WHITE PAPER
Role Mining in RBAC via a Graph-based Implementation

Nulli - Identity Management: Seyed Hossein Ahmadinejad, Hadi Ahmadi, Derek Small

Abstract

Role Based Access Control (RBAC) is one of the most popular access control models widely adopted by large organizations. However, engineering high quality roles has proven to be a challenge when implementing RBAC. There are two approaches to engineering roles in RBAC: top-down and bottom-up (aka role mining). This paper, which addresses the latter, a) reviews the alternative approaches to role mining, and b) proposes a graph-based implementation for role mining.

1 Introduction

Protecting resources and assets against unauthorized access has always been a concern for organizations of any size. A typical solution to this is to assign a fine-grained entitlement (i.e., permission) to every resource and then require users who need to access a resource to possess the resource's corresponding entitlement. Building and maintaining such an entitlement system is significantly onerous in large organizations with many protected resources and many users requesting access to them. RBAC [4] was proposed to address this issue by grouping fine-grained entitlements to coarse-grained and thus easier to manage roles. However, how to group or aggregate entitlements to define roles remained to be an open problem. Top-down and bottom-up role engineering are two approaches that can be applied to building an RBAC system.

Top-down role engineering involves analyzing all organization policies and job descriptions, and interviewing employees to define a set of roles that accurately and semantically match job titles and positions. This has been proven to be a very expensive and sometimes never-ending process. Role engineering often needs to be performed at companies who have, for a while, been using RBAC. In such situations, the top-down approach might not be feasible due to the disruptive impact it may have on existing user-entitlement assignments.

The bottom-up approach, on the other hand, requires analyzing the existing set of user-entitlement assignments to find patterns that eventually lead to collecting entitlements into roles. This is a less expensive, less disruptive process that comes at the cost of defining roles that may not necessarily match with real-world job titles and positions. Bottom-up role engineering is often referred to as "role mining" in literature.

Studying existing role mining techniques shows there are challenges that need to be addressed before they can be applied in the real world. The two main problems are: a) large complexity, and b) semantic-free low-quality roles. In practice, some role mining techniques are so computationally complex that they could not be actually used. There are also efficient role mining techniques that usually result in low quality roles that cannot be mapped to real job titles. That is mainly because organizations have not been rigorous enough to assign entitlements only to employees who are required to have those entitlements to complete tasks required of their job titles. This implies that there is so much noise in the existing user-entitlement assignments that the mined roles will always be of low-quality and with no semantic.

This work is based on a real-world implementation of role mining for an organization, with 5500 employees who are assigned 12000 entitlements through 330000 assignments. A top-down approach is out of the question due its cost and disruptive nature of the approach.

Role mining, in essence, is about finding patterns in user-entitlement assignments, where an assignment refers to a relationship between a user and an entitlement. These user-entitlement relationships can best be defined in a graph. A graph is simply a collection of nodes that are connected through edges or relationships. Given the recent technology advances in graph-databases, this work is an attempt to solve the role mining problem through a graph-based implementation of a few existing relationship-based role mining ideas. This work is not claiming to propose a new role-mining approach, but rather, it puts forward the idea of using graph databases, in particular Neo4j™, to address the complexity of role mining.

In Section 2, we first define a simple graph-based data model used throughout this paper to describe users, entitlements, roles, and their relationships in our RBAC state. Section 3 will then present a measure for evaluating an RBAC state. This will help with determining the performance of a role mining technique. Section 4 will examine three popular

role mining techniques when implemented in Neo4j. Section 5 will explain how we implement a hybrid role mining method that utilizes well-established graph algorithms to mine roles in an RBAC state graph. Mapping roles to employee attributes will be discussed in Section 6, and finally Section 7 will conclude the paper.

2 RBAC State Graph Model

An RBAC state consists of users, entitlements, and roles. Entitlements and roles are assigned to users. A role defines one or more entitlements. This data can be modelled in a graph where nodes are users U, entitlements E, and roles R. There are three types of edges in this graph:

- user-entitlement assignments UE, defining what entitlements are "directly" assigned to what users,
- user-role assignments UR, defining what roles are assigned to what users, and
- role-entitlement assignments RE, defining what entitlements are included in what roles.

Figure 1 depicts a graph modelling of a simple RBAC state where keys are entitlements and hats are roles. Note that an RBAC state could still allow direct user-entitlement assignments.

3 Role Mining Objectives

As discussed earlier, the main motivation behind RBAC is to simplify the maintenance of the access control system. This implies we need to pursue the same objective when we are defining roles. In other words, the number of roles should be minimized. This is the first measure that can be used to evaluate a role mining technique.

Another factor contributing to the complexity of an RBAC state is the number of user-role and role-entitlement assignments because they represent the amount of work system administrators need to apply to maintain roles and assign them to users. Lastly, the number of direct user-entitlements assignments should be minimized because that would increase the complexity of maintaining the RBAC system.

Li *et al.* [6] introduced a measure called Weighted Structural Complexity that captures these factors. Given an RBAC state λ and a set W of weights w_r,w_u, w_p, and w_d, the Weighted Structural Complexity is defined by

$$wsc(\lambda, W) = w_r \times |R| + w_u \times |UR| + w_p \times |RE| + w_d \times |UE| \quad (1)$$

Note that one might set w_d to be larger than other weights because the goal in RBAC is to ideally not have any direct user-entitlement assignment. For example, if we assume all weights are 1, the complexity of the RBAC state illustrated in Figure 1 will be equal to $11 = (2 + 4 + 4 + 1)$.

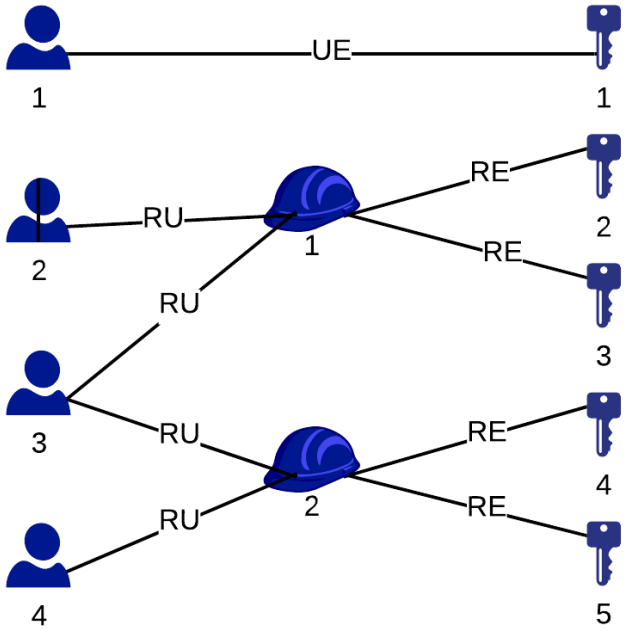

Figure 1: Sample RBAC state graph.

Throughout this work, we use Weighted Structural Complexity to evaluate our role mining algorithms and tune their parameters.

4 Role Mining

Generally, there are three different role mining approaches we can take to define roles. We will describe them in the next three sections. Note that since we model the RBAC state as a graph, the role mining techniques will be also defined against graphs.

4.1 Group Entitlements to Roles

This approach groups entitlements that are assigned to the same users into a role. For example, in Figure 2, entitlements 2 and 3 can form a role because they are assigned to both users 1 and 2. The justification is that if some entitlements are exercised by the same users, it is as if a larger entitlement (that is, a role) has been given to those users, and there is no point in giving individual entitlements to the users. We will call this technique the entitlement-based role mining approach.

Now assume Figure 3 illustrates the RBAC state. The entitlement-based approach cannot create any role because users of entitlements 1 and 2 are not exactly the same. This

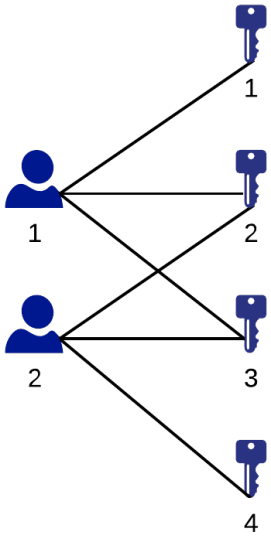

Figure 2: Sample RBAC state graph.

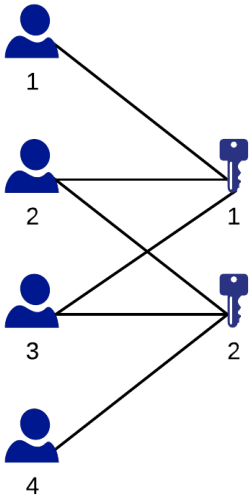

Figure 3: Sample RBAC state graph.

is a weakness of this approach because it only focuses on entitlements. The next section provides another role mining technique that can address this issue.

4.2 Group Users to Roles

This approach is very similar to entitlement-based role mining except that instead of grouping entitlements into roles, it creates roles out of users. In this approach, if some users have the same set of entitlements, they can define a single role which will be assigned to all of them and grant them those

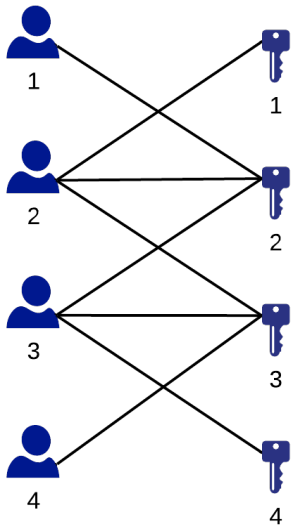

Figure 4: Sample RBAC state graph.

shared entitlements. As an example, users 2 and 3 in Figure 3 have exact same entitlements. As a result, we can use this new role mining approach to create a role that gives them entitlements 1 and 2. For the sake of simplicity, we call this method user-based role mining.

However, this approach has the same weakness as of entitlement-based role mining because it only focuses on entity type. For instance, it cannot create any role for the RBAC state illustrated in Figure 2.

4.3 Group User-Entitlement Assignments to Roles

There are two issues with the two approaches defined above:

(a) In practice, it is very unlikely to find entitlements that are assigned to exact same users or users with exact same entitlements. Figure 4 is an example of an RBAC state where both these approaches fail.

(b) They focus on either users or entitlements and that is why they do not work for the use cases where the other approach works.

To address these problems, this section provides the third role mining approach, called user-entitlement-based role mining which was first introduced by [1]. Instead of considering either entitlements or users, this approach takes both into consideration and group user-entitlement assignments to role. To do so, it reduces the role mining problem to the classic graph problem of finding maximal cliques. A clique is a subgraph where there is an edge between every pair of nodes. In the following we explain how this is done.

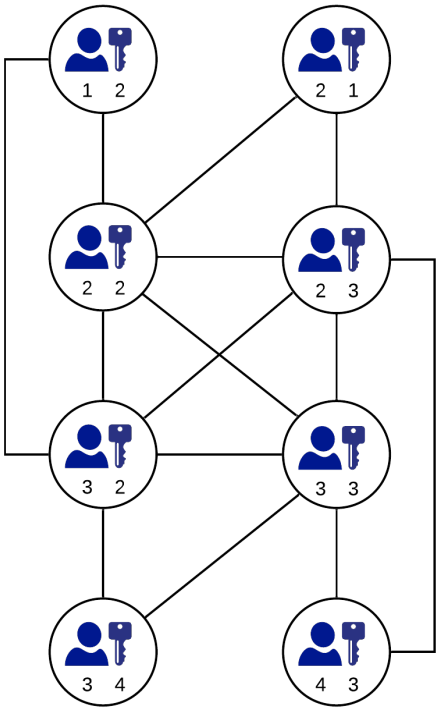

Figure 5: Dual graph for RBAC state of Figure 4

We first need to create a dual graph of the RBAC state graph. For every edge between user u_i and entitlement e_j, a node $u_i e_j$ is added to the dual graph to represent that user-entitlement assignment. Next, an edge will be added between every pair of nodes $u_i e_j$ and $u_m e_n$ if any of the following conditions are met:

(a) u_i and u_m are the same user,

(b) e_j and e_n are the same entitlement, or

(c) in the RBAC state graph, there is an edge between u_i and e_n and between u_m and e_j (i.e., both users u_i and u_m are assigned both entitlements e_j and e_n).

Once the dual graph is created, roles can be identified by finding maximal cliques in the dual graph.

Figure 5 shows the dual graph created for the RBAC state of Figure 4. Now every clique in this graph is a potential role but maximal cliques are preferred because they result in larger roles and consequently, a smaller number of roles, which means less complexity of the RBAC state. The largest clique in Figure 5 consists of u_2e_2, u_2e_3, u_3e_2, and u_3e_3. Figure 6 shows the RBAC state of graph 4 when we run user-entitlement-based role mining against it.

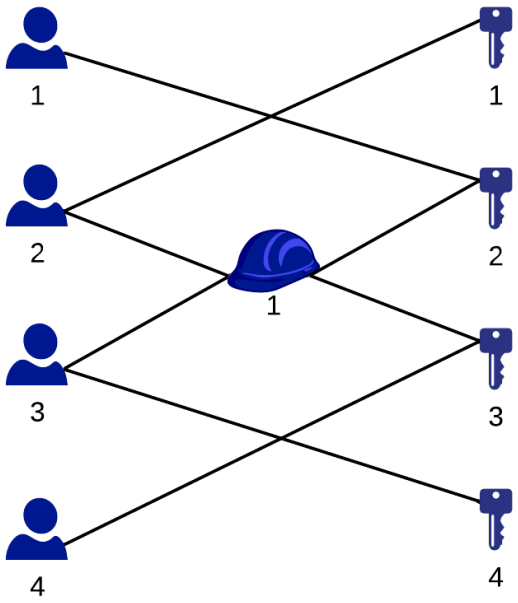

Figure 6: RBAC state of Figure 4 after user-entitlement-based role mining.

5 Implementation

Section 4 described three approaches to discover roles in an RBAC state modelled in a graph. Clearly, the user-entitlement-based approach is the most complete one because not only it can support all use cases where the other two approaches are applicable, but also it works when the other two fail. This section explains how we implement this role mining technique against Neo4j, why we end up with a hybrid approach which is a combination of the first and the third approaches, and how we eventually implement this hybrid approach. Neo4j is a native graph database that grew its popularity in recent years by supporting a rich query language called Cypher and a data science library that provides several graph algorithms applicable to data mining problems in graphs, including role mining.

5.1 Implementation of the User-Entitlement-Based Role Mining

To be able to model an RBAC state graph, we need a data storage solution that lets us store graphs. We choose Neo4j for this purpose for the reasons described above.

The next step is devising an algorithm to find maximal cliques in the state graph. Finding maximal cliques is an NP-Hard problem. We would need to benefit from soft computing or approximation methods to find maximal cliques. Neo4j in particular provides some performance-improved graph algorithms that can be used to tackle the problem of clique find-

ing. Neo4j does not provide an algorithm for finding maximal cliques but there is an algorithm for calculating cluster coefficient [3] of nodes, that is the probability that all neighbours of a node form a clique, which equally means the probability that the node itself belongs to a clique. Algorithm 1 shows the pseudo code of our algorithm for finding maximal cliques in a Neo4j graph.

Algorithm 1 Finding maximal cliques in Neo4j

1: Find cluster coefficient of all nodes.
2: Identify nodes with a cluster coefficient of 1.
3: Sort the identified nodes descendingly based on the degree of the node.
4: **while** there are nodes left in the list **do**
5: Pick the node from the top of the list, and define a new role for the selected node and all its neighbours.
6: Prune the list by removing nodes that were already covered by the newly defined role.

Note that the reason for step 3 is to define the largest roles first. Note also that we designed this algorithm in such a way that we use as many out-of-the-box (OOTB) functionalities in Neo4j as possible.

Running this algorithm against real-world RBAC state graphs turns out to be somewhat inefficient and slow. It is so resource consuming that our Neo4j instance with 8GB of memory eventually crashes due to lack of enough memory. By further studying the clique finding problem, we understood that this problem can be reduced to the graph coloring problem [5], that is coloring nodes of a graph using as few colors as possible where adjacent nodes cannot be assigned the same color. According to [2], finding maximal cliques in a graph is equivalent to coloring of its complement. This equivalence may not be completely perfect but it suits our needs because Neo4j provides a graph coloring algorithm that might perform better than our own implementation of the maximal clique finding algorithm.

To find a role by coloring the RBAC state graph, we still need to build the dual graph, then find its complement graph, color it, and then every group of nodes with the same color will form a role. Our experiments show that this implementation is significantly more efficient than the other one but still could not work against graphs with more than around 5500 edges, given our Neo4j server specification. For that reason, we decide to implement a hybrid approach where the user-entitlement-based approach is used only for defining hard-to-detect roles. For other roles, we use a variation of the entitlement-based approach which will be described in the next section.

5.2 Hybrid Role Mining

Section 4.3 pointed out the two main problems with the entitlement-based approach, that are mainly due to requiring entitlements that are shared by exact same users. It is very unlikely to find such entitlements. However, if we relax this constraint and instead require entitlements to have a large-enough overlap of users, we will be able to define roles for many entitlements. The main advantage of this approach is that it is a lot more efficient than the user-entitlement-based approach. As a result, it can be used as the first step of role mining in an efficient and accurate-enough manner, and then we can feed the remaining entitlements to the user-entitlement-based approach which is more powerful, though less efficient.

To implement this variation of the entitlement-based approach, we can use a similarity measure called Jaccard Similarity [7]. Jaccard Similarity between two sets A and B is defined by

$$\frac{|A \cap B|}{|A \cup B|} \tag{2}$$

Jaccard Similarity between two entitlements can be defined over their corresponding sets of users. Jaccard Similarity of 1 between two entitlements means they are assigned to exactly the same users. But because this rarely happens, a lower threshold can be defined on Jaccard Similarity between entitlements to determine if they are similar enough to be merged together and form a role.

To define Jaccard Similarity in the RBAC state graph, we add new edges between entitlements that are weighted with the Jaccard Similarity of the entitlements at the two ends of the edge. Given that, now the goal is to find groups of entitlements with large enough Jaccard Similarity. In our enhanced RBAC state graph, this translates to detecting communities of entitlements connected through Jaccard Similarity edges. Since community detection is a well-explored graph problem, we can use existing community detection algorithms to find potential roles in the RBAC state graph.

Neo4j has OOTB tools for evaluating Jaccard Similarity between nodes as well as detecting communities.

Algorithm 2 shows a pseudo code for an iterative entitlement-based role mining based on Jaccard Similarity and community detection against the RBAC state graph.

Algorithm 2 Iterative entitlement-based role mining

1: Set j_l to be a lower threshold on Jaccard Similarity.
2: Set $j = 1$ as the current threshold on Jaccard Similarity.
3: **while** $j > j_l$ **do**
4: Add a new edge between every pair of entitlements if Jaccard Similarity between them is at least equal to j.
5: Run community detection algorithms to find communities of similar entitlements.
6: Create a role per community and connect it to corresponding users and entitlements.
7: If no community is found, slightly reduce the value of j.

This algorithm proves to be quite efficient and thorough in covering lots of user-entitlement assignments by defining roles. Because we now allow entitlements with Jaccard Similarities even less than one to form roles, the entitlement-based role mining can even discover harder-to-find roles as well.

Nonetheless, there will be user-entitlement assignments left at the end that could not be covered by the roles defined so far. To tackle these assignments, we feed them to a user-entitlement-based role mining algorithm implemented using the graph coloring technique. Note that the remaining subgraph might still be too large to be directly consumed by the user-entitlement-based role mining. For that reason, we need to break the remaining subgraph of the RBAC state graph to smaller subgraphs, and then feed them separately to the user-entitlement-based role mining method. This can be done again through running community detection algorithms that find communities of entitlements with large-enough Jaccard Similarities.

6 Associating Roles with Employee Attributes

Once roles are identified for an RBAC system through role mining, a typical next phase is to map the engineered roles to employee attributes that are usually defined by the Human Resources (HR) department. This enables automatic provisioning of roles to new hires and even current employees. However, how accurate this can be done depends on the quality of the existing user-entitlement assignments. If an organization has not been rigorous in a) assigning entitlements to employees, and b) attaching well-defined position-dependent HR attributes to employee identities, it would be very unlikely that engineered roles can be associated with any HR attributes. This was our observation in this project too.

Nevertheless, we designed an algorithm that maps roles to HR attributes to assign probabilities to such mappings. Using these probabilities our platform can work as a recommendation engine, rather than an automated role provisioning tool, that can suggest what roles should be assigned to an employee with what probability, given his/her HR attributes. Note that this algorithm was not written to run against a graph structure.

7 Conclusion

Role mining can simplify the goal of transitioning to a well-structured RBAC system. It can achieve this goal through multiple iterations since it relies heavily on the quality of the already-defined entitlements. Organizations can use the result of every iteration of role mining to further improve the quality of their data. This may require administrators to apply manual steps to the process. Over time these manual process steps should be minimal as the quality of the data should improve. Having the RBAC state graph modeled as graph in Neo4j allows those administrators to use the Neo4j analytical tools and its rich query language, Cypher, to explore the data, and identify inaccurate user-entitlement assignments.

This work presented several approaches to role mining. It showed how one can run well-defined graph algorithms, like the community detection algorithm against an RBAC state graph. This proposed hybrid approach combines a more advanced version of the entitlement-based approach with the user-entitlement-based approach to cover as many user-entitlement assignments as possible in defining new roles. The result of this project shows that these techniques are effective if the quality and accuracy of existing user-entitlement assignment data is high enough. Otherwise, the RBAC state needs to be cleaned up from incorrect or inappropriate assignments before role mining can produce meaningful results.

References

[1] Dong, L., Wu, J., Gong, C., and Pi, B. A network-cliques based role mining model. *Journal of Networks 9*, 8 (2014), 2079.

[2] Ene, A., Horne, W., Milosavljevic, N., Rao, P., Schreiber, R., and Tarjan, R. E. Fast exact and heuristic methods for role minimization problems. In *Proceedings of the 13th ACM symposium on Access control models and technologies* (2008), pp. 1–10.

[3] Holland, P. W., and Leinhardt, S. Transitivity in structural models of small groups. *Comparative group studies 2*, 2 (1971), 107–124.

[4] INCITS, A. Incits 359-2004. american national standard for information technology-role based access control, american national standards institute. *Inc., NY, USA* (2004).

[5] Jensen, T. R., and Toft, B. *Graph coloring problems*, vol. 39. John Wiley & Sons, 2011.

[6] Li, N., Li, T., Molloy, I., Wang, Q., Bertino, E., Calo, S., and Lobo, J. Role mining for engineering and optimizing role based access control systems. *Purdue University, IBM TJ Watson Research Center* (2007).

[7] Tan, P.-N., Steinbach, M., and Kumar, V. *Introduction to data mining*. Pearson Education India, 2016.