Migrating from DAC to RBAC

Emre Uzun, David Lorenzi, Vijayalakshmi Atluri, Jaideep Vaidya, Shamik Sural

Migrating from DAC to RBAC

Emre Uzun[1], David Lorenzi[1], Vijayalakshmi Atluri[1], Jaideep Vaidya[1] and Shamik Sural[2]

[1] MSIS Department, Rutgers Business School, USA
{emreu,dlorenzi,atluri,jsvaidya}@cimic.rutgers.edu
[2] School of Information Technology, IIT Kharagpur, India
shamik@sit.iitkgp.ernet.in

Abstract. Role Based Access Control (RBAC) is one of the most popular means for enforcing access control. One of the main reasons for this is that it is perceived as the least expensive configuration with respect to security administration. In this paper, we demonstrate that security administration is not always cheaper under RBAC when compared to the traditional Discretionary Access Control (DAC). If RBAC proves to be beneficial, organizations may choose to migrate from DAC to RBAC. There have been many algorithms developed to generate RBAC configurations from DAC configuration. Although these algorithms provide an RBAC configuration, the *quality* of the generated RBAC configuration could vary among different algorithms and DAC configurations. In this paper, we propose a decision support framework, which provides a basis for comparison among different potential RBAC derivations from DAC to determine the most desirable outcome with respect to the cost of security administration.

1 Introduction

Access control facilitates controlled sharing and protection of resources in an enterprise. While there exist a variety of formal authorization models to meet a variety of organizational requirements in specifying their access control policies, most of today's commercial software including operating systems, database systems, enterprise resource planning and workflow systems offer either Discretionary Access Control (DAC) or Role Based Access Control (RBAC) as their primary mechanism to enforce security. Typically, under DAC, users are directly assigned permissions to access resources, called User Permission Assignment (UPA).

In contrast to DAC where users are directly assigned to permissions, in RBAC, users are assigned to roles (user role assignment, UA), and roles in turn are groups of permissions (permission role assignment PA). It is assumed that this additional layer facilitates easier administration when users and their access permissions change. This is simply because the total number of assignments that the administrator has to manage under RBAC ($|UA| + |PA|$) is typically lower than that of DAC ($|UPA|$), since typically the number of roles is expected to be much smaller than the number of permissions.

Because of the ease of administration as well as its flexibility and intuitiveness, RBAC is one of the most preferred access control models used in computer systems. As a result, many organizations are attempting to migrate from their existing access control model, which is usually DAC, to RBAC in order to enjoy its benefits.

The process of discovering roles from DAC configuration has received significant attention recently. This process, called role mining, is a bottom-up approach that utilizes the existing permission assignments (UPA) of DAC and groups these permissions to obtain roles. There also exist top-down approaches for role discovery, that examine all of the job functions to determine the permissions required to pursue the tasks assigned to them. Although this approach generates meaningful roles that represent various job functions, the overall process of forming roles could be tedious and labor intensive [3],[8]. Moreover, in the presence of certain exceptional cases in permission assignments where a user that belongs to a job function needs an additional permission that none of the other users of the same job function has, there would be problems forming roles that are as generic as possible, while preserving each user's permission requirements. Nevertheless, there is a case study that has results in favor of a top down approach implemented in a bank [15].

The advantage of the bottom-up approach is that it is an unsupervised approach and therefore is a more automated process as opposed to the manual top-down approach and also has the additional benefit of taking care of the exceptional cases discussed above while forming roles. However, in this approach, the existing business processes are ignored and the roles obtained as a result might not represent meaningful job functions. The automated process of obtaining an RBAC configuration with the bottom-up approach is useful for many organizations as they would like to migrate to RBAC to realize its benefits without a huge investment of time and money as in the case of top-down approach. Therefore, in this paper, we take the path of the bottom-up approach to migrate from DAC to RBAC.

However, we argue in this paper that migrating from DAC to RBAC may not always be beneficial with respect to reducing the administrative load. Therefore, before one jumps on to such a decision to migrate from DAC to RBAC, some key questions should be asked at that point:

1. *Is migrating to RBAC from an existing DAC configuration a wise option?* In this paper, we demonstrate that, while RBAC eases the burden of security administration for certain access control configurations, DAC may be more beneficial for other configurations. Towards this end, we present a decision support model to help with this analysis.
2. *If migrating to RBAC is more beneficial, what is the best suitable approach for a given DAC configuration to help with this migration from DAC to RBAC?* In this paper, we study a number of such approaches and present our experimental results. These approaches include three previously proposed algorithms – FastMiner [21], MBC [3] and ORCA [16], as well as one new algorithm, called DEMiner, proposed in this paper.

Towards this end, in this paper, we make the following major contributions: First, we propose a framework that can be utilized in three different ways: (1) Determining whether migrating to RBAC from an existing DAC configuration is beneficial, (2) Comparing alternative RBAC derivations of the same DAC configuration to pick the most desirable one in terms of administrative load, and (3) Comparing the results of different role mining algorithms.

Our framework can serve as a *Decision Support System* to aid organizations to make an informed decision about whether it is desirable to switch to an RBAC configuration. Using our framework, organizations will be able to determine if they really need to implement RBAC, and if so which alternative configuration provides the best "fit" to its specific policies. Our framework comprises a number of different cost metrics derived from the UPA and the corresponding RBAC derivation from a role mining approach to determine if it is worth migrating.

Second, we propose a new and pragmatic role mining algorithm, called DEMiner, which does not necessarily produce optimal roles, but is quite simple, since it produces disjoint roles.

The paper is organized as follows: In Section 2, we elaborate with examples on the key questions discussed above. In Section 3, we discuss our proposed cost metrics that will aid in distinguishing the performance of different role mining algorithms in terms of ease in administration, along with the main framework for comparison. In Section 4 we review different role mining algorithms and also present a new role mining algorithm. Finally, in Section 5 we provide experimental results of our proposed decision support framework for four different role mining algorithms to give an insight about the administrative load of the RBAC decompositions they produce.

2 Key Questions

Now let us turn our attention to each of the questions we raised in the previous section.

2.1 Is DAC to RBAC migration beneficial at all?

In the following, we show with examples that it is beneficial to migrate from DAC to RBAC for certain DAC configurations and not so for other configurations.

Consider the two example UPAs given in Tables 1 and 4. Essentially, A permission is of the form (resource, access mode). For example a permission p_1 = (File A, read) indicates the read access to File A. In the User-Permission Assignment (UPA) matrix, 1 indicates that a particular user has access to a specific resource in the system. For example, the presence of 1 in the first cell of Table 1 indicates that Alice has permission p_1.

Now, deriving RBAC from DAC is simply to decompose UPA into two matrices, UA and PA. Assume that UPA in Table 1 is decomposed into an RBAC configuration with the minimal number of edges [19]. The decomposition is given in Tables 2 and 3. This decomposition has $|UA|+|PA| = 12$ assignments, whereas

	p_1	p_2	p_3	p_4
Alice	1	1	1	1
Bob		1	1	
Cathy			1	1
David		1		

Table 1. Example UPA 1

	r_1	r_2	r_3	r_4
Alice	1	1	1	
Bob	1			1
Cathy		1		
David	1			

Table 2. UA for Example UPA 1

	p_1	p_2	p_3	p_4
r_1		1		
r_2			1	1
r_3	1			
r_4			1	

Table 3. PA for Example UPA 1

the UPA has only 9 assignments ($|UPA| < |UA| + |PA|$). It is clear that, this optimal RBAC decomposition has more administrative cost than that of DAC.

On the other hand, consider the DAC configuration given in Table 4. The optimal RBAC decomposition with minimum number of edges is given in Tables 5 and 6. This RBAC configuration has $|UA|+|PA| = 9$ assignments. However, the DAC configuration has 10 assignments, hence making the RBAC configuration better than the DAC in terms of administrative cost ($|UPA| > |UA| + |PA|$).

	p_1	p_2	p_3	p_4
Alice	1	1	1	1
Bob	1	1		
Cathy	1	1		
David			1	1

Table 4. Example UPA 2

	r_1	r_2
Alice	1	1
Bob	1	
Cathy	1	
David		1

Table 5. UA for Example UPA 2

	p_1	p_2	p_3	p_4
r_1	1	1		
r_2			1	1

Table 6. PA for Example UPA 2

The above examples clearly demonstrate that for certain DAC configurations, migrating to RBAC will reduce the administrative load, whereas for certain other DAC configurations migrating to RBAC will indeed increase the administrative load. Therefore, blindly switching to RBAC without first evaluating its benefits is not a good idea.

2.2 What is the best DAC to RBAC configuration that minimizes the administrative burden?

A poor RBAC decomposition, regardless of how fast it is obtained, could degrade the usefulness of RBAC up to the very extreme point where the corresponding DAC is significantly better in terms of its average effort of administration. For example, consider two alternative decompositions to the UPA given in Table 1. The first one is obtained by minimizing the number of roles (given in Tables 7 and 8) and the second one is an arbitrary decomposition – no specific optimization performed (given in Tables 9 and 10).

Although the first decomposition is an optimal decomposition for the UPA, is clear that this decomposition creates redundancy, since the permission-role assignment relation actually is the same as the UPA itself, and the user-role

	r_1	r_2	r_3	r_4
Alice	1			
Bob		1		
Cathy			1	
David				1

Table 7. Alternative UA 1 for Example UPA 1

	p_1	p_2	p_3	p_4
r_1	1	1	1	1
r_2		1	1	
r_3			1	1
r_4		1		

Table 8. Alternative PA 1 for Example UPA 1

	r_1	r_2	r_3	r_4	r_5
Alice	1		1	1	1
Bob		1			
Cathy				1	1
David			1		

Table 9. Alternative UA 2 for Example UPA 1

	p_1	p_2	p_3	p_4
r_1	1			
r_2		1	1	
r_3		1		
r_4			1	
r_5				1

Table 10. Alternative PA 2 for Example UPA 2

assignment relation is simply an identity matrix. Also, the total number of assignments is 13, which is more than that of the DAC configuration. The other decomposition also has a higher administrative cost, since not only is there more assignments in total ($|UA| + |PA| = 14$), but also there is one additional role than the previous decomposition. Considering the three decompositions given the UPA in Table 1 (Tables 2,3; 7,8 and 9,10), the one that has the least administrative cost is given in Tables 2 and 3.

There are many different algorithms proposed to construct an RBAC configuration from a DAC configuration. The very first algorithms aim to find a decomposition to a given UPA matrix. CompleteMiner [21], FastMiner [21] and ORCA [16] are some of these algorithms used to perform this process. After the formalization of the role mining problem (RMP) and its variants by Vaidya et al. [19], many different new algorithms that are capable of handling the new objectives are proposed. Many of these new algorithms are basically an adaptation of the solution procedures of an existing problem. Some examples are: Utilizing Minimum Database Tiling Problem, Discrete Basis Problem, Minimum Biclique Cover Problem and Graph Optimization [26], [3], [19]. There is also a study to mine temporal roles when the role assignments are dependent on temporal constraints [11]. While providing the same level of security as long as the resulting RBAC configuration is an exact decomposition of the DAC, each of these algorithms could perform differently, among which some could be superior.

In order make an analytical comparison, Molloy et al. [14] propose a method where, the metric for comparison is a weighted sum of the number of roles, user-role assignments, permission-role assignments, role-role assignments and user-permission assignments that is obtained via a role mining algorithm. Although these components provide crucial information about the decomposition, they do

not provide a comparison perspective with respect to the marginal gain of using a particular decomposition.

3 Cost Metrics for Comparing Alternative RBAC Decompositions

Role Mining is a dataset dependent process in addition to being algorithm dependent. A role mining algorithm can output a satisfactory result on one dataset, yet may perform worse on another. Moreover, expectations on the resulting RBAC decomposition could vary in different applications. For example, a startup company would expect to get a flexible role distribution – more likely with a higher number of roles – to quickly adapt to potential new positions without defining new roles from scratch each time. On the other hand, a more mature company would wish to have a more compact set of roles that precisely define its job functions. Therefore, in order to cover as many different objectives as possible, while remaining as generic as possible, we develop certain cost metrics that are applicable to any dataset for various purposes.

We base our metrics to capture the basic principle of why RBAC should perform better. RBAC imposes the *role* layer to the UPA in order to *group* permissions to reduce *redundancy* in the permission assignments. If there is less redundancy, the administration of the access control system would get better. So, if a specific RBAC decomposition does not address this *redundancy* or if the UPA does not have any *redundancy*, then RBAC will not provide substantial benefits in terms of its ease in administration. This issue could be of many different types, such as having roles with only a single permission, having more roles than the number of permissions, or even having a decomposition that has more assignments than the UPA. Towards this end, in the following, we present our proposed cost metrics that are used to determine the amount of benefits realized from a transition from DAC to RBAC. We develop some of these metrics on the End-User Benefits of the NIST study related to RBAC [17].

Let the sets U, R, $PRMS$ denote users, roles and permissions, respectively. $UA \subseteq U \times R$ and $PA \subseteq PRMS \times R$ denote user to role and permission to role relations, respectively. We define $UPA \subseteq U \times PRMS$ as the user to permission relation for DAC. Before we present our cost metrics, we provide definitions for some statistical measures that we obtain from DAC and RBAC configurations.

Definition 1. ***Average Number of Users per Role*** *(AUR)**:** Let $(u, r) \in UA$ be a user-role assignment tuple. Number of user assignments for role r is denoted as $UR_r = |\{(u, r) : (u, r) \in UA\}|$. Then $AUR = \sum_{r \in R} UR_r/|R|$ is the average number of users that a role has in a given configuration.*

Definition 2. ***Average Number of Roles per User*** *(ARU)**:** Let $(u, r) \in UA$ be a user-role assignment tuple. Number of role assignments for user u is denoted as $RU_u = |\{(u, r) : (u, r) \in UA\}|$. Then $ARU \sum_{u \in U} RU_u/|U|$ is the average number of roles that a user has in a given configuration.*

Definition 3. ***Average Number of Permissions per Role*** *(APR): Let* $(p, r) \in PA$ *be a permission-role assignment tuple. Number of permission assignments for role* r *is denoted as* $PR_r = |\{(p, r) : (p, r) \in PA\}|$. *Then* $APR = \sum_{r \in R} PR_r/|R|$ *is the average number of permissions that a role has in a given configuration.*

Definition 4. ***Average Number of Permissions per User*** *(APU): Let* $(u, p) \in UPA$ *be a user-permission assignment tuple. Number of permission assignments for user* u *is denoted as* $PU_u = |\{(u, p) : (u, p) \in UPA\}|$. *Then* $APU = \sum_{u \in U} PU_u/|U|$ *is the average number of permissions that a user has in a given configuration.*

Under these definitions, we present four different cost metrics that we utilize in our decision support framework:

Amount of generic roles (GEN): First, we define exclusive roles (EXC). A role is said to be *exclusive* if that role is assigned to only a *few* users to reflect exceptional situations. Usually, they are assigned to a single user to provide access to an exclusive task that is not covered by any other roles. For example, the roles r_1, r_3 and r_4 in the example decomposition given in Table 3 are said to be exclusive roles. A high number of exclusive roles is considered undesirable from a management perspective. For this metric, we utilize AUR and APR to determine for a specific role $r \in R$, how different the values of UR_r and PR_r are from AUR and APR. A role $r \in R$ is said to be exclusive if

$$\frac{AUR - UR_r}{AUR} > \epsilon_1 \text{ and } \frac{APR - PR_r}{APR} > \epsilon_2$$

where $\epsilon_1 \in [0, 1]$ and $\epsilon_2 \in [0, 1]$ are threshold values that statistically distinguish an exclusive role. Any role r that satisfies the above relations is in $r \in R_x \subseteq R$. Then, $EXL = \frac{|R_x|}{|R|}$, where R_x is the set of exclusive roles. Any role that is not exclusive is said to be a generic role, hence, the amount of generic roles, GEN, is calculated as $GEN = 1 - EXL$.

Amount of assignments (ASN): The number of user-permission, user-role and permission-role assignments provide a basis for comparison to determine whether RBAC provides a better configuration as opposed to DAC in terms of both usability and security. A high number of total assignments would not only increase the amount of time required to perform assignments but also it makes the system more prone to potential security violations due to misconfiguration. Hence we will compare $|UPA|$ with $|UA| + |PA|$ to determine the amount of reduction in the number of assignments. The amount of assignments, ASN, is the percentage deviation of the total number of user-role and permission-role assignments in RBAC with respect to user-permission assignments in DAC. More formally, the amount of assignments, ASN, is determined by

$$ASN = max(0, \frac{|UPA|-(|UA|+|PA|)}{|UPA|})$$

Although the metric itself can never be negative, the component $\frac{|UPA|-(|UA|+|PA|)}{|UPA|}$ can have negative values, which means there is a need for more user-role and permission-role assignments in RBAC to represent the associated DAC configuration. Therefore, when $ASN = 0$, it means that the DAC configuration is better in terms of amount of assignments.

Average cost of an administrative operation (ADM): The cost of an administrative operation is the amount of changes in the number of user-permission, user-role and permission-role assignments. In RBAC, user-role assignments are modified more often. Permission-role assignments, however, would not usually be altered unless a new role is created. Following the NIST study [17], we base our calculations on ARU as it shows an estimate of how many role assignments are required per user on average. So, the average cost of an administrative operation is

$$ADM = max(0, \frac{APU-ARU}{APU})$$

Sizes of the decomposed matrices (SIZ): This metric is basically the percentage difference of the total size of the RBAC decomposition with respect to its DAC counterpart. Essentially, it is directly dependent on the number of roles $|R|$ in the decomposition. In a majority of the cases where number of roles is far less than the number of permissions, RBAC is preferable. But the configurations given in the example decompositions in Section 1 would make this decision less obvious, as there is just a small difference between the number of user-permission assignments in DAC and user-role and role-permission assignments in RBAC. We calculate the sizes of the decomposed matrices, SIZ, using

$$SIZ = max(0, \frac{|U||PRMS|-(|U||R|+|PRMS||R|)}{|U||PRMS|})$$

This metric also has a similar type of output as ASN. Although SIZ can never be negative, $\frac{|U||PRMS|-(|U||R|+|PRMS||R|)}{|U||PRMS|}$ can get negative values. This means that the decomposed matrices are larger in size than the original user-permission assignment matrix. From an administrative perspective, there will be more possibilities for assignment – since there are more *cells* in the decomposed matrices to *flip* while making updates in role and permission assignments. So, this makes the decomposition more error-prone due to misconfiguration. Also, since the RBAC matrix sizes are greater than the DAC configuration, the decomposition is likely to be sparser than the original configuration which is also undesirable in terms of memory requirements.

Our decision support framework is a function that outputs a percentage value denoting how favorable an RBAC configuration is with respect to the DAC configuration that it is generated from. The function is basically a linear combination of the four cost metrics defined in Section 3.

Definition 5. ***Decision Support Function:*** *The function given by* $f = w_1(GEN) + w_2(ASN) + w_3(ADM) + w_4(SIZ)$*, where* w_1, w_2, w_3 *and* w_4 *are nonnegative weights for the corresponding cost component* ($\sum_{i=1}^{4} w_i = 1$)*, denotes the percentage benefit of a given RBAC decomposition with respect to a DAC that is used to generate it.*

We note that, each metric we define in this section could be used individually to determine if an RBAC decomposition is favorable. It is better to have higher values for each metric, as this would imply migrating RBAC is favorable. The decision support function combines these metrics to obtain a more compact framework where the weights are picked based on the preferences of the system administrator or the management that governs the RBAC migration. As discussed previously in this section, a startup company where a handful of employees work, the administrative cost would not so significant, hence w_3 could be set to a low value, whereas in a more mature company with a large number of employees would like to reduce the administrative cost so they should set w_3 a higher value. Similarly, it is more beneficial for a mature company to reduce the number of assignments as much as possible, so the value of w_2 should also be higher. More discussion on the effects of the weights is provided in Section 5.

4 Role Mining Algorithms

In this section, we briefly discuss the role mining algorithms that we use in our experiments.

4.1 FastMiner

FastMiner is proposed by Vaidya et al. [21] and it is the simplified version of CompleteMiner, which is a subset enumeration based role mining algorithm. CompleteMiner has three major steps:

1. Identification of Initial Set of Roles: The algorithm starts with an initial set of roles populated by each user's permissions. So, all users who have the exact same set of permissions are grouped together. These form the initial set of roles, say $InitRoles$.
2. Subset Enumeration: In this phase, all of the potentially interesting roles are discovered by computing all possible intersection sets of all the roles created in the identification step and each unique intersection is placed in the set $GenRoles$.
3. User Count Computation: In this phase, for each generated role in $GenRoles$, the number of users who have the permissions associated with that role is counted.

CompleteMiner has exponential time complexity in terms of the size of the $InitRoles$. However, FastMiner only considers pairwise intersections, hence reducing the time complexity to $O(|U|^2|PRMS|)$.

4.2 MBC

Role Mining using Minimum Biclique Cover is proposed by Ene et al. [3]. They map the UPA into a bipartite graph where there are two sets, users U and permissions $PRMS$ and the assignments in UPA are mapped as edges in the graph, such that if $(u, p) \in UPA$, then there exists an edge between the user u and the permission p in the graph. The users and permissions that are included in a role create a biclique on this graph. Hence, this mapping allows algorithms to solve minimum biclique cover problem to be applied for role minimization problem.

4.3 ORCA

ORCA is proposed by Schlegelmilch and Steffens [16] and is a clustering based role mining algorithm. The clusters are formed using permissions and the users that are assigned to those permissions. The algorithm, then, merges these clusters based on the maximum overlap between the users.

Specifically, the ORCA algorithm begins with a cluster set with one cluster c_r for every permission. The permissions in a cluster are represented by $permissions(c_r)$ and the users associated with the permissions within a cluster are represented by $members(c_r)$. The algorithm starts with a partially ordered set of clusters ($\prec$), as the cluster hierarchy, which is initially set to be an empty set. Then, pairs of clusters with maximal overlap among the users are first merged and placed in the $\prec$. More formally, for every pair of clusters $\langle c_i, c_j \rangle$, $members(\langle c_i, c_j \rangle) = members(c_i) \cap members(c_j)$ and $permissions(\langle c_i, c_j \rangle) = permissions(c_i) \cup permissions(c_j)$. Among the pairs, the one with the highest number of common users and highest number of permissions is selected. The ties are broken randomly. The newly generated cluster (c_n) becomes a super cluster to its child-clusters (c_i, c_j) in the hierarchy. c_i and c_j are no longer considered in the algorithm. The algorithm continues in this fashion until there remains no clusters to merge. ORCA has a time complexity of $O(|PRMS|^2|U|)$.

4.4 DEMiner: A simple role mining algorithm

In this section, we present a new and simple role mining algorithm, called DEMiner, which is focused on obtaining a decomposition with roles as compact as possible while maintaining them disjoint among each other. This will achieve three important outcomes: (1) It provides an upper bound to $|R| \leq |PRMS|$, which is significant in terms of scalability. (2) Implementing Dynamic Separation of Duty constraints becomes easier especially when permission level SoD constraints are defined (3) Administration becomes much easier especially when a role with a specific permission is sought for – there will not be more than one alternative role to determine which one suits the best.

At each iteration the DEMiner algorithm generates a candidate role to be intersected with the existing roles in R. In particular, each candidate role is simply the collection of permissions of a user already assigned in the UPA. The

algorithm begins with two roles, $R = \{\{p : (u_1, p) \in UPA\}, \{p : (u_1, p) \notin UPA\}\}$, namely the permissions of the first user and the remaining permissions. Then, at each iteration the new candidate role is intersected with the roles in R. The UA assignments are performed at each iteration to reflect the updates in R. The algorithm runs until the users set is traced completely. The complete pseudocode for the DEMiner algorithm is shown in Algorithm 1.

Algorithm 1 DEMiner Algorithm

```
Input UPA
Initialize and set R = {{p : (u1, p) ∈ UPA}, {p : (u1, p) ∉ UPA}}
for all Users u ∈ U\{u1} do
   Create a candidate role r* = {p : (u, p) ∈ UPA}
   for all r ∈ R do
      R = R\r
      Set R = R ∪ {{r* ∩ r}, {r*\r}, {r\r*}}
   end for
end for
for all Roles r ∈ R do
   for all Permissions p ∈ r do
      for all Users u ∈ U do
         if (u, p) ∈ UPA then
            UA = UA ∪ {(u, r)}
         end if
      end for
   end for
end for
Output UA, R.
```

The DEMiner generate UA given in Table 12 and PA given in Table 13 for the UPA given in Table 11.

	p_1	p_2	p_3	p_4
Alice	1	1	1	1
Bob		1	1	1
Cathy			1	1
David		1		

Table 11. Example UPA

	r_1	r_2	r_3
Alice	1	1	1
Bob		1	1
Cathy			1
David		1	

Table 12. UA obtained by DEMiner

	p_1	p_2	p_3	p_4
r_1	1			
r_2		1		
r_3			1	1

Table 13. PA obtained by DEMiner

Proposition 1. *The average number of roles created at an arbitrary iteration is bounded by* $|PRMS|/|U|$.

Proof. It is trivial from the output of the algorithm that since all of the roles are disjoint, $|R| \leq |PRMS|$, i.e., one role for each permission. Since there is one

iteration per user, there will be at most $|PRMS|/|U|$ roles that can be generated per iteration on the average.

5 Experimental Results

We perform our experiments using various sized real world data sets obtained from HP Labs [3]. We use FastMiner [21], MBC [3], ORCA [16] and DEMiner as the algorithms. We show how different algorithms can provide different decompositions – as seen through the outputs of our cost component metrics – on the same dataset, so that our framework facilitates an informed decision about whether to switch to a particular RBAC decomposition or not. In the experiments, we set $\epsilon_1 = \epsilon_2 = 0.80$ and all weights equal ($w_i = 0.25$), but specific precedence could be given to the desired metric to obtain customized results.

In Table 14, we report the percentage benefits of the RBAC decompositions. In column "Size", we report the size of the UPA in terms of number of users and number of permissions. The columns "GEN", "ASN", "ADM" and "SIZ" denote the specific value of each cost metric that a dataset and its RBAC decomposition produce for each role mining algorithm. RBAC provides more benefit as these values approach to 1. Conversely, if any of these values are 0, then RBAC configuration will not provide any benefit in terms of administration. For example, the RBAC decomposition for Domino obtained by FastMiner has 81% generic roles and improves the administrative operations by 79%. Also the size of the decomposition is improved by 61%. However, RBAC decomposition has more assignments than the DAC. On the other hand, for Emea, FastMiner produces a decomposition that has 70% generic roles and improves the administrative operations by 99%. But, both the decomposition size and the amount of assignments for RBAC configuration are greater than that of the DAC, making this RBAC decomposition not favorable in terms of these cost metrics. Finally, the column "Total" shows the value obtained from the Decision Support Function.

Figure 1 provides a graphical comparison with respect to total percentage improvements. The decompositions obtained for Firewall 1, Firewall 2, Americas Small and Americas Large outperform the other datasets, regardless of the role mining algorithm. Especially for Firewall 2, there will be a 97% total improvement with respect to the DAC configuration. For these datasets, RBAC is likely to improve administration. On the other hand, the decompositions for Customer dataset is the worst in terms of the percentage benefits with an average of 25% total improvement. The other datasets have around 50% improvement, which could be treated as satisfactory.

When we compare the configurations obtained from different role mining algorithms, we see that they perform close to each other with the exception of the Healthcare and Customer datasets. In Healthcare, MBC and in Customer, ORCA provide a much better decomposition than the other algorithms. Finally, we point out some interesting facts in the details of the Customer dataset. Notice that the cost metrics ASN, ADM and SIZ are all 0% for FastMiner, MBC and DEMiner. This shows that the resulting RBAC decompositions actually have

Table 14. Values of the Cost Metrics for Different Role Mining Algorithms and Datasets

	Size		Fast Miner				
Dataset	**Users**	**Perms.**	**GEN**	**ASN**	**ADM**	**SIZ**	**Total**
Healthcare	46	46	1.00	0.45	0.74	0.26	0.61
Domino	79	231	0.81	0.00	0.79	0.64	0.56
Emea	35	3046	0.70	0.00	0.99	0.00	0.42
Firewall 2	325	590	1.00	0.95	0.98	0.95	0.97
Firewall 1	365	709	0.99	0.83	0.93	0.73	0.87
Apj	2044	1164	0.93	0.13	0.50	0.37	0.49
Customer	10961	284	0.97	0.00	0.00	0.00	0.24
Americas Small	3477	1587	0.76	0.80	0.90	0.82	0.82
Americas Large	3485	10127	0.86	0.32	0.97	0.78	0.73

	Size		MBC				
	Users	**Perms.**	**GEN**	**ASN**	**ADM**	**SIZ**	**Total**
Healthcare	46	46	1.00	0.74	0.91	0.35	0.75
Domino	79	231	0.85	0.00	0.76	0.66	0.57
Emea	35	3046	0.62	0.00	1.00	0.02	0.41
Firewall 2	325	590	1.00	0.95	0.98	0.95	0.97
Firewall 1	365	709	0.97	0.84	0.94	0.71	0.87
Apj	2044	1164	0.89	0.22	0.54	0.39	0.51
Customer	10961	284	0.84	0.00	0.00	0.00	0.21
Americas Small	3477	1587	0.72	0.80	0.89	0.80	0.81
Americas Large	3485	10127	0.76	0.50	0.98	0.84	0.75

	Size		ORCA				
	Users	**Perms.**	**GEN**	**ASN**	**ADM**	**SIZ**	**Total**
Healthcare	46	46	0.79	0.78	0.94	0.00	0.63
Domino	79	231	0.81	0.00	0.00	0.20	0.46
Emea	35	3046	0.98	0.04	0.95	0.00	0.49
Firewall 2	325	590	1.00	0.96	0.99	0.93	0.97
Firewall 1	365	709	0.90	0.91	0.97	0.54	0.83
Apj	2044	1164	0.84	0.35	0.61	0.16	0.49
Customer	10961	284	0.90	0.20	0.22	0.00	0.33
Americas Small	3477	1587	0.89	0.90	0.95	0.61	0.84
Americas Large	3485	10127	0.80	0.75	0.96	0.35	0.71

	Size		DEMiner				
	Users	**Perms.**	**GEN**	**ASN**	**ADM**	**SIZ**	**Total**
Healthcare	46	46	0.95	0.68	0.71	0.17	0.63
Domino	79	231	0.82	0.34	0.66	0.35	0.54
Emea	35	3046	0.94	0.40	0.82	0.00	0.54
Firewall 2	325	590	1.00	0.95	0.97	0.95	0.97
Firewall 1	365	709	0.92	0.86	0.88	0.64	0.82
Apj	2044	1164	0.69	0.16	0.33	0.22	0.35
Customer	10961	284	0.84	0.00	0.00	0.00	0.21
Americas Small	3477	1587	0.79	0.77	0.78	0.68	0.75
Americas Large	3485	10127	0.85	0.78	0.83	0.45	0.73

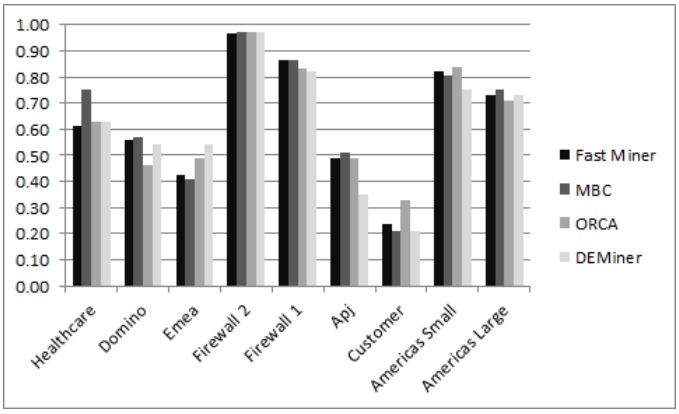

Fig. 1. Comparison of Total Percentage Benefits

more assignments, and the size of the RBAC decomposition is greater than that of the original DAC configuration and the average administrative operation takes longer amount of time. Configurations like these show that there could be cases where one should retain the original DAC configuration in order not to increase the complexity needlessly.

The selection of the weights is important in the analysis and could yield a totally different result for the same role mining algorithm and for the same dataset. Consider the output of FastMiner on Emea. Suppose that emphasis is given on the size of the decomposition and the number of assignments (hence w_2 and w_4 are set to 0.50 and w_1 and w_3 are set to 0.00) then the decision support system will output 0.00% total improvement since the ASN and SIZ are all 0.00 implying that the RBAC decompositions have more roles than permissions and more assignments. On the other hand, if the emphasis is given on the ease in administrative time (ADM), therefore setting $w_3 = 1.00$, the decision support system will output 99% improvement. So, the selection of weights has significant importance on the output.

Finally, we compare the performance of DEMiner with the other algorithms. For Emea, DEMiner performs better (6% or more) than all other algorithms. For Firewall 2, all algorithms perform the same. DEMiner also outperforms ORCA for Domino and Americas Large datasets. Overall, the DEMiner outperforms at least one algorithm in nine datasets and despite being a simple algorithm DEMiner performs within 2% range to the other algorithms on the average over all datasets.

6 Related Work

Role mining is the process of obtaining an RBAC decomposition from a DAC configuration. There are various different algorithms proposed in the literature.

CompleteMiner and FastMiner are proposed by Vaidya et al. [21, 22] for generating a set of candidate roles from an input UPA using subset enumeration. Clustering based algorithm, ORCA, is proposed by [16]. There are also approaches focused on role mining complexity [1, 25], on role hierarchies [12, 13], mining semantically meaningful roles based either on optimization of policy quality metrics [24], consideration of user attributes and use of entitlements [13] or consideration of business information [2]. [23] proposes an approach that deals with scalability issues.

Vaidya et al. propose the problem of deriving an optimal RBAC configuration from a UPA as the role mining problem. Such an optimal configuration consists of a set of roles, a user assignment relation UA, a permission assignment relation PA and optionally a role hierarchy RH. Optimality can be achieved by minimizing one or more metrics. Basic-RMP [19, 20] is the problem of minimizing the number of roles from a given UPA and it has been shown to be NP-complete. Vaidya et al. [19] have mapped it to the Minimum Tiling Problem of databases. Ene et al. [3] show the equivalence between Basic-RMP and the Minimum Biclique Cover (MBC) problem and propose a greedy algorithm to solve Basic-RMP. Huang et al. [7, 6] propose an alternative approach to Basic-RMP using Set Cover Problem. Several variants of Basic-RMP have also been proposed. δ-approx RMP [19, 20], MinNoise RMP [19], Edge-RMP [10] and Extended Role Mining [18] and User-Oriented Exact RMP [9]. Zhang et al. [26] propose a variant of Edge-RMP where the sum of the sizes of the derived role set, UA and PA is minimized. There are also some extensions that focus on the role hierarchy [4, 5, 13], and mining roles on Temporal RBAC [11].

7 Conclusions

Although the RBAC model is perceived to incur lower security administration cost, in certain security policy configurations, DAC may indeed have lower administration cost than that of RBAC. Moreover, the different role mining algorithms that facilitate migration from DAC to RBAC should be carefully examined to determine if the resulting RBAC decomposition provides sufficient improvement with respect to the DAC so that it makes administrative sense to migrate to that particular decomposition. The decision support framework proposed in this paper is a tool that could be utilized by organizations to perform the comparison between different RBAC decompositions to figure out the one that would provide them with the greatest benefit.

8 Acknowledgments

This work is supported in part by NSF under grant number 1018414.

References

1. A. Colantonio, R. Di Pietro, A. Ocello, and N. V. Verde. Taming role mining complexity in rbac. *computers & security*, 29(5):548–564, 2010.

2. A. Colantonio, R. Di Pietro, and N. V. Verde. A business-driven decomposition methodology for role mining. *Computers & Security*, 31(7):844–855, 2012.
3. A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan. Fast exact and heuristic methods for role minimization problems. *SACMAT*, pages 1–10, 2008.
4. M. Frank, J. M. Buhman, and D. Basin. Role mining with probabilistic models. *ACM Transactions on Information and System Security (TISSEC)*, 15(4):15, 2013.
5. Q. Guo, J. Vaidya, and V. Atluri. The role hierarchy mining problem: Discovery of optimal role hierarchies. In *ACSAC*, pages 237–246. IEEE, 2008.
6. H. Huang, F. Shang, J. Liu, and H. Du. Handling least privilege problem and role mining in rbac. *Journal of Combinatorial Optimization*, pages 1–24, 2013.
7. H. Huang, F. Shang, and J. Zhang. Approximation algorithms for minimizing the number of roles and administrative assignments in rbac. In *COMPSACW*, pages 427–432. IEEE, 2012.
8. M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining - revealing business roles for security administration using data mining technology. *SACMAT*, 2003.
9. H. Lu, Y. Hong, Y. Yang, L. Duan, and N. Badar. Towards user-oriented rbac model. In *DBSec*, pages 81–96. Springer, 2013.
10. H. Lu, J. Vaidya, and V. Atluri. Optimal boolean matrix decomposition: Application to role engineering. *ICDE*, pages 297 – 306, 2008.
11. B. Mitra, S. Sural, V. Atluri, and J. Vaidya. Toward mining of temporal roles. In *Data and Applications Security and Privacy XXVII*, pages 65–80. Springer, 2013.
12. I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining roles with semantic meanings. In *SACMAT*, pages 21–30. ACM, 2008.
13. I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining roles with multiple objectives. *TISSEC*, 13(4):36, 2010.
14. I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo. Evaluating role mining algorithms. *SACMAT*, 2009.
15. A. Schaad, J. Moffett, and J. Jacob. The role-based access control system of a european bank: A case study and discussion. *SACMAT*, pages 3 – 9, 2001.
16. J. Schlegelmilch and U. Steffens. Role mining with orca. *SACMAT*, 2005.
17. G. Tassey, M. P. Gallaher, A. C. OConnor, and B. Kropp. The economic impact of role-based access control. *Economic Analysis*, 2002.
18. E. Uzun, V. Atluri, H. Lu, and J. Vaidya. An optimization model for the extended role mining problem. In *DBSec*, pages 76–89. Springer, 2011.
19. J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: Finding a minimal descriptive set of roles. *SACMAT*, pages 175 – 184, 2007.
20. J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: A formal perspective. *ACM Transactions on Information and System Security (TISSEC)*, 13(3):27, 2010.
21. J. Vaidya, V. Atluri, and J. Warner. Roleminer: mining roles using subset enumeration. *CCS*, pages 144 – 153, 2006.
22. J. Vaidya, V. Atluri, J. Warner, and Q. Guo. Role engineering via prioritized subset enumeration. *TDSC*, 7(3):300–314, 2010.
23. N. V. Verde, J. Vaidya, V. Atluri, and A. Colantonio. Role engineering: from theory to practice. In *DBSec*, pages 181–192. ACM, 2012.
24. Z. Xu and S. D. Stoller. Algorithms for mining meaningful roles. In *SACMAT*, pages 57–66. ACM, 2012.
25. D. Zhang, K. Ramamohanarao, S. Versteeg, and R. Zhang. Graph based strategies to role engineering. In *CSIIRW*, page 25. ACM, 2010.
26. D. Zhang, K. Ramamohanrao, and T. Ebringer. Role engineering using graph optimisation. *SACMAT*, pages 139 – 144, 2007.