

Migrating to Optimal RBAC with Minimal Perturbation*

Jaideep Vaidya
MSIS Department and CIMIC
Rutgers University
jsvaidya@rbs.rutgers.edu

Vijayalakshmi Atluri
MSIS Department and CIMIC
Rutgers University
atluri@rutgers.edu

Qi Guo
MSIS Department and CIMIC
Rutgers University
qigu@pegasus.rutgers.edu

Nabil Adam
MSIS Department and CIMIC
Rutgers University
adam@adam.rutgers.edu

ABSTRACT

Devising a complete and correct set of roles has been recognized as one of the most important and challenging tasks in implementing role based access control. A key problem related to this is the notion of goodness - when is a set of roles good? Recently, the *role mining problem* (RMP) has been defined as the problem of discovering an optimal set of roles from existing user permissions. Several different objectives for optimality have been proposed. However, one problem with these definitions is that often organizations already have a deployed set of roles and wish to optimize this set. Even if an optimal set of roles is discovered, if this is widely different, it is impossible to simply throw out the deployed roles and start using the new ones as this may disrupt organizational processes and separation of duty constraints that are defined on roles. Essentially, what is missing is taking *role migration* cost into account when defining optimality, which would allow us to come up with the best suited set of roles.

In this paper, we define a fundamentally different Role Mining Problem that takes the problem of deployed roles into account. We define the Minimal Perturbation RMP as the problem of discovering an *optimal* set of roles from existing user permissions that are *similar* to the currently deployed roles. In order to do this, we discuss the concept of similarity of roles and propose suitable definitions. Solutions also need to be parameterized to set relative weight of similarity and minimality to find the optimal set. We propose a heuristic solution based on the previously developed FastMiner algorithm that meets these requirements. We demonstrate the effectiveness of the algorithm through our experimental results.

*Portions of this work were supported by award CNS-0746943 by the National Science Foundation.

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

SACMAT'08, June 11–13, 2008, Estes Park, Colorado, USA.
Copyright 2008 ACM 978-1-60558-129-3/08/06 ...\$5.00.

Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection—*Access controls*; H.2.8 [Database Management]: Database Applications—*Data Mining*

General Terms

Security

Keywords

RBAC, role engineering, role mining

1. INTRODUCTION

Role-based access control (RBAC), due to its ability to model a wide range of access control requirements, has been widely deployed in most commercial software including operating systems, database systems, enterprise resource planning systems, workflow systems, etc. As a result of its commercial success, it has become a standard to implementing access control in many of today's organizations. Moreover, the concept of a "role" being a commonly understood notion, RBAC has been easily adopted by organizations. Essentially, roles represent organizational agents that perform certain job functions within the organization. Users, in turn, are assigned appropriate roles based on their responsibilities and qualifications [12, 5]. As a result, RBAC simplifies security administration as users' security configuration need not be changed when users leave or join the organization. A role, can be viewed as a set of permissions.

In order to deploy RBAC, an organization must first identify a complete and correct set of roles, and assign users and permissions to these roles. This process, known as *role engineering* [2], has been identified as one of the costliest components in realizing RBAC [6].

There are two basic approaches towards role engineering: *top-down* and *bottom-up*. Under the top-down approach, roles are defined by carefully analyzing and decomposing business processes into smaller units in a functionally independent manner. These functional units are then associated with permissions on information systems. In other words, this approach begins with defining a particular job function and then creating a role for this job function by associating needed permissions. Often, this is a cooperative process

where various authorities from different disciplines understand the semantics of business processes of one another and then incorporate them in the form of roles. Since there are often dozens of business processes, tens of thousands of users and millions of authorizations, this is a rather difficult task. Therefore, relying solely on a top-down approach in most cases is not viable, although some case studies [13] indicate that it has been done successfully by some organizations (though at a high cost).

In contrast, the bottom-up approach utilizes the existing permission assignments to formulate roles. Starting from the existing permissions before RBAC is implemented, the bottom-up approach aggregates these into roles. It may also be advantageous to use a mixture of the top-down and the bottom-up approaches to conduct role engineering. While the top-down model is likely to ignore the existing permissions, a bottom-up model may not consider business functions of an organization [7]. However, the bottom-up approach has the advantage of automating the role engineering process. Role mining can be used as a tool, in conjunction with a top-down approach, to identify potential or candidate roles which can then be examined to determine if they are appropriate given existing functions and business processes.

There have been several attempts to propose good bottom-up techniques to finding roles [8, 14, 19]. However, an inherent problem with all of the above approaches is that there is no formal notion of goodness/interestingness of a role. Recently, [18] has formally defined the role mining problem, and has analyzed its theoretical bounds. Assuming that one can represent the user permissions as a binary matrix, informally, [18] defines the *basic* role mining problem (basic-RMP) as follows: Given a $m \times n$ binary matrix A representing the user-permission assignments (UPA), decompose A into two matrices B and C , where B is a $m \times k$ matrix representing the user-role assignment (UA) and C is a $k \times n$ matrix representing the role-permission association (PA), such that k is minimal.

Minimality is a good notion, since it allows one to formally define the problem. Without semantics (i.e., human expert knowledge), minimality serves as a best approximation for realizing good *descriptive* roles. [18] shows that the decision version of the basic-RMP is NP-complete by reducing the known NP-complete *set basis problem* to this. An optimal set of roles is desirable since it minimizes the administrative burden by reducing the number of roles.

However, adopting this minimal set of roles suffers from the following limitations. First, since this process does not consider job functions or any semantics, the discovered set of roles may not accurately represent the organization's requirements. Therefore, such a role discovery process can only serve as a guideline to security administrators to launch RBAC. Second, this role mining process completely ignores the existing set of roles. This probably is acceptable if an organization is at a preliminary stage, or it is at a stage of completely revamping of its processes. However, this approach of redefining roles from scratch is not permissible for organizations that have an RBAC in place. This is because, if the organization has spent considerable effort in role engineering (perhaps using the top-down approach), these efforts are simply a waste. Moreover, changes to the role set may result in drastic changes to the way in which organizations conduct their businesses. This may cause disruptions to the proper functioning of the organizations. These disruptions

may be in the form of changes to the organizational processes and separation of duty constraints that are defined on roles. Furthermore, there may be some previously defined roles that cannot be changed or removed due to certain functional restrictions.

Therefore, although one would like to use an optimal set of roles, migrating to this new set of roles from existing set of roles (called deployed roles or *DROLES*) should cause as less disruption as possible. In this paper, we propose an approach that identifies a set of roles (*ROLES*) that are as close as possible to both *DROLES* and the optimal set of roles. We denote this problem as the *minimal perturbation RMP* and use a similarity metric based on Jaccard coefficient to formalize this problem.

Essentially, the solution to the minimal perturbation RMP provides a formal means to combine both top-down and bottom-up role engineering approaches. Additionally, even if RBAC is in place, since it evolves over time, it gives a formal way to measure the goodness of the current RBAC assignments.

As a simple example, consider the hypothetical organization shown in Figure 1 consisting of 16 users and 12 permissions. 1(a) shows the organization *UPA*. The *UPA* can be completely described by the following 8 roles, i.e., the optimal set of roles are: $\{\{1, 3, 9\}, \{11, 3, 8\}, \{3, 8, 9\}, \{4, 5, 8\}, \{10, 5, 7\}, \{1, 10\}, \{2\}, \{3, 4\}\}$. Assume that the deployed roles (shown in Figure 1(b)) consist of each permission in its own role; i.e., there are 12 deployed roles. Hence $DROLES = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}, \{10\}, \{11\}, \{12\}\}$. The minimal perturbation RMP also has a weight parameter that allows us to select how close the generated roles should be to the deployed set of roles. With the weight set to 0, the following 9 roles are generated by our algorithm: $\{\{1, 3, 8, 11\}, \{1, 3, 9\}, \{1, 10\}, \{2, 3\}, \{3, 4\}, \{3, 5, 7, 8, 10, 11\}, \{3, 8, 9\}, \{4, 5, 8\}, \{5, 7, 10\}\}$. With the weight set to 0.2, the following 10 roles are generated: $\{\{1\}, \{1, 3, 8, 11\}, \{2, 3\}, \{3, 4\}, \{3, 8, 9\}, \{3, 8, 9, 11\}, \{3, 9\}, \{4, 5, 8\}, \{5, 7, 10\}, \{10\}\}$. Finally, with the weight set to 1, the following 11 roles are generated: $\{\{1\}, \{1, 3, 8, 11\}, \{2, 3\}, \{3\}, \{3, 8, 9, 11\}, \{3, 9\}, \{4\}, \{5\}, \{5, 7, 10\}, \{8\}, \{10\}\}$. We can see that though the number of roles is increasing, the roles generated are getting closer to the set of deployed roles. Figure 1(c) shows the roles generated with the weight set to 0.2 along with the correspondence to the original set of deployed roles. In general, by appropriately tuning the weight parameter, our algorithm allows one to come up with a suitable set of roles in between the set of optimal roles and deployed roles.

The rest of this paper is organized as follows. In Section 2, we review the RBAC model and some preliminary definitions employed in the paper. In Section 3, we define our minimal perturbation RMP and discuss the results about their complexity. In Section 4, we present the algorithm to arriving at minimal perturbation RMP from the given set of deployed roles and present our experimental results that demonstrate the usefulness of the similarity metric in arriving at optimal with minimal perturbation with respect to the deployed set of roles. In Section 5, we discuss the rationale behind employing the minimal perturbation RMP. Section 6 discusses the related work in this area. Finally, Section 7 provides conclusions and the future work to be done in this direction.

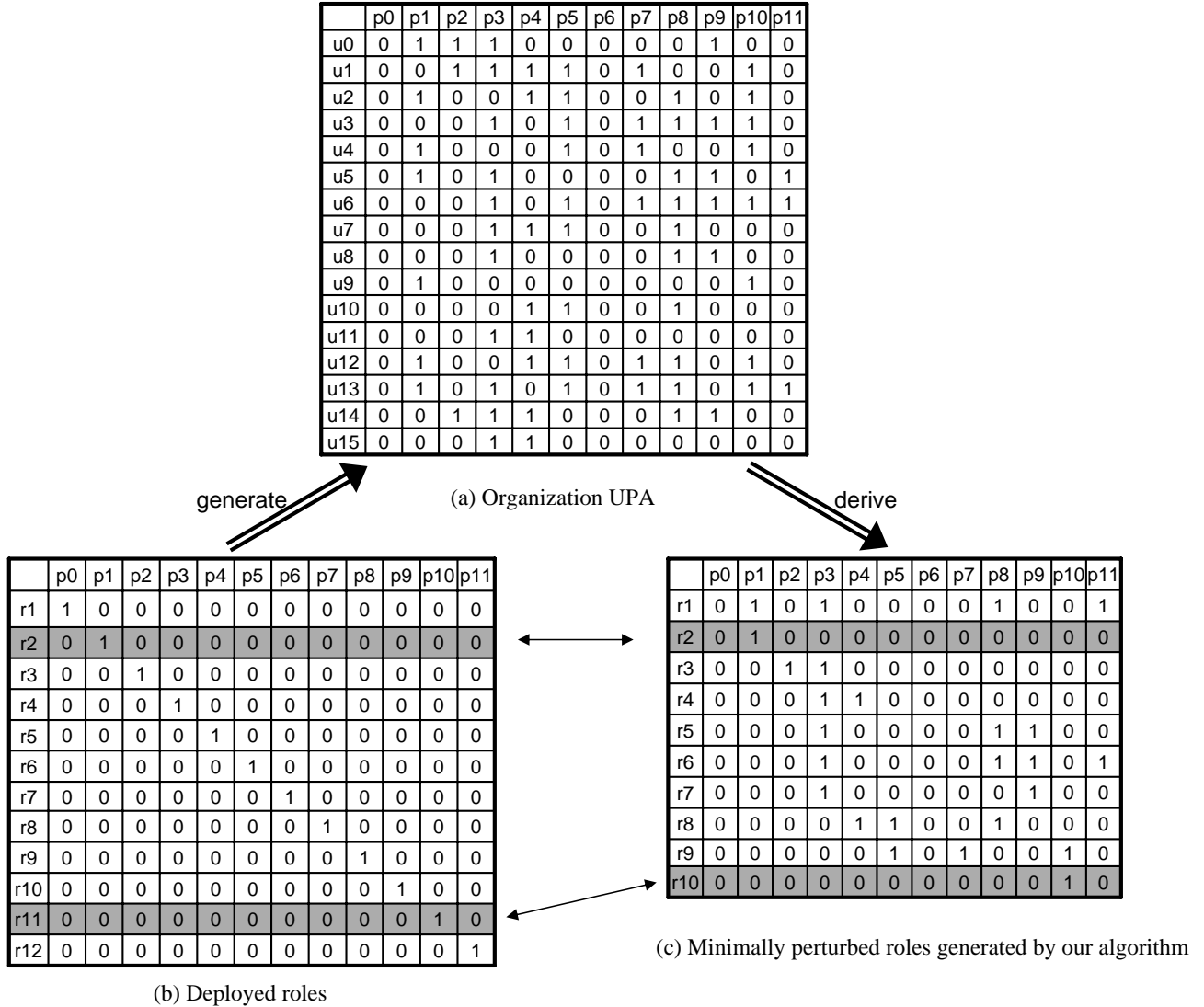


Figure 1: Hypothetical Organization

2. PRELIMINARIES

In this section, we review the basic RBAC model, the formal role mining problem identified in [18] and the Jaccard Coefficient.

2.1 Role Based Access Control Model

We adopt the NIST standard of the Role Based Access Control (RBAC) model [5]. For the sake of simplicity, we do not consider sessions, role hierarchies or separation of duties constraints in this paper. In other words, we restrict ourselves to $RBAC_0$ without considering sessions.

DEFINITION 1 (RBAC).

- $U, ROLES, OPS$, and OBJ are the set of users, roles, operations, and objects.
- $UA \subseteq U \times ROLES$, a many-to-many mapping user-to-role assignment relation.

- $PRMS$ (the set of permissions) $\subseteq \{(op, obj) \mid op \in OPS \wedge obj \in OBJ\}$
- $PA \subseteq ROLES \times PRMS$, a many-to-many mapping of role-to-permission assignments.¹
- $UPA \subseteq U \times PRMS$, a many-to-many mapping of user-to-permission assignments.
- $assigned_users(R) = \{u \in U \mid (u, R) \in UA\}$, the mapping of role R onto a set of users.
- $assigned_permissions(R) = \{p \in PRMS \mid (p, R) \in PA\}$, the mapping of role R onto a set of permissions.

¹Note that in the original NIST standard [5], PA was defined as $PA \subseteq PRMS \times ROLES$, a many-to-many mapping of permission-to-role assignments.

2.2 The Basic Role Mining Problem

Given m users, n permissions and k roles (i.e., $|U| = m$, $|PRMS| = n$, $|ROLES| = k$), the user-to-role mapping (UA) can be represented as an $m \times k$ boolean matrix where a 1 in cell $\{ij\}$ indicates the assignment of role j to user i . Similarly, the role-to-permission mapping (PA) can be represented as an $k \times n$ boolean matrix where a 1 in cell $\{ij\}$ indicates the assignment of permission j to role i . Finally, the user-to-permission mapping (UPA) can be represented as an $m \times n$ boolean matrix where a 1 in cell $\{ij\}$ indicates the assignment of permission j to user i .

In the following, we present the formal definition of the basic-RMP [18].

DEFINITION 2 (ROLE MINING PROBLEM (RMP)). *Given a set of users U , a set of permissions $PRMS$, and a user-permission assignment UPA , find a set of roles, $ROLES$, a user-to-role assignment UA , and a role-to-permission assignment PA consistent² with UPA and minimizing the number of roles, k .*

Given the user-permission matrix, the above basic-RMP asks us to find a user-to-role assignment UA and a role-to-permission assignment PA such that UA and PA describe UPA while minimizing the number of roles. Put another way, it asks us what is the minimum number of roles necessary to fully describe the given data (and what are those roles, and the corresponding user assignments)?

For example, considering the UPA shown in figure 2(a), the solution to the basic-RMP is nothing but the UA and PA with minimum number of roles as shown in figure 2(b).

2.3 Jaccard Coefficient

The Jaccard coefficient is a well known statistic used for comparing the similarity and diversity of sample sets. Specifically, the Jaccard's coefficient (measure of similarity) and the Jaccard's distance (measure of dissimilarity) are measurements of asymmetric information on binary (and non-binary) variables. For non-binary data, Jaccard's coefficient can be computed as follows:

Given two sets A and B , the Jaccard Coefficient $JC_{AB} = \frac{|A \cap B|}{|A \cup B|}$.

For example, let $A = \{a, b, c, d, e\}$ and $B = \{a, d, e, f, g\}$. Then $A \cap B = \{a, d, e\}$ and $A \cup B = \{a, b, c, d, e, f, g\}$. Therefore, $JC_{AB} = \frac{3}{7} = 0.429$.

While the above Jaccard coefficient computes the similarity between two sets, the dissimilarity between two sets, called the Jaccard distance $JD_{AB} = 1 - JC_{AB}$. In the above example, $JD_{AB} = \frac{4}{7} = 0.571$.

3. MINIMAL PERTURBATION RMP

In this section, we formalize the notion of minimal perturbation RMP by first defining the similarity / distance between pairs of roles and pairs of sets of roles. Since a role

²In [18], it was defined as 0-consistent because it defines additional variations of the RMP, such as the δ -approx RMP and MinNoise RMP. Although these variations are also applicable to the minimal perturbation RMP, in this paper, we limit our attention to the simple minimal perturbation RMP problem.

is nothing but a set of permissions, we can use the Jaccard coefficient to measure similarity / distance. The similarity and distance between a pair of roles can be formulated as follows.

DEFINITION 3 (ROLE-ROLE SIMILARITY / DISTANCE). *For any two roles R_1 and R_2 , let $P_1 = \text{assigned_permissions}(R_1)$ and $P_2 = \text{assigned_permissions}(R_2)$, denote the set of permissions assigned to R_1 and R_2 , respectively. Let $n_i = |P_1 \cap P_2|$, and $n_u = |P_1 \cup P_2|$. We define similarity between R_1 and R_2 as $\text{sim}(R_1, R_2) = n_i/n_u$, and the distance between them as $d(R_1, R_2) = 1 - \text{sim}(R_1, R_2)$.*

For example, consider two roles $R_1 = \{p_1, p_2, p_3\}$ and $R_2 = \{p_2, p_4, p_5\}$. Since there is only one permission common to both R_1 and R_2 (i.e., p_2) and five permissions in total, $\text{sim}(R_1, R_2) = 1/5 = 0.2$. Correspondingly, the distance $d(R_1, R_2) = 1 - \text{sim}(R_1, R_2) = 1 - 0.2 = 0.8$.

The above definition has many favorable properties. When two roles are identical, their similarity is computed to be 1. When two roles have mutually exclusive permission sets, their similarity is 0. In general, the similarity (and distance) is a value between 0 and 1. We can also easily extend the definition to measure distance between two sets of roles. First, we define the similarity/distance between a role and a set of roles.

DEFINITION 4 (ROLE-ROLES SIMILARITY/DISTANCE). *Given a role R_1 and a set of roles SR_1 , we define the similarity between R_1 and SR_1 as follows: $\text{sim}(R_1, SR_1) = \max_{R_2 \in SR_1} \text{sim}(R_1, R_2)$ and the distance $d(R_1, SR_1) = 1 - \text{sim}(R_1, SR_1)$.*

In this definition, the similarity is computed as the maximum similarity between the role and any role in the other set. We choose this as an effective measure since if an identical role is found in the other set, the similarity reported is 1. Similarly, if no role is found with even one overlapping permission, the similarity reported is 0. For example, consider the role $R_1 = \{p_1, p_2, p_3\}$ and the set of roles $SR_1 = \{\{p_4, p_5\}, \{p_3, p_6\}, \{p_1, p_2, p_4, p_7\}\}$. In this case the similarity would be computed as follows: $\text{sim}(R_1, \{p_4, p_5\}) = 0/5 = 0$. Similarly, $\text{sim}(R_1, \{p_3, p_6\}) = 1/4 = 0.25$. Finally, $\text{sim}(R_1, \{p_1, p_2, p_4, p_7\}) = 2/5 = 0.4$. Since the maximum similarity is 0.4, $\text{sim}(R_1, SR_1) = 0.4$. While we choose the maximum similarity, we could easily follow another way of aggregation such as computing the average instead of the maximum, etc., if the situation warrants it. For now, we think that this is a suitable general purpose measure.

Measuring the similarity / distance between sets of roles is a significantly more complex task. It is unclear whether a single role should correspond to only one other role or to a set of roles. Similarly, it is not clear if a role can be involved in more than one matching (i.e., once a role is picked as part of a suitable match, can it be considered for matching with another role(s)?).

If we restrict matches to single roles, an easy way to define this would be by extending the earlier *Role - Role* metric. Thus, we could simply define the similarity as the number of identical roles divided by the number of total roles. The following definition formalizes this.

DEFINITION 5 (ROLES-ROLES SIMILARITY/DISTANCE). *Given two sets of roles SR_1 and SR_2 , we define the similarity between SR_1 and SR_2 as follows: Let $n_i = |S|$ where*

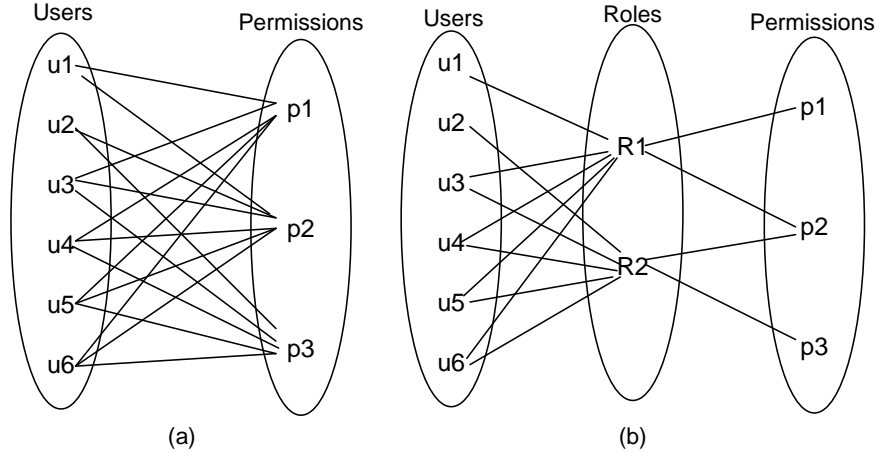


Figure 2: Basic RMP

($S = \{(u, v)\}$, such that $u \in SR_1, v \in SR_2$ and $sim(u, v) = 1$). Thus $n_i = |SR_1 \cap SR_2|$. Also, let $n_u = |SR_1 \cup SR_2|$. We define $sim(SR_1, SR_2) = n_i/n_u$ and $d(SR_1, SR_2) = 1 - sim(SR_1, SR_2)$.

For example, consider two sets of roles $SR_1 = \{\{p_1, p_2\}, \{p_2, p_4\}, \{p_3, p_4, p_5\}\}$, and $SR_2 = \{\{p_2, p_4\}, \{p_3, p_4, p_6\}\}$. In this case, only one role in SR_1 is identical to a role in SR_2 (the role $\{p_2, p_4\}$). Therefore, according to the earlier definition, the similarity would be calculated as $1/4 = 0.25$. While this is permissible for identical roles, it completely disregards the similarity between roles. For example, the roles $\{p_3, p_4, p_5\}$ and $\{p_3, p_4, p_6\}$ differ only in one permission but still do not contribute to the overall similarity. In general, this could be worse, especially with larger roles which may not be identical but are very similar. Instead, we would like to have a definition of similarity that takes non-identical role-role similarity into account as well.

To do this, we extend our similarity/distance measure by using the prior defined similarity measure between a role and a set of roles. Instead of counting identical roles, we take the maximum similarity each role has with the other set of roles, and average across all of the roles. If there is an identical role in the earlier set, the earlier defined similarity metric will also give 1 as desired. The advantage is that this also works for non-identical roles. To capture this, we now provide an alternative definition for the ROLES-ROLES similarity/distance, which is as follows:

DEFINITION 6. (*Granular ROLES-ROLES Similarity / Distance*). Given two sets of roles SR_1 and SR_2 , we define the similarity between SR_1 and SR_2 as follows: if the sizes of the two sets are not equal, without loss of generality, assume that SR_1 is the smaller set. Then, $sim(SR_1, SR_2) = avg_{R \in SR_1} sim(R, SR_2)$.

For example, consider two sets of roles $SR_1 = \{\{p_1, p_2\}, \{p_3, p_4\}\}$ and $SR_2 = \{\{p_1, p_2\}, \{p_3, p_5\}\}$. The similarity between the four possible pairs are: $sim(\{p_1, p_2\}, \{p_1, p_2\}) = 2/2 = 1$, $sim(\{p_1, p_2\}, \{p_3, p_5\}) = 0/4 = 0$, $sim(\{p_3, p_4\}, \{p_1, p_2\}) = 0/4 = 0$, and $sim(\{p_3, p_4\}, \{p_3, p_5\}) = 1/3 = 0.33$. Therefore, $sim(SR_1, SR_2) = (1 + 0.33)/2 = 0.665$. In our algorithm presented in the next section, we employ this granular ROLES-ROLES similarity measure rather than the prior ROLES-ROLES similarity measure.

The similarity measure defined above is still quite straightforward. In general, we may wish to define similarity between sets of roles in a much more sophisticated fashion. For example, we may want to count the average similarity, while eliminating the outliers (or while tolerating a certain number of low similarity roles, etc.). For now we do not bother about this. Also note that both of the above measures still assume that a role can only be mapped to one other role. In general this is not true. For example, let one set has the role $\{p_1, p_2, p_3\}$ and the other set has the roles $\{\{p_1\}, \{p_2, p_3\}\}$. While our similarity measures will give a score of 0.33 and 0.66 respectively for the two roles, it should be clear that taken together, the sets of roles are quite similar. However, this is also significantly more complex and we leave considerations of this sort to future work.

Now that we know how to measure similarity, we still need to find a way to incorporate it into the role selection. Thus, given two different objectives, we would like to define a way to combine the two objectives so as to minimize a single global function. For the Basic RMP, we minimize the number of roles. Now, we would like to additionally minimize the distance between identified roles and deployed roles. Many ways of combining the two objectives are possible. An additional problem here is that the similarity/distance is a number between 0 and 1 while the number of roles is between 0 and $\min(m, n)$ where m is the number of users and n is the number of permissions. An easy way to resolve this is to use a linear combination of the two. We would also like to weigh the relative contribution of the two factors. Therefore we define our combination function as follows:

DEFINITION 7 (COMBINATION FUNCTION). Given some number of roles k , and a distance score d between two sets of roles, we define a combination function $CF(k, d) = (1 - w)k + wk d$ where w is a user defined weighting coefficient for the similarity.

In the above definition, the distance is multiplied by k to bring both numbers into a comparable range. With all of the prior definitions in place, we can now define the Minimal Perturbation RMP as follows:

DEFINITION 8 (MINIMAL PERTURBATION RMP). Given a set of users U , a set of permissions $PRMS$, a user - permission assignment UPA , and a deployed set of roles

DROLES, find a set of roles *ROLES*, a user-to-role assignment *UA*, and a role-to-permission assignment *PA* consistent with *UPA* such that it minimizes the combination function of the number of roles and the distance between *ROLES* and *DROLES* (i.e., minimizing $CF(|ROLES|, d(ROLES, DROLES))$).

An interesting aside is that in all of the above distance / similarity definitions, we assume that all permissions are given equal weight. However, in real situations this may not be the case. However, our definitions can be easily extended to include permission weighting by changing the basic Role-Role definition to include it. Permission weights could be set by the user or even automatically identified from the *UPA* according to some strategy.

Also, while we have defined the Minimal Perturbation RMP in terms of the Basic RMP, the same idea applies to all of the variants – δ -approx RMP and the MinNoise RMP – proposed in [18]. All these problems can be extended to include the concept of deployed roles in a similar manner to these variants.

3.1 Complexity

The Minimal Perturbation RMP is an NP-hard problem. This follows from the observation that the Basic RMP is a special case of the Minimal Perturbation RMP (with $w_1 = 1$ and $w_2 = 0$). Since the Basic RMP is known to be NP-hard [18], the Minimal Perturbation RMP is also NP-hard.

4. ALGORITHM

We now present a heuristic algorithm to find a set of roles satisfying the minimal perturbation RMP objective. The algorithm proceeds in two independent phases. In the first phase, we generate a set of candidate roles. This is currently done using the FastMiner algorithm developed by Vaidya et al. [19]. FastMiner generates candidate roles simply by intersecting all unique user pairs. In general, any technique can be used to generate the candidate roles. In the second phase, we select the final roles from among these candidates. For this selection, we follow a greedy strategy. Essentially, the best candidate role is selected from the remaining candidate roles until the original *UPA* can be completely reconstituted. Thus, in each iteration, for every remaining candidate role we compute the uncovered area of that role as well as the similarity of that role to the deployed roles. The uncovered area of a role can be easily computed by finding the number of 1s in $M(UPA)$ that are not already covered by any of the roles in *ROLES*. The similarity of the role to *DROLES* is computed as in Definition 4, by finding the maximum similarity to any of the roles in *DROLES*. The weighted score is then calculated by taking the area, similarity, and weight into consideration. One more optimization is possible to improve efficiency. If the set of roles is sorted in descending order by the area of the roles, the length of each iteration can be reduced. When a new candidate role is considered, if the total area of that role is less than the currently seen maximum score, we know that it is impossible for that role to be the best (since the similarity, and weight are bounded between 0 and 1, the total area gives the upper bound on the maximum score from that role). Indeed, since the roles are sorted, we know that none of the roles following this can be the best role either. Therefore, we im-

Algorithm 1	Minimal	Perturbation
<i>RMP(UPA, DROLES)</i>		

Require: User-Permission assignment, *UPA*

Require: Initial set of deployed roles, *DROLES*

Require: Weight factor for similarity, $w \in [0, 1]$

```

1: Create a candidate set of roles, CROLES, using the
   FastMiner [19] algorithm {Create candidate set of roles}
2: Sort CROLES according to the area of each role
3: ROLES  $\leftarrow \phi$ 
4: while UPA is not covered do
5:   BestRole  $\leftarrow \phi$ 
6:   BestScore  $\leftarrow 0$ 
7:   for each role C in CROLES do
8:     if area(C) < BestScore then
9:       Exit the FOR loop {Since max. similarity can be
        1, we have already found the best possible role}
10:    end if
11:    carea  $\leftarrow$  UncoveredArea(C, UPA, ROLES)
        {compute uncovered area of candidate role}
12:    Compute csim  $\leftarrow$  Similarity(C, DROLES)
13:    Score  $\leftarrow (1 - w) \cdot \text{carea} + w \cdot \text{carea} \cdot \text{csim}$ 
14:    if Score > BestScore then
15:      BestScore  $\leftarrow$  Score
16:      BestRole  $\leftarrow$  C
17:    end if
18:  end for
19:  ROLES  $\leftarrow$  ROLES  $\cup$  C {Add C to the set of roles,
   ROLES}
20:  Remove C from CROLES
21: end while
22: Return ROLES

```

mediately stop the iteration and use the best role found so far. This can significantly help in reducing the overall time. Algorithm 1 gives the details.

EXAMPLE 1. We now briefly go through a small example that helps to demonstrate the working of the algorithm. We use the same hypothetical organization described in the introduction in Figure 1, along with set of deployed roles shown in Figure 1(b). We go through the run of the algorithm when run with a weight for similarity of 0.2. In this case, 10 roles are found with a final similarity of 0.45 as computed according to Definition 6 (Algorithm 3). Since it would be quite tedious to show all of the 10 iterations (one role is picked in each iteration), we instead just show a few of the iterations. Figure 3(a) shows the very first iteration where the role $\{p_5, p_7, p_{10}\}$ is chosen as the role with the best score (maximum uncovered area and similarity).

Algorithm 2 Similarity(*C*, *DROLES*)

```

MaxSim  $\leftarrow 0$ 
for each role R  $\in$  DROLES do
  ni  $\leftarrow$   $|C \cap R|$  {number of common permissions}
  nu  $\leftarrow$   $|C \cup R|$  {total number of unique permissions}
  sim  $\leftarrow$   $n_i / n_u$ 
  if sim > MaxSim then
    MaxSim  $\leftarrow$  sim
  end if
end for
Return MaxSim

```

Algorithm 3 Similarity(CROLES, DROLES)

```
MaxSim  $\leftarrow$  0
AvgSim  $\leftarrow$  0
MaxSimSet  $\leftarrow$  0
Counter  $\leftarrow$  0
for each role  $c \in CROLES$  do
  for each role  $d \in DROLES$  do
     $n_i \leftarrow |c \cap d|$  {number of common permissions}
     $n_u \leftarrow |c \cup d|$  {total number of unique permissions}
     $sim \leftarrow n_i/n_u$ 
    if  $sim > MaxSim$  then
       $MaxSim \leftarrow sim$ 
    end if
  end for
   $AvgSim \leftarrow AvgSim + MaxSim$ 
  increase Counter by 1
   $MaxSim \leftarrow 0$ 
end for
 $AvgSim \leftarrow AvgSim/Counter$ 
Return  $AvgSim$ 
```

Algorithm 4 UncoveredArea($C, UPA, ROLES$)

```
 $UC \leftarrow 0$ 
for each user  $u \in assigned\_users(C)$  do
  for each permission  $p \in assigned\_permissions(C)$  do
    Mark each cell (u, p) as uncovered
  end for
end for
for each role  $R \in ROLES$  do
  for each user  $u \in R$  do
    for each permission  $p \in R$  do
      Mark each cell (u, p) of  $R$  as covered
    end for
  end for
end for
Let  $UC$  be the number of cells marked as uncovered
Return  $UC$ 
```

Figure 3(b) shows the fifth iteration when the role $\{p_1\}$ is picked. Finally, Figure 3(c) shows the final iteration when the role $\{p_{11}\}$ is picked and the remaining uncovered area at that point drops to 0 which terminates the algorithm.

4.1 Computational Complexity

The computational complexity of the algorithm depends on two factors: the complexity of the candidate generation phase and the complexity of the candidate selection phase. Since the FastMiner algorithm uses pairwise intersection of unique users to generate candidate roles, it requires $O(n^2)$ time, where n is the number of users. Since at most n roles are necessary to describe the UPA (each user is in a role by itself), at most n iterations are required for candidate selection. Thus in the absolute worst case, the overall cost is $O(n^3)$ which is still significantly better than the exponential worst case of tiling. However, in practice, due to the sorting and quick termination strategy, the algorithm takes an order of magnitude less time.

4.2 Experimental Evaluation

To check the effect of weight on the results, we ran some

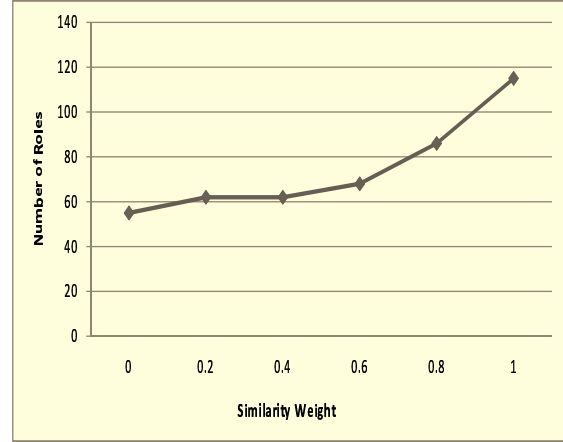


Figure 4: Number of roles vs. weight

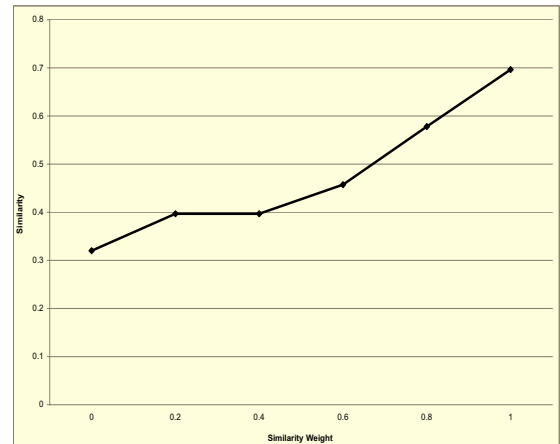


Figure 5: Similarity vs. weight

experiments. Figure 4 shows the number of roles as a function of the weight. Figure 5 shows the similarity of the roles generated to the deployed roles as a function of the weight. In these experiments, the number of users was set to 200 and the number of permissions was set to 400.

5. DISCUSSION

As mentioned earlier, the goal of the Minimal Perturbation RMP is to discover a set of roles that are minimal and are as similar as possible to the set of deployed roles. By assigning weights to both, it enables organizations to set their own priorities on the relative importance of future maintenance vs. the cost of changing over. In other words, by tuning the weight, one can tune the trade-off between the maintenance cost and the change over cost.

However, this does not provide a complete solution to role migration. Essentially, once a set of roles is discovered, how does an organization move from its current set of roles to the new set of roles? In one sense this is trivial. Since the algorithm returns UA as well as PA , in terms of operationalizing the new set of roles, we already know what roles need to be

	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
u0	0	1	1	1	0	0	0	0	1	0	0	0
u1	0	0	1	1	1	0	0	0	0	0	1	0
u2	0	1	0	0	1	1	0	0	1	0	1	0
u3	0	0	0	1	0	1	0	1	1	1	1	0
u4	0	1	0	0	0	0	0	0	0	0	0	0
u5	0	1	0	1	0	0	0	0	1	1	0	1
u6	0	0	0	1	0	1	0	1	1	1	1	1
u7	0	0	0	1	1	1	0	0	1	0	0	0
u8	0	0	0	1	0	0	0	0	1	1	0	0
u9	0	1	0	0	0	0	0	0	0	0	1	0
u10	0	0	0	0	1	1	0	0	1	0	0	0
u11	0	0	0	1	1	0	0	0	0	0	0	0
u12	0	1	0	0	1	1	0	1	1	0	1	0
u13	0	1	0	1	0	1	0	1	1	0	1	1
u14	0	0	1	1	1	0	0	0	1	1	0	0
u15	0	0	0	1	1	0	0	0	0	0	0	0

Uncovered Area = 53

(a) Iteration 1

	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
u0	0	1	1	1	0	0	0	0	1	0	0	0
u1	0	0	1	1	1	1	0	1	0	0	1	0
u2	0	1	0	0	1	1	0	0	1	0	1	0
u3	0	0	0	1	0	1	0	1	1	1	1	0
u4	0	0	0	0	0	1	0	1	0	0	1	0
u5	0	1	0	1	0	0	0	0	1	1	0	1
u6	0	0	0	1	0	1	0	1	1	1	1	1
u7	0	0	0	1	1	1	0	0	1	0	0	0
u8	0	0	0	1	0	0	0	0	1	1	0	0
u9	0	1	0	0	0	0	0	0	0	0	1	0
u10	0	0	0	0	1	1	0	0	1	0	0	0
u11	0	0	0	1	1	0	0	0	0	0	0	0
u12	0	1	0	0	1	1	0	1	1	0	1	0
u13	0	1	0	1	0	1	0	1	1	0	1	1
u14	0	0	1	1	1	0	0	0	1	1	0	0
u15	0	0	0	1	1	0	0	0	0	0	0	0

Uncovered Area = 12

(b) Iteration 5

	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
u0	0	1	1	1	0	0	0	0	0	1	0	0
u1	0	0	1	1	1	1	0	1	0	0	1	0
u2	0	1	0	0	1	1	0	0	1	0	1	0
u3	0	0	0	1	0	1	0	1	1	1	1	0
u4	0	1	0	0	0	1	0	1	0	0	1	0
u5	0	1	0	1	0	0	0	0	1	1	0	1
u6	0	0	0	1	0	1	0	1	1	1	1	1
u7	0	0	0	1	1	1	0	0	1	0	0	0
u8	0	0	0	1	0	0	0	0	1	1	0	0
u9	0	1	0	0	0	0	0	0	0	0	1	0
u10	0	0	0	0	1	1	0	0	1	0	0	0
u11	0	0	0	1	1	0	0	0	0	0	0	0
u12	0	1	0	0	1	1	0	1	1	0	1	0
u13	0	1	0	1	0	1	0	1	1	0	1	1
u14	0	0	1	1	1	0	0	0	1	1	0	0
u15	0	0	0	1	1	0	0	0	0	0	0	0

Uncovered Area = 0

(c) Iteration 10

Figure 3: Minimal Perturbation RMP Iterations

assigned to each user, etc. However, this ignores the fact that RBAC is composed of much more than roles. Since we limit our attention to $RBAC_0$ [5] in this paper, we do not consider role hierarchies and separation of duty constraints. It is unclear what such constraints and hierarchies will be in the new set of roles. This represents a large investment - such information is critical for ease of deployment. Ideally, what we would like is a small set of similar roles, *along* with a mapping from the old roles to the new roles. In addition, it is not clear what are the effects of role migration on the existing separation of duty constraints. Therefore, one must consider this aspect into account while minimizing the perturbation.

6. RELATED WORK

A number of approaches have been proposed in the literature to accomplish the task of role engineering, which can be categorized into three approaches: top-down, bottom-up, and hybrid. While the top-down approach defines roles by examining the business processes, the bottom-up approach typically aggregates existing permissions to come up with roles.

Coyne [2] is the first to describe the role engineering problem, and to present the concepts of the top-down approach. A top-down approach to determine the needed permissions of roles using use cases has been proposed by Fernandez and Hawkins [4]. A GUI interface has been developed by Brooks [1] to migrating to a role-based environment. Later, Roeckle et al. [11] present a process-oriented approach, which analyzes business processes to deduce roles and access permissions on systems are assigned to the roles. Shin et al. [16] present a system-centric approach that examines backward and forward information flows and employs UML to conduct a top-down engineering of roles. Thomsen et al. [17] propose a bottom-up approach, which derives permissions from objects and their methods and then derive roles derived from these permissions. Neumann and Strembeck [10] consider usage scenarios as a semantic unit for deriving permissions, which are then aggregated into roles. Epstein and Sandhu [3] propose to use UML for facilitating role engineering, where

roles can be defined in either a top-down or a bottom-up manner. Kern et al. [7], propose a life-cycle approach, an iterative-incremental process, that considers different stages of the role life-cycle including role analysis, role design, role management, and role maintenance.

Kuhlmann, Shohat, and Schimpf [8] present another bottom-up approach, which employs a clustering technique similar to the k-means clustering. As such, it is required to first pre-define the number of clusters. In [14], Schlegelmilch and Steffens propose an agglomerative clustering based approach to role mining (called ORCA), which discovers roles by merging permissions appropriately. However, in ORCA, the order in which permissions are merged determines the outcome of roles. Moreover, it does not allow overlapping roles (i.e., a user cannot play multiple roles), which is a significant drawback. More recently, Vaidya et al. [19] propose an approach based on subset enumeration, called RoleMiner, which eliminates the above limitations.

An inherent problem with all of the above approaches is that there is no formal notion of *goodness/interestingness* of a role. All of the algorithms above present heuristic ways to find a set of candidate roles. While offering justifications for the identified roles, there is no integrative view of the entire set of roles. For insightful bottom-up analysis, we need to define interestingness metrics for roles. [19] takes a first step towards this by ordering candidate roles on the basis of their support (i.e., roles that are more prevalent are ordered higher). However, this metric still is quite ad-hoc and preliminary. Also, while one may come up with interestingness metrics for a role by itself, this does not directly lead to the notion of a good collection of roles. Indeed, there is no formal definition of what is a good *collection of roles*. Defining this is critical for the security administrator to gain confidence and be able to fully utilize the output of any role mining algorithm beyond a piece-meal fashion.

Recently, [18] has formally defined the role mining problem, and has analyzed its theoretical bounds. Assuming that one can represent the user permissions as a binary matrix, informally, [18] has defined the *basic* role mining problem (basic-RMP). It has proposed several variants of the basic-RMP, including the min-noise RMP, δ -approximate RMP

and edge-RMP, that map to the different decomposition criteria mentioned above. Essentially, the solution to basic RMP gives optimal set of roles that characterize the existing permissions of users. While δ -approximate RMP allows a limited amount of inexactness, which may result in far less number of roles by capturing almost all the existing permissions, min-noise RMP allows the security administrators to specify the number of roles yet derive the best possible set of roles. All these variants have significance in solving the role engineering problem and help the security administrators to pick and choose the one that perfectly suits to the organizational needs. [9] presents a unified framework for modeling the basic RMP and its variants using binary integer programming. Such modeling helps in directly adopting the huge body of heuristic solutions and tools developed for binary integer programming.

However, these solutions are entirely bottom-up approaches to role engineering. On the other hand, the minimum perturbation RMP proposed in this paper is a role engineering approach that provides a means to formally combine the bottom-up and top-down approaches. It also helps organizations to adopt a set of roles that are close to the optimal set of roles with minimum disruption to the existing set of roles.

Our approach for minimal perturbation RMP relies on computing the similarity or distance between two roles as well as between two sets of roles using Jaccard Coefficient. In [15], four types of relationships between two roles, namely, contain, overlap, equivalent and not related, have been identified. These relationships can be determined by examining the permissions contained in role and the semantic equivalences of these permissions. Our approach to computing similarity can be enhanced by considering such semantics among permissions.

7. CONCLUSIONS

In this paper, we have formally defined the problem of migrating closer to the optimal set of roles from the currently deployed roles with as little disruptions as possible. We denote this as the minimal perturbation RMP. Assume the currently deployed roles are devised using a top-down approach and the optimal set of roles using the bottom-up approach. The minimal perturbation RMP provides a formal means to combine the roles derived from the bottom-up and top-down approaches. Moreover, this approach helps organizations to migrate to a new set of optimal roles without having to disrupt much of their processes. Additionally, this provides a formal means of merging their respective roles when two or more organizations merge. This relies on examining the similarity between pairs of roles and pairs of sets of roles. We also provide a means to add weight that specifies how close one wants to the optimal set versus the deployed set of roles.

Since this is an NP-hard problem, we provide a heuristic algorithm that has $O(n^3)$ complexity. However, due to the sophisticated pruning strategy employed in our implementation, for many practical data sets, it takes significantly less time. Our experimental results indicate that the minimal perturbation RMP is a good way of balancing the deployed roles versus the optimal roles.

It is important to note that, in many cases, while migrating to a new set of roles, one should be able to specify

the desired set of unchanged parameters. These could include certain user-assignments, certain roles or certain role-permission assignments. We plan to include a different weight metric to emphasize certain parameters to be unchanged while migrating. Additionally, our minimal perturbation RMP only finds the new set of roles, but does not provide a mapping between exiting set of roles to the new set. This would involve, in addition to discovering the similarity/distance, finding out containment amongst roles. Additionally, there could be some currently specified separation of duty constraints. So the role migration process should be as less disruptive as possible with respect to these constraints.

Acknowledgements

Initial discussions with Dr. Sachin Lodha at the Tata Research Design and Development Centre led to the formulation and formalization of this problem. We would like to sincerely thank him for suggesting this line of work. We also acknowledge the helpful comments by the anonymous referees which helped in improving this paper.

8. REFERENCES

- [1] K. Brooks. Migrating to role-based access control. In *ACM Workshop on Role-Based Access Control*, pages 71–81, 1999.
- [2] E.J.Coyne. Role-engineering. In *1st ACM Workshop on Role-Based Access Control*, 1995.
- [3] P. Epstein and R. Sandhu. Engineering of role/permission assignment. In *17th Annual Computer Security Application Conference*, December 2001.
- [4] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. In *ACM Workshop on Role-Based Access Control*, pages 121–125, 1997.
- [5] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *TISSEC*, 2001.
- [6] M. P. Gallagher, A. O'Connor, and B. Kropp. The economic impact of role-based access control. *Planning report 02-1, National Institute of Standards and Technology*, March 2002.
- [7] A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett. Observations on the role life-cycle in the context of enterprise security management. In *7th ACM Symposium on Access Control Models and Technologies*, June 2002.
- [8] M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *Symposium on Access Control Models and Technologies (SACMAT)*. ACM, June 2003.
- [9] H. Lu, J. Vaidya, and V. Atluri. Optimal boolean matrix decomposition: Application to role engineering. In *IEEE International Conference on Data Engineering*, to appear, April 2008.
- [10] G. Neumann and M. Strembeck. A scenario-driven role engineering process for functional rbac roles. In *7th ACM Symposium on Access Control Models and Technologies*, June 2002.
- [11] 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 ACM, editor, *RBAC*, 2000.
- [12] R. S. Sandhu et al. Role-based Access Control Models. *IEEE Computer*, pages 38–47, February 1996.
- [13] A. Schaad, J. Moffett, and J. Jacob. The role-based access control system of a european bank: A case study and discussion. In *Proceedings of ACM Symposium on Access Control Models and Technologies*, pages 3–9, May 2001.
- [14] J. Schlegelmilch and U. Steffens. Role mining with orca. In *Symposium on Access Control Models and Technologies (SACMAT)*. ACM, June 2005.
- [15] B. Shafiq, J. B. Joshi, E. Bertino, and A. Ghafoor. Secure interoperation in a multidomain environment employing rbac policies. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1557–1577, 2005.
- [16] D. Shin, G.-J. Ahn, S. Cho, and S. Jin. On modeling system-centric information for roleengineering. In *8th ACM Symposium on Access Control Models and Technologies*, June 2003.
- [17] D. Thomsen, D. O’Brien, and J. Bogle. Role based access control framework for network enterprises. In *14th Annual Computer Security Application Conference*, pages 50–58, December 1998.
- [18] J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: Finding a minimal descriptive set of roles. In *The Twelfth ACM Symposium on Access Control Models and Technologies*, pages 175–184, Sophia Antipolis, France, June20-22 2007.
- [19] 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*, pages 144–153, 2006.