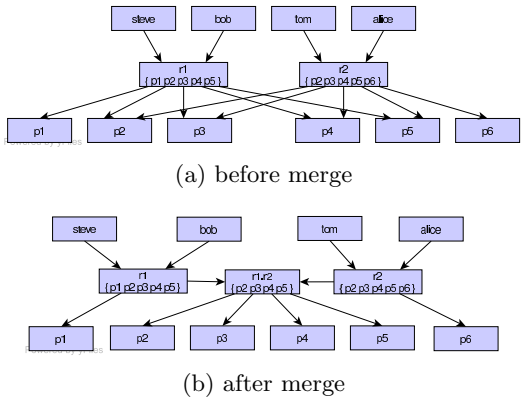
In the first scenario, the two roles that are under investigation have the same permission set. When two roles are the same, they are merged into one role with their relations updated as in Figure 4. Users or roles that have been assigned either of the previous roles are now assigned to this new merged role. Merging two roles into one role will always improve the optimisation metric, reducing the number of roles by one and possibly reducing the number of edges.

The second scenario analyses roles where the permission set of the first role is a subset or superset of the permission set of the second role. When this situation occurs, a link from the superset to the subset is created and outgoing links from the superset role are updated (Figure 5).

This operation in scenario two will decrease or have no effect on the optimisation metric. When the role of the subset contains one link out, the creation of a link between the superset and this subset adds a link between the two roles and removes the direct links from the superset to the target of the link of the subset. This has no effect on the metric. Subsets with links out of the role greater than one are expected to decrease the optimality metric when an additional link between the subset and superset is added and at least one link is removed from the superset.

In scenario three, there is an overlap in the permission sets of the two roles where neither role's permission set is a subset of the other role's permissions (Figure 6). In this scenario, a role containing the overlap of the two roles is created and two links to this new role are created. The increase to the optimisation metric is the addition of the links from the original roles to the newly created role and the reduction to the optimisation metric is related to how many outgoing edges connect to the newly created role. Each edge leaving the newly created role potentially merges edges from the original two roles, causing a decrease in the number of edges. Generally when the created role contains two or more links out, the optimisation metric is unchanged or improved.

Finally, in the fourth scenario, the two roles analysed have distinct permission sets. No merge can be created between



(a) before merge

(b) after merge

Figure 6: Scenario 3: two roles that have a common set of permissions

two such roles. The only operation that can be performed is to create a new role that is a superset of the permissions in the analysed roles and create relationships with the original roles. However, doing so would increase the metric by increasing the number of roles and also increasing the number of edges. No action is preformed when roles with no overlapping permissions are dealt with.

The above description of the algorithm was been created to address Problem 4. To incorporate the possibility of partially defined roles as in Problem 5, additional role sets may be included as part of step 1 in Algorithm 1. These roles can be added as nodes into the graph with the corresponding relationships with other roles or permissions added. These roles will exist in the graph and merged according to the algorithm unless they violate the original partial definitions. Roles and edges can be tagged to be unmodified.

During runtime, the initial iteration checks for all role pair combinations. This operation is $O(n^2)$ where n is the initial number of roles. In each subsequent iteration of the algorithm, only pairs involving the modified roles from the previous iteration need to be analysed. Iterations proceed until the structure is stable and no more pairs can be analysed or no further improvement of the optimisation metric is possible.

4. RESULTS AND DISCUSSION

The algorithm was tested on a small and larger test set of user permission assignments from The University of Melbourne. Test environments did not have RBAC implementations. The permission sets are access control lists of permissions assigned to every user within The Department of Computer Science and Software Engineering. The smaller test set consisted of 20 users, 50 permissions and 161 permission to user assignments.

When the smaller test set were placed into a graph, there were 90 nodes and 181 edges. The 90 nodes consisted of 20 users, 20 corresponding roles and 50 permissions. The edges were 20 relationships between users and roles and 161 permission assignments from roles to permissions. The initial starting metric is $181 + 20 = 201$.

During Graph Optimisation, 33 successful merging operations took place. After the operation, there were 26 roles between the 20 users and 50 permissions, totaling in 96 nodes

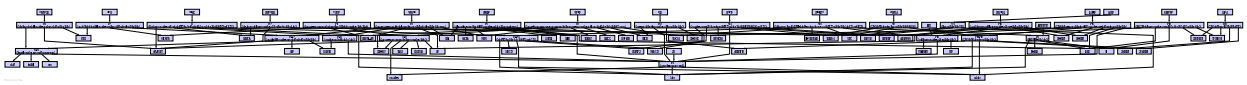


Figure 7: Produced hierarchical layout of small test data. The purpose of this diagram is to show the general architecture and complexities of a produced role based access control infrastructure

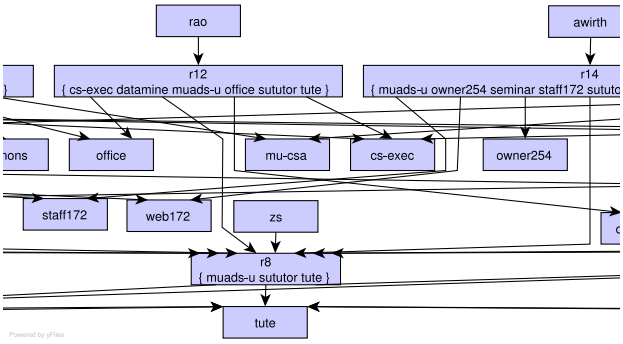


Figure 8: Section of resultant hierarchical layout produced from the Graph Optimisation Algorithm

in the graph. While there was an increase in roles, the number of edges were reduced by 42. The final number of edges in the graph was 139, creating a resultant optimisation metric of $139 + 26 = 165$. Figure 7 shows the final hierarchy created by the Graph Optimisation approach on the small test set. This type of layout is a common RBAC infrastructure for a small set of roles.

A section of the produced hierarchy is shown in Figure 8. In this section of the graph, it is interesting to note that role $r8$ containing permission set {muads-u, sututor, tute} has been referenced by eight other distinct roles. Only one user in the initial graph infrastructure was assigned only permissions {muads-u, sututor, tute}. The produced hierarchical infrastructure of even a small sample set is capable of identifying permission sets that represent effective roles.

The algorithm was also tested on a larger permission set. This test set contained 338 users, 1297 permissions and 898 permission assignments. When this test set was placed into the initial graph, there were 1973 nodes from 338 users, 338 initial roles and 1297 permissions. There were 1235 edges from 898 permission assignments and 338 role to user assignments. The initial starting metric is $1236 + 338 = 1574$.

During Graph Optimisation on this test set, 716 merges operations were completed to produce a graph with 1965 nodes and 1036 edges. This final graph was similar in format and architecture to Figure 7, with more breath due to the increase in nodes but similar depth. In the 1965 nodes, 331 were roles. The final graph metric is $1036 + 331 = 1367$.

5. CONCLUSION AND FUTURE WORK

This paper presents a new role engineering technique for extracting roles in a hierarchy to assist existing organisations wishing to implement RBAC for security management. The approach uses matrix decomposition of user permission assignments to obtain the optimal role hierarchy graph. For efficient management, the produced role hierarchy offers a more optimal representation of access control infrastructure.

The proposed algorithm was designed as a bottom up approach that analyses existing user permission assignments to extract roles for hierarchical RBAC. Depending on information present, partial role definitions can also be incorporated into the algorithm to produce a hybrid approach.

The results of the algorithm on real permission assignments show successful role engineering to produce effective roles in a comprehensive role inheritance hierarchy. Future work may incorporate the analysis of access logs to identify which edges offer better representations of a user's permission list. In the situations where permission assignments are assigned erroneously, no longer referenced or no longer reflect the correct access permissions of a particular user, information in access logs of a user's actual permission access may offer better insight into the user's role within the enterprise. Using this information, edges may be removed from the optimisation graph and a more accurate representation can result.

6. REFERENCES

[1] 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.

[2] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. In *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, pages 121–125, New York, NY, USA, 1997. ACM Press.

[3] D. F. Ferraiolo, R. S. Sandhu, S. I. 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.

[4] A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett. Observations on the role life-cycle in the context of enterprise security management. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 43–51, New York, NY, USA, 2002. ACM Press.

[5] M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 179–186, New York, NY, USA, 2003. ACM Press.

[6] G. Neumann and M. Strembeck. A scenario-driven role engineering process for functional RBAC roles. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 33–42, New York, NY, USA, 2002. ACM Press.

[7] H. Roeckle, G. Schimpf, and R. Weidinger. Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. In *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, pages 103–110, New York, NY, USA, 2000. ACM Press.

[8] G. Saunders, M. Hitchens, and V. Varadharajan. Role-based access control and the access control matrix. *SIGOPS Operating Systems Review*, 35(4):6–20, 2001.

[9] J. Schlegelmilch and U. Steffens. Role mining with ORCA. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 168–176, New York, NY, USA, 2005. ACM Press.

[10] 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.