Evolving Role Definitions Through Permission Invocation Patterns

Wen Zhang
EECS Dept.
Vanderbilt University
Nashville, TN, USA
wen.zhang.1@vanderbilt.edu

You Chen
Biomedical Informatics Dept.
Vanderbilt University
Nashville, TN, USA
you.chen@vanderbilt.edu

Carl A. Gunter
Dept. of Computer Science
University of Illinois
Urbana, IL, USA
cgunter@illinois.edu

David Liebovitz
Dept. of Medicine
Northwestern University
Chicago, IL, USA
davidl@northwestern.edu

Bradley Malin
Biomedical Informatics Dept.
Vanderbilt University
Nashville, TN, USA
b.malin@vanderbilt.edu

ABSTRACT

In role-based access control (RBAC), roles are traditionally defined as sets of permissions. Roles specified by administrators may be inaccurate, however, such that data mining methods have been proposed to learn roles from actual permission utilization. These methods minimize variation from an information theoretic perspective, but they neglect the expert knowledge of administrators. In this paper, we propose a strategy to enable a controlled evolution of RBAC based on utilization. To accomplish this goal, we extend a subset enumeration framework to search candidate roles for an RBAC model that addresses an objective function which balances administrator beliefs and permission utilization. The rate of role evolution is controlled by an administrator-specified parameter.

To assess effectiveness, we perform an empirical analysis using simulations, as well as a real world dataset from an electronic medical record system (EMR) in use at a large academic medical center (over 8000 users, 140 roles, and 140 permissions). We compare the results with several state-of-the-art role mining algorithms using 1) an outlier detection method on the new roles to evaluate the homogeneity of their behavior and 2) a set-based similarity measure between the original and new roles. The results illustrate our method is comparable to the state-of-the-art, but allows for a range of RBAC models which tradeoff user behavior and administrator expectations. For instance, in the EMR dataset, we find the resulting RBAC model contains 22% outliers and a distance of 0.02 to the original RBAC model when the system is biased toward administrator belief, and 13% outliers and a distance of 0.26 to the original RBAC model when biased toward permission utilization.

SACMAT'13, June 12–14, 2013, Amsterdam, The Netherlands.

Categories and Subject Descriptors

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

General Terms

Security, Algorithms

Keywords

Audit logs, role-based access control, role mining

1. INTRODUCTION

Role-based access control (RBAC) is a framework that has been adopted widely for managing the rights of users in information systems [16]. It was designed to simplify the allocation of access rights by mapping users to a set of roles, each of which is associated with a set of permissions. Beyond the specification of rights, RBAC can be employed for intrusion detection purposes [1, 25]. For instance, the actions of a user can be compared to those of the users associated with the same role to determine if deviation from expected behavior has transpired. Thus, roles which consist of users with relatively similar behavior are desirable for detecting and preventing insider threats [14].

The process of defining roles, which is often referred to as role engineering [5], is a notoriously challenging problem. In general, role engineering approaches have fallen into two camps: i) *top-down* and ii) *bottom-up*. In the top-down setting, organizational experts (or system administrators) model the workflows associated with an enterprise, which are subsequently decomposed into tasks and roles [17, 20]. Bottom-up approaches (e.g., [18, 23]), on the other hand, discover roles by leveraging information that already exists in the system. Many of these approaches (e.g., [18, 22, 23, 26]) propose roles based on patterns in existing user-permission assignments.

There are benefits and drawbacks to each camp. Top-down approaches, for instance, are based on expert reasoning, in-depth interviews, and tend to reflect organizational expectations [21]. However, these approaches often result in

high costs to an enterprise [17] because they require a substantial amount of time to document the workflows which exist. They may also be subject to the problem of informant inaccuracy [7] and, thus, access control models which are incomplete or contain errors [12]. By contrast, bottom-up approaches enable an RBAC system to be derived automatically, such that their cost is significantly lower than their top-down counterparts. Yet, there is no guarantee that users in the same role, as defined by their permissions, will exhibit similar behavior.

Historically, role engineering strategies have treated these camps independently, but we believe there is merit in combining them into a more comprehensive role engineering framework. Consider, while it may be that expert-specified RBAC configurations are not entirely representative of an enterprise, it is unlikely that such information is completely uninformed. As such, the goal of this paper is to propose a role engineering approach that evolves roles in a manner that balances 1) the desire to retain an existing RBAC configuration with 2) the need to assign users with similar behavior into common roles.

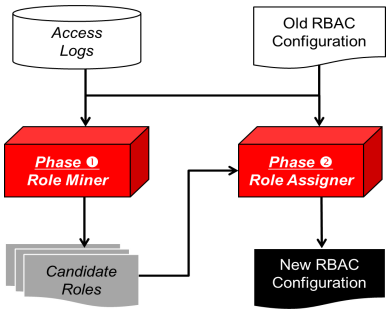


Figure 1: An architectural overview of DDRE algorithm

From a high-level, our evolution strategy, which we call the Data Driven Role Evolution (DDRE) algorithm, consists of mainly two phases as shown in Figure 1. In the first phase, we mine a set of candidate roles, which are selected to optimize an objective function that balances distance from the original roles with behaviorial similarity in the form of permission invocation in access logs. In the second phase, each user is assigned to roles according to a criterion that mitigates redundancy in the access control model. There are several primary contributions of this paper, including:

- **A new objective function for the role mining problem.** We devise an objective that balances the administrator's belief with the evidence in existing access logs. The function is parameterized, such that a user can bias the resulting RBAC configurations toward belief or evidence as deemed desirable.
- **A hybrid role engineering algorithm.** We propose a new role engineering algorithm that builds on a subset enumeration technique employed in previous role engineering strategies. Our algorithm evolves existing RBAC configurations into new configurations which are are more effective at addressing administrators' beliefs and permission utilization goals than current role engineering strategies.
- **A multi-objective empirical evaluation.** To evaluate the resulting RBAC configurations, we compare our algorithm with state-of-the-art role mining techniques using a real dataset derived from a large electronic medical record system, as well as a controlled synthetic dataset. The results show that our role evolution algorithm can, unlike previous methods, produce a range of RBAC configurations in comparison to previous methods. Moreover, we show the resulting configurations follow the expected bias of the algorithm and indicate utilization patterns exist in the real dataset.

The remainder of this paper is organized as follows. Section 2 reviews the foundations upon which our method is built, including the role mining problem, access logs, distance measures, and outlier detection methods. Section 3 then describes our role evolution algorithm. Section 4 presents the experiments performed to evaluate our approach. Section 5 provides a survey of related work in role engineering and role mining. Finally, Section 6 concludes the paper and suggests next steps for extending this work.

2. PRELIMINARIES

In this section, we review several topics that inform the development of our role revision method. This section begins with a formalization of a generalized version of the role mining problem. Next, we provide a description of access logs as they are utilized in our method. Then, we introduce a formalization of the objective function invoked in our variation of the role mining problem. Finally, this section concludes with a description of one-class support vector machines, an effective outlier detection algorithm, which we employ as a measure of the quality of RBAC configurations.

2.1 Generalized Role Mining Problem

We begin with a generalized perspective of the role mining problem, which will be refined to model the problems studied in this work.

Definition: [*Generalized Role Mining Problem*] Let $t = \langle U, P, UPA \rangle$ denote an access control configuration, where $U = \{u_1, u_2, \ldots, u_m\}$ is a set of users, $P = \{p_1, p_2, \ldots, p_n\}$ is a set of permissions, and UPA is an $m \times n$ Boolean matrix indicating the mapping between U and P.

The goal of the *Generalized Role Mining Problem* is to find an RBAC configuration $c = \langle U, P, R, URA, RPA \rangle$[1], subject to $UPA = URA \otimes RPA$, such that an objective function $f()$ is optimized. In the configuration, $R = \{r_1, r_2, \ldots, r_k\}$ is a set of roles, URA is an $m \times k$ Boolean matrix indicating the mapping between U and R, and RPA is a $k \times n$ Boolean matrix indicating the mapping between R and P.[2] □

The matrices in Figures 2(b) and (c) depict an example of an RBAC configuration. It can be seen there are six users, seven permissions, and two roles.

Given a role r_l, we can readily extract the corresponding users and associated permissions. In this paper, we use $\mathbb{P}_l^{(c)} = \{p_x \mid RPA_{lx} = 1\}$ to denote the set of permissions assigned to r_l, and $\mathbb{U}_l^{(c)} = \{u_y \mid URA_{yl} = 1\}$ to denote the set

[1] In this paper, we only consider the $RBAC_0$ model (i.e., we do not consider role hierarchies or constraints).

[2] $x = a \otimes b$ denotes the Boolean matrix product, in which an element is defined as $x_{ij} = \vee_k (a_{ik} \wedge b_{kj})$.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
u_1	116	485	151	402	249		
u_2	181	797	21	58	33		
u_3	199	819	77	196	91		
u_4			29	81	44	402	174
u_5			108	278	161	530	215
u_6			25	62	31	334	118

(a) *UPIM*

	r_1	r_2
u_1	1	0
u_2	1	0
u_3	1	0
u_4	0	1
u_5	0	1
u_6	0	1

(b) *URA*

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
r_1	1	1	1	1	1	0	0
r_2	0	0	1	1	1	1	1

(c) *RPA*

Figure 2: An example of a user-permission invocation matrix (*UPIM*) and an RBAC configuration (*URA* and *RPA*).

of users under r_l in the RBAC configuration c. When appropriate, we adopt the standard convention of representing a role as its corresponding set of permissions. For example, $\gamma = \{p_1, p_2\}$ represents a role possessing two permissions. In general, all users whose permission set is the superset of γ automatically obtain this role. There are, however, exceptions to this role that will be introduced in Section 3.

Various objective functions have been proposed for the role mining problem. Certain functions are based on the size of R [23], while others use variations of structural complexity [26]. With regard to the latter, objective functions have been based on the size of R and the total number of elements in *URA* and/or *RPA*. In this paper, we define the objective function from the perspective of i) user behavior similarity and ii) distance to the initial RBAC configuration.

2.2 Access Log

In this work, an access log is represented as an $m \times n$ user-permission invocation matrix *UPIM*. We use ω_{ij} to denote the number of times user u_i invoked permission p_j. Figure 2(a) depicts the *UPIM* that corresponds to the RBAC configuration in Figures 2(b) and (c). To mitigate bias which may occur from working with the raw frequency counts, we preprocess *UPIM* through a row-wise normalization (i.e., all numbers are divided by their rowsum) to represent *UPIM* as a set of user-specific probability distributions.

To measure the homogeneity of a role, we need to extract the corresponding access records from *UPIM*. This is accomplished through the application of a projection matrix.

Definition: [*Projection Matrix*] Given an RBAC configuration c and user-permission invocation matrix *UPIM*, the *projection matrix* M_{r_l} for role r_l is an $p \times q$ matrix, where $p = |\mathbb{U}_l^{(c)}|$ and $q = |\mathbb{P}_l^{(c)}|$. Each row (column) of M_{r_l} represents a user (permission) associated with r_l. Let β_{ij} be defined as an element of M_{r_l} as follows. If the i^{th} user in $\mathbb{U}_l^{(c)}$ is u_f in U, and the j^{th} permission in $\mathbb{P}_l^{(c)}$ is p_g in P, then $\beta_{ij} = \omega_{fg}$.□

2.3 Objective Function

To balance existing beliefs in roles with actual user behavior, we propose a new objective function for the role mining problem, which is based on two goals. The first goal is to enable each role to possess high homogeneity in the rate at which permissions are accessed. The second goal is to ensure the new and pre-existing RBAC are "near" one another. We use functions $h()$ and $j()$ to measure the first and second goal, respectively, and define the objective function as:

$$f(c_{new}) = \alpha \cdot h(c_{new}) + (1 - \alpha) \cdot j(c_{old}, c_{new}) \qquad (1)$$

where c_{new} is the RBAC configuration proposed by a role mining algorithm, c_{old} is the existing RBAC configuration, and α is a real value between 0 and 1 to bias the system from $h()$ to $j()$. The following subsections provide details regarding how the functions $h()$ and $j()$ are computed.

2.3.1 RBAC homogeneity

In this section, we formally introduce the notion of homogeneity, which will be applied to characterize the similarity of the users in a role.

Definition: [*Homogeneity*] Given an RBAC configuration $c = \langle U, P, R, URA, RPA \rangle$ and a user-permission matrix *UPIM*, the *role homogeneity* of r_l is:

$$\mathsf{ho}(r_l) = m^{-1} \sum_{i=1}^{m} (1 - \mathsf{cosine}(\mathbf{x}_i, \mathbf{c}_l)), \qquad (2)$$

where m is the number of row vectors in M_{r_l}, $\mathbf{x}_i$ is the i^{th} row vector of M_{r_l}, $\mathbf{c}_l$ is the mean vector of all row vectors in M_{r_l}, and $\mathsf{cosine}(\mathbf{a}, \mathbf{b})$ is the cosine similarity $\frac{\mathbf{a} \bullet \mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$.

The *RBAC homogeneity* of c is then defined as:

$$h(c) = |R|^{-1} \sum_{r_l \in R} \mathsf{ho}(r_l) \qquad (3)$$

□

Role and RBAC homogeneity (Equations 2 and 3) have a natural geometric interpretation. Consider, if a role consists of a set of highly similar users, then the vectors representing the behaviors of these users will form a relatively compact cluster in $\mathbb{R}^k$ (where k is the dimensionality of the vectors) and the degree of the angle between each vector and the mean of the cluster, measured by $1 - \mathsf{cosine}(\mathbf{x}_i, \mathbf{c}_l)$, will tend to be small. Conversely, if the users in a role exhibit highly diverse behavior, then the cluster will tend to have a long diameter and the degree of the angle will be large.

2.3.2 Distance Between RBAC Configurations

In order to measure how far a new RBAC configuration has migrated from the initial configuration, we introduce a set-based similarity measure. First, we define the distance between two roles.

Definition: [*Role Distance*] Let γ_i and δ_j be roles in RBAC configurations c_1 and c_2, respectively. The *role distance* between the roles is defined as:

$$\mathsf{jac}(\gamma_i, \delta_j) = 1 - \frac{|(\mathbb{P}_i^{(c_1)} \times \mathbb{U}_i^{(c_1)}) \cap (\mathbb{P}_j^{(c_2)} \times \mathbb{U}_j^{(c_2)})|}{|(\mathbb{P}_i^{(c_1)} \times \mathbb{U}_i^{(c_1)}) \cup (\mathbb{P}_j^{(c_2)} \times \mathbb{U}_j^{(c_2)})|} \qquad (4)$$

where $A \times B$ is the Cartesian product of sets A and B. □

In our setting, a role corresponds to the Cartesian product of its associated set of permissions and set of users. This enables the comparison of two roles to be performed in the joint space of permissions and users. Thus, our definition corresponds to the Jaccard distance, a widely used measure for the comparison of two sets [9].

We leverage the distance between roles to define the distance between a role and a role set.

Definition: [*Role Set Distance*] The *role set distance* from role γ to role set R is the minimum distance to any role in the set:

$$\mathsf{minjac}(\gamma, R) = \min_{\delta \in R} \mathsf{jac}(\gamma, \delta) \qquad (5)$$

□

Finally, we can define the distance from one RBAC configuration to another.

Definition: [*RBAC Distance*] Let c_i and c_j be RBAC configurations. The *RBAC distance* from c_i to c_j is:

$$j(c_i, c_j) = |R_i|^{-1} \sum_{\gamma \in R_i} \mathsf{minjac}(\gamma, R_j) \tag{6}$$

where R_i and R_j are the role sets of c_i and c_j, respectively. □

2.3.3 Quality of a Role

We further use the metrics above to define a heuristic function that computes a score for a role γ. This function, which we call the *role score* rs, is defined as:

$$\mathsf{rs}(\gamma) = \alpha \cdot \mathsf{ho}(\gamma) + (1 - \alpha) \cdot \mathsf{minjac}(\gamma, R), \tag{7}$$

where α is as defined in Equation 1 and R is the role set of c_{old} in Equation 1. This function will be leveraged to guide our role evolution algorithm (described in Section 3).

2.4 One-Class SVM

To evaluate the homogeneity of the resulting roles, we employ an outlier detection algorithm. The selection of this strategy is based on the hypothesis that the more homogeneous a role is, the smaller the number of outlying users it will contain. In this paper, we use a one-class support vector machine (SVM) [19] to detect outlying users for each role. SVMs have been reported as comparable, and often superior, to other anomaly detection methods in various settings [11], including intrusion detection [6].

One-class SVMs can be applied to learn a region that contains only the training set, which is expected to be typical data for a class. Any data point in a test set that falls out of the region will be predicted as an anomaly. Theoretically, the goal of SVM in this scenario is to find a hyperplane $\mathbf{w} \in F$ that separates the training set from the origin with the maximum margin. This can be formalized as an optimization problem as follows:

$$\begin{aligned} &\min_{\mathbf{w} \in F, \xi \in \mathbb{R}^l, \rho \in \mathbb{R}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{v \cdot l} \sum_i \xi_i - \rho \\ &\textit{subject to } (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i, \xi_i \geq 0 \end{aligned}, \tag{8}$$

where ξ_i are non-zero slack variables to be penalized in the objective function. When values for $\mathbf{w}$ and ρ can be found which solve the optimization function, the majority of the training set satisfies $\mathsf{sgn}(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) \geq \rho$, while the regularization term $\|\mathbf{w}\|$ remains small. The parameter v determines the tradeoff between these two goals. With $\mathbf{w}$ and ρ, we have a decision function $f(\mathbf{x}) = \mathsf{sgn}(\mathbf{w} \cdot \Phi(\mathbf{x}) - \rho)$ to determine if an new instance $\mathbf{x}$ is anomalous.

In this work, we specifically use one-class SVMs with an RBF kernel, as defined in Equation 9:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathsf{exp}(-g \cdot \|\mathbf{x}_i - \mathbf{x}_j\|^2) \tag{9}$$

The parameters g in Equation 9 and v in Equation 8 are key factors that influence the performance of one-class SVMs. We utilize a grid search technique to find values for g and v that enable a robust SVM [10].

For evaluation, for each role r_l, we split the row vectors of a projection matrix into a training set and a test set. We perform gird search on the training set to obtain g and v, which are then applied to train and test a one-class SVM. The proportion of outlying users identified by the one-class SVM is applied to measure the homogeneity of the role.

3. ROLE EVOLUTION BY PERMISSION UTILIZATION

This section begins by formally defining the *Role Evolution By Permission Utilization* (REPU) problem.

Definition: [*REPU Problem*] Given an existing RBAC configuration $c = \langle U, P, R, URA, RPA \rangle$, a user-permission assignment UPA ($UPA = URA \otimes RPA$) and a user-permission invocation matrix $UPIM$, *REPU* is to find a new RBAC configuration $c^* = \langle U, P, R^*, URA^*, RPA^* \rangle$ subject to $UPA = URA^* \otimes RPA^*$, such that the objective function $f(c^*) = \alpha \cdot h(c^*) + (1 - \alpha) \cdot j(c^*, c)$ is minimized. □

The role mining problem has been shown to be NP-complete [23]. The REPU problem is a variation on role mining and can be reduced to this problem, making it NP-complete as well. Thus, we propose a heuristic-based search strategy based on a two-phase process as defined below.

3.1 Algorithm Description

To address the REPU problem, we designed the *Data-Driven Role Evolution* (DDRE) algorithm. Here we provide a walkthrough of the process and refer the reader to Algorithm 1 for specific details. The two phases of the algorithm are: i) candidate role generation and ii) role assignment.

3.1.1 Candidate Role Generation

The first phase begins with a set of *unit roles* UR, such that there is one permission per role and no roles have the same permission (i.e., a one-to-one mapping of permissions and unit roles). Next, UR is copied to a candidate role set CR. The algorithm then iterates until a termination condition is satisfied. Each iteration begins by instantiating a set of new roles into an empty pool DP, which is based on a pairwise union for all roles in CR. For example, the union of $\{p_x\}$ and $\{p_y\}$ yields $\{p_x, p_y\}$.

The roles in DP are sorted by their quality scores (defined in Section 2.3.3), such that DP serves as a priority queue, where the best role is at the top. The algorithm then proceeds through the queue, by moving a role from the top of DP to CR and flipping the 1's in the user-permissions assignment UPA_{temp} that are covered by the role to 0's. This process continues until every element in UPA_{temp} is set to 0. The algorithm then reiterates.

3.1.2 Role Assignment

Once CR is stable or the max number of iterations is reached (i.e., the termination condition), the algorithm enters the second phase of role assignment, the details of which are in Algorithm 3. The goal of this phase is to ensure that each user is assigned to non-redundant roles. By default, any user whose permission set is a superset of one role will automatically obtain this role. This would result in redundancy and an increasing unnecessary complexity in the system. For example, consider a set of users assigned to roles $\gamma = \{p_1, p_2, p_3\}$ and $\mu = \{p_1, p_2\}$. The latter role μ is redundant because the affiliated users can accomplish their task using only the γ role. Thus μ could be removed for the sake of succinctness.

Algorithm 1 Data-Driven Role Evolution

Input: $c = \langle U, P, R, URA, RPA\rangle$, $UPIM$, α, $maxTimes$
Output: $c^* = \langle U, P, R^*, URA^*, RPA^*\rangle$

$t \leftarrow 0, DP \leftarrow \varnothing, CR \leftarrow \varnothing, CR_{old} \leftarrow \varnothing, UR \leftarrow \varnothing, UPA = URA \otimes RPA, UPA_{temp} = UPA$
for each $p_i \in P$ **do**
  $UR \leftarrow UR \cup \{\{p_i\}\}$
end for
$CR \leftarrow UR$
while $|CR_{old} - CR| > 0$ && $t++ \leq maxTimes$ **do**
  $DP \leftarrow \varnothing$
  for each $\mu_i \in CR$ **do**
    for each $\mu_j \in CR$ **do**
      $DP \leftarrow DP \cup \{\mu_i \cup \mu_j\}$
    end for
  end for
  $CR_{old} \leftarrow CR, CR \leftarrow \varnothing$
  Sort(DP, $UPIM$, c, α){Sort roles in DP according to their quality score. See Algorithm 2 for details.}
  for each $\gamma_i \in DP$ **do**
    if every element in UPM_{temp} is 0 **then**
      break
    end if
    if γ_i cannot cover any 1's in UPA_{temp} **then**
      continue
    end if
    $CR \leftarrow CR \cup \gamma_i$
    change all 1's in UPA_{temp} covered by γ_i to 0's
  end for
end while
$\{R^*, URA^*\} = RoleAssignment(U, P, UPA, CR)$
Initialize a $p \times n$ Boolean matrix RPA^* with all elements equal to zero, where $p = |R^*|$ and $n = |P|$.
for each $\mu'_i \in R^*$ **do**
  for each $p_j \in P$ **do**
    if $p_j \in \mu'_i$ **then**
      $RPA^*_{ij} = 1$
    end if
  end for
end for
return $c^* = \langle U, P, R^*, URA^*, RPA^*\rangle$

Algorithm 2 Sort()

Input: DP, $UPIM$, α, $c = \langle U, P, R, URA, RPA\rangle$
Initialize an array of real value, $score[]$, which has the same size as DP
for each $\gamma_i \in DP$ **do**
  $score[i] = rs(\gamma_i)$
end for
Sort DP in ascending order according to $score[]$
return

The problem of winnowing the system down to a minimal set of roles for each user is similar to the set cover problem: given a user who possesses a set of permissions $PMS_i = \{p_{i_1}, p_{i_2}, \ldots, p_{i_k}\}$ and a set of roles $ROLES_i = \{\gamma_1, \gamma_2, \ldots, \gamma_l\}$ whose elements are all subsets of PMS_i, identify the smallest number of roles in $ROLES_i$ whose union equals PMS_i. It has already been shown that this problem is NP-complete [8]. Given the complexity of the problem, we adopt an approximation algorithm [3] to resolve the problem, as shown in Algorithm 3. At each iteration, we select the role in $ROLES_i$ with the largest number of permissions in common with PMS_i. The role is added to R_i and the permissions which were in common with PMS_i are removed from further consideration. This procedure repeats until no elements exist in PMS_i.

Algorithm 3 RoleAssignment()

Input: U, P, UPA, CR
Output: URA, R

$R \leftarrow \varnothing$, $m = |U|$
for each $u_i \in U$ **do**
  $PMS_i \leftarrow \{p_j | \forall p_j \in P, UPA_{ij} = 1\}, ROLES_i \leftarrow \varnothing$
  while $PMS_i \neq \varnothing$ **do**
    Select role μ_k from CR, such that $\mu_k \subseteq PMS_i$ and $|PMS_i \cap \mu_k|$ is maximized.
    $PMS_i \leftarrow PMS_i - \mu_k, ROLES_i \leftarrow ROLES_i \cup \{\mu_k\}$
  end while
  $R \leftarrow R \cup ROLES_i$
end for
Re-index the roles in R using integers 1 to $h = |R|$, such that $R = \{\mu'_1, \mu'_2, \ldots, \mu'_h\}$
Construct $m \times h$ Boolean matrix URA, such that if $\mu'_j \in ROLES_i$, $URA_{ij} = 1$, otherwise $URA_{ij} = 0$
return R, URA

Finally, roles in R_i are assigned to user i. After assigning roles for each user, we obtain a final role set R^* and a user-role assignment URA^*. Specifically, we generate a role-permission assignment RPA^* from the permission sets of the roles. Thus, a new RBAC configuration $c^* = \langle U, P, R^*, URA^*, RPA^*\rangle$ is returned.

3.2 An example

In this section, we use the RBAC configuration and $UPIM$ in Figure 2 with $\alpha = 1$ to illustrate how the DDRE algorithm works in detail.[3]

UPA and RPA indicate there are two roles and six users. The roles are represented by permission sets $\{p_1, p_2, p_3, p_4, p_5\}$ and $\{p_3, p_4, p_5, p_6, p_7\}$. For this example, we create a set of ideal roles as the optimal solution, from which we design a series of generative models to construct $UPIM$. This set contains three roles, which correspond to $\{p_1, p_2\}$, $\{p_3, p_4, p_5\}$ and $\{p_6, p_7\}$. The generative model for each of the roles follows a fixed distribution, which for this example is set to $\{0.2, 0.8\}$, $\{0.2, 0.5, 0.3\}$, and $\{0.7, 0.3\}$, respectively. This means, for instance, that for an arbitrary user u_k associated with the first role, $UPIM_{k1}$:$UPIM_{k2}$ is 1:4.

First, the algorithm initializes the system with a set of unit-roles: $\{\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_5\}, \{p_6\}, \{p_7\}\}$. Next, the algorithm performs a pairwise combination of the unit-roles to derive a pool DP of the form $\{\{p_1, p_2\}, \{p_1, p_3\}, \ldots, \{p_6, p_7\}\}$. From this pool, four roles, $\{p_1, p_2\}$, $\{p_3, p_4\}$, $\{p_6, p_7\}$, and $\{p_4, p_5\}$, are selected for the next round of pairwise combination because they comprise the top four positions of the pool and are able to recover the UPA. When this set of roles is combined, it updates the pool to become $\{\{p_1, p_2\}, \{p_3, p_4\}, \{p_4, p_5\}, \{p_6, p_7\}, \{p_1, p_2, p_3, p_4\}, \{p_1, p_2, p_4, p_5\}, \{p_3, p_4, p_5\}, \{p_4, p_5, p_6, p_7\}\}$.

[3] $\alpha = 1$ implies the algorithm is completely biased to generate a set of roles with high homogeneity in user behavior (i.e., it ignores the structure of the original roles).

At this point, we select another four roles, $\{p_1, p_2\}$, $\{p_3, p_4\}$, $\{p_6, p_7\}$ and $\{p_3, p_4, p_5\}$, from the pool because they comprise the top four positions in the pool and are able to recover the *UPA*. Again, these roles are combined to update the pool to become $\{\{p_1, p_2\}, \{p_3, p_4\}, \{p_3, p_4, p_5\}, \{p_6, p_7\}, \{p_1, p_2, p_3, p_4\}, \{p_1, p_2, p_3, p_4, p_5\}, \{p_3, p_4, p_6, p_7\}, \{p_3, p_4, p_5, p_6, p_7\}\}$. At this point, roles in the top four positions of the pool, $\{p_1, p_2\}$, $\{p_3, p_4\}$, $\{p_6, p_7\}$ and $\{p_3, p_4, p_5\}$, are selected to constitute the candidate role set. Since the candidate role set is the same as the previous round, this phase of the DDRE algorithm terminates and returns this candidate role set.

Next, the roles $\{p_3, p_4\}$ are redundant in the presence of $\{p_3, p_4, p_5\}$, so they are discarded in the second phase.

Finally, the remaining three roles $\{p_1, p_2\}$, $\{p_3, p_4, p_5\}$, and $\{p_6, p_7\}$ constitute the role set in RBAC configuration as a solution, which are the same as the three ideal roles alluded to earlier.

4. EXPERIMENTS

We investigated the performance of the DDRE algorithm on both synthetic and real world datasets. In the process, we varied α to characterize how the resulting RBAC configuration changes. Additionally, we compared DDRE with several related role mining algorithms, including the minimal perturbation role mining algorithm [24] and role mining with latent Dirichlet allocation (LDA) [13]. We defer the description of these methods to the Related Work (Section 5).

Table 1: Statistics for the EMR access log.

	Users	Job Titles	Reasons	Accesses
Total #	8095	140	143	1,138,555

4.1 Description of Datasets

4.1.1 Electronic Medical Record Roles & Access Logs

The real world dataset was extracted from three consecutive months of access logs from the Cerner Corporation's PowerChart electronic medical record (EMR) system in use at Northwestern Memorial Hospital, which is an 854 bed primary teaching affiliate of Northwestern University. All clinicians retrieve clinical context and enter inpatient notes and orders using the system. Each entry of the log contains information about a distinct access made to the EMR, including user-id, patient-id, time, job title of the user, reason for the access, type of service, and location where the access transpired.

Table 1 depicts summary statistics for the access logs. Although the EMR is not based on RBAC, the reason is an option selected when a chart is accessed by the user during a patient's hospitalization and the options available are tied to the job title of the user. As a result, we believe it is reasonable to utilize the reasons as privileges and job titles as roles in the system. Table 2 shows how we acquire an RBAC configuration $c = \langle U, P, R, URA, RPA \rangle$ and a user-permission invocation matrix *UPIM* from the access log.

4.1.2 Synthetic Roles & Access Logs

To allow for replication of our study and comparison to the EMR dataset, we created a synthetic dataset which consists of an RBAC configuration $c' = \langle U', P', R', URA', RPA' \rangle$ and a corresponding *UPIM*. As in the example in Section 3.2, there are several ideal roles, each of which has a corresponding probability distribution over its affiliated permissions.

Table 2: A summary of how the RBAC configuration and *UPIM* are derived from the EMR access logs.

Feature	Derivation Process
U	*The set of users in the access logs.*
R	*The set of job titles in the access logs.*
P	*The union of reason sets available to each job title in R.*
URA	$\|U\| \times \|R\|$ *Boolean matrix. If the* i^{th} *user and* j^{th} *job title (role) co-occur in one entry of the access log,* $URA_{ij} = 1$*; otherwise* $URA_{ij} = 0$.
RPA	$\|R\| \times \|P\|$ *Boolean matrix. If the* j^{th} *reason (permission) belongs to the reason set available to* i^{th} *job title (role),* $RPA_{ij} = 1$*; otherwise* $RPA_{ij} = 0$.
$UPIM$	$\|U\| \times \|P\|$ *real value matrix. If the* i^{th} *user and* j^{th} *reason (permission) co-occur in the same entry of the access log t times, then* $UPIM_{ij} = t$.

To enable a clean analysis, there is no overlap in the permission sets of these roles. We merge the permission sets of several ideal roles to realize an *actual* role in the RBAC system. For each user under one actual role, we utilize the ideal roles hiding in the actual role to generate its corresponding vector in *UPIM*, where the numbers corresponding to one ideal role need to follow the probability distribution of this ideal role. For instance, we can merge two ideal roles $\{p_1, p_2, p_3\}$ and $\{p_4, p_5, p_6\}$ whose distributions are $\{0.2, 0.3, 0.5\}$ and $\{0.1, 0.7, 0.2\}$, respectively, to create an actual role $\{p_1, p_2, p_3, p_4, p_5, p_6\}$. The rates of permissions invoked by each user u_i assigned to this role need to be consistent with the distributions of both ideal roles, which means $UPIM_{i1} : UPIM_{i2} : UPIM_{i3} = 2:3:5$ and $UPIM_{i4} : UPIM_{i5} : UPIM_{i6} = 1:7:2$. The *UPIM* matrix is constructed by performing this procedure for each user. A more detailed example is reported in the Appendix.

For this study, we created 10 ideal roles, and use 10 actual roles, which are derived by merging different sets of the ideal roles as R'. We synthesize 20 users per role (i.e., 200 users in total) as U'. For each actual role, the ideal roles used for merging are randomly selected from the 10 ideal roles. Since the actual roles are represented by permission sets, RPA' is derived accordingly. In addition, we derive P' by uniting the permission sets of all actual roles. Thus, a synthetic RBAC c' is successfully constructed.

4.2 Evaluation Measures

We use two measures to assess the quality of the resulting RBAC system.

RBAC Evolution Distance: This measure characterizes the distance between the old and new RBAC configurations. It directly corresponds to Equation 6.

Outlier Rate: This measure characterizes the homogeneity of users' behavior in the resulting roles. For this measure, we use the rate at which users are predicted to be outliers in the system. The outlier rate is computed as follows. For each role r_l, we perform outlier detection on the corresponding projection matrix M_{r_l} using one-class SVM.[4] To do so, the row vectors in M_{r_l} are split into three equally-sized partitions $\{part_1, part_2, part_3\}$. We pick one partition as the test set, and the remaining two partitions as training

[4]All SVM calculations were performed in *libsvm* [2].

and validation sets for a one-class SVM. After we obtain a one-class SVM model, we perform the outlier detection on the test set. All vectors classified as negatives are designated as outliers. This process is performed in three-fold cross-validation (i.e., three times with a different test set), so that each row vector in M_{r_l} is evaluated. The outlier rate of a role r_l is computed as:

$$\mathsf{or}_l = \frac{\sum_{i=1}^{3}(\#\ of\ outliers\ in\ part_i)}{\#\ of\ row\ vectors\ in\ M_{r_l}} \tag{10}$$

Finally, the outliers from each role are consolidated to calculate the outlier rate for the entire RBAC system:

$$\mathsf{oor} = \frac{\sum_i or_i \cdot n_i}{\sum_i n_i} \tag{11}$$

where n_i is the number of users who are members of role r_i.

Detecting the outlier rate is a more intuitive and straightforward way to measure the homogeneity of the entire RBAC system[5] because the two concepts are strongly related. As proof, Figure 3 shows the relationship for both the EMR and synthetic datasets, where each point is derived from the RBAC configuration from the DDRE algorithm over a range of α values. The correlation coefficient (r^2) for a linear regression was found to be 0.912 and 0.837 for the EMR and synthetic datasets, respectively. Thus, we conclude that the outlier rate is positively correlated with RBAC homogeneity and use it to measure the homogeneity of the system in the following evaluation.

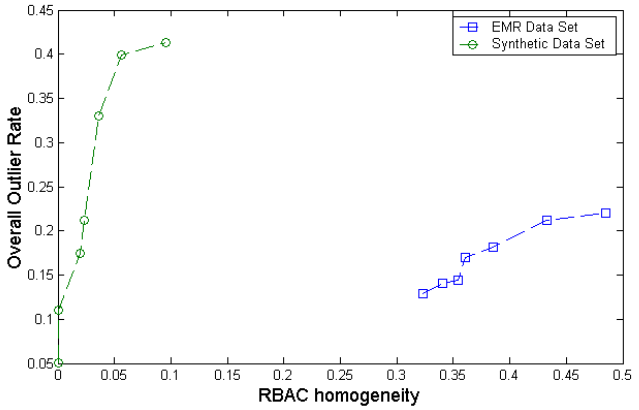


Figure 3: Relationship between RBAC homogeneity and outlier rate.

4.3 Results

4.3.1 Assessing the Tradeoff

All of the following experiments were run on an Intel Core i5 2.40GHz CPU with 4G memory and a Windows XP operating system. We ran the DDRE algorithm with a range of α values to assess its efficiency. Table 3 shows the time consumed by the algorithm on the EMR dataset. The longest time was 21.7 minutes, which shows the algorithm can terminate in a practical amount of time. Moreover, the runtime is directly correlated with α.

Next, we investigated how the RBAC configuration yielded by DDRE changes with α. Figure 4 summarizes this result, where the number near each point in the curves of DDRE corresponds to the value of α used to generate the corresponding RBAC configuration. In this figure, it can be seen that when α biases the system towards behavior, the overall outlier rate is low, whereas the distance between the old RBAC configuration and the resulting RBAC configuration is large. On the contrary, when α is biased towards the distance to old RBAC configuration, the overall outlier rate is high, while the distance between the two RBAC configurations is significantly smaller. In particular, we find that the outlier rate corresponding to $\alpha = 1$ is 41.1% and 87.7% lower than that corresponding to $\alpha = 0$ on the EMR and synthetic datasets, respectively. The Jaccard distance between the two RBAC configurations when $\alpha = 0$ is 90.9% and 100% lower than that corresponding to $\alpha = 1$ on the EMR and synthetic datasets, respectively. This observation indicates that we can obtain an almost identical RBAC configuration to the initial one when $\alpha = 0$. These results suggest that the DDRE algorithm is effective.

Table 3: Runtime of the DDRE algorithm.

α	1	0.97	0.93	0.9	0.8	0.7	0
Runtime(min)	21.7	15.5	12.7	12.5	10.4	7.3	6.4

We also note that the EMR dataset yields a much smaller range of outlier rates and Jaccard distances than the synthetic dataset. We hypothesized that this is because each actual role in the synthetic dataset is composed of more ideal roles than the actual roles in the EMR dataset. For instance, imagine there is an actual role composed of m ideal roles. When we compute the distance from one ideal role to the actual role, a larger m means the ideal role has permission set with a smaller size. This will result in a smaller numerator in Equation 4 and, thus, will yield a larger value for $j()$. Moreover, the ideal role exhibits a strong pattern as a single role, but the more ideal roles that aggregate into an actual role, the faster their patterns are diluted. This leads to a significant increase in outlier detection. By studying the original roles (actual roles) and the resulting roles (ideal roles) yielded by the DDRE algorithm with $\alpha = 1$, we find that each original role in the EMR dataset possesses 1.5 roles on average in the corresponding resulting role set. By contrast, each original role in the synthetic dataset possesses 5.4 roles on average in the corresponding resulting role set. We believe this finding validates our hypothesis.

Figure 4 also depicts the results of the minimal perturbation role mining (RM-MP) and role mining with LDA (RM-LDA) algorithms.[6] The number near each point of the RM-MP curve corresponds to the value of w that controls the balance between the number of roles generated and the *Roles_Roles Distance*, a set-based distance between new role set and old role set (D in the objective function in [24]). It can be seen that the curves for RM-MP have the same

[5]The RBAC homogeneity in Definition 3 could be replaced with the outlier rate. However, RBAC homogeneity has significantly lower time complexity ($O(mn)$, where m is the number of roles which ever exist in DP of Algorithm 1, n is the average number of users contained by each role) and can be computed in a feasible amount of time on a commodity server. By contrast, the outlier rate requires computation on the order of ($O(mn^2)$).

[6]Following the strategy in [13], the number of topics (i.e.,roles) specified for the LDA is $\sqrt{|U|}$.

tendency as that generated through DDRE. This is an intuitive and expected finding. Consider, when w is biased towards the number of generated roles, the algorithm will prefer the roles with larger size to those that are closer to original roles. This can lead to low homogeneity and large distances to the original roles. In addition, we notice that RM-MP yields curves that are close to those from DDRE, however, DDRE has a broader range of solutions, which can be observed by observed at the points when α approaches the boundary cases of 0 and 1. This indicates DDRE can yield better results when α be biased toward either sole objective. The result of RM-LDA on the EMR dataset shows it yields an overall outlier rate that is comparable to the results of DDRE when biased towards permission utilization, however, the RBAC it generated is significantly different than the original RBAC. The result of RM-LDA on the synthetic dataset is in the neighboring region of that of DDRE, but it is easy to find a solution from the curve of DDRE that has both a lower RBAC distance and a lower outlier rate than RM-LDA.

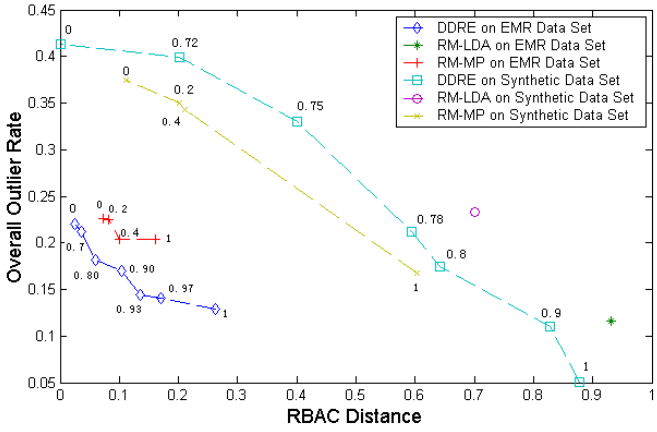


Figure 4: Summary of the tradeoff between the distance of old and new RBAC configurations (i.e., RBAC distance) and the rate of outlying behavior for the EMR and synthetic datasets.

4.3.2 *Influence of SVM Training on Outliers*

Next, we investigated how the results of one-class SVM are influenced with respect to the v parameter.[7] This experiment is performed to determine if there exist patterns in the EMR dataset or if our results are based on random effects. As mentioned earlier, v controls the tradeoff between the fraction of training instances falling into the learned region and the value of the regularization term. As v increases, less instances in the training set will fall into the learned region. So, if the entire dataset follows a pattern, the test set will be distributed in approximately the same region as the training set even though v is decreasing. Otherwise, due to the high diversity of the support vectors, the test set will likely be located in a different region.

To perform this portion of our analysis, we created two uncontrolled versions of *UPIM* for the two data sets used earlier by assigning a random value to $UPIM_{ij}$ that was originally $UPA_{ij} = 1$. The uncontrolled version of *UPIM* is used for simulating the access log without any pattern. We then employed one-class SVM to compute the accuracy (calculated by $1 - \mathsf{oor}$) for the RBAC configurations with the real and uncontrolled *UPIM* matrices for the EMR dataset (called EMR and EMR_{UN}) and synthetic dataset (called SYN and SYN_{UN}). It is expected that the accuracy on the uncontrolled dataset will decrease more quickly than the controlled dataset.

Table 4: Role prediction accuracy as a function of v.

	v =0.16	v=0.21	v=0.26	v=0.31	v=0.36
EMR_{UN}	79.48%	75.48%	71.08%	66.25%	61.93%
EMR	82.48%	77.35%	75.10%	71.51%	67.02%
SYN_{UN}	78.10%	72.00%	66.59%	60.75%	54.93%
SYN	77.18%	73.82%	68.73%	64.18%	58.91%

Table 4 shows the accuracy of one-class SVM with different v on the resulting RBAC configurations. Here it can be seen that the accuracy of SYN_{UN} decreases by 29.67%, while the accuracy on SYN decreases by 23.67%. By performing a proportion test, the latter accuracy decrease rate is slower than the former one with 90% confidence. This observation confirms our suspicion. We further note that the accuracy on EMR_{UN} decreases by 22.08%, while the accuracy on EMR decreases by 18.74%, and the difference between them is also proven statistically significant with 90% confidence by the proportion test, which suggests permission invocation patterns exist in the real EMR dataset.

4.3.3 *Statistics of Generated Roles*

Finally, Figures 5 and 6 provide summary statistics of the roles generated when DDRE is applied to the EMR dataset. In Figure 5, each circle (x, y) represents one role γ. x is calculated by $\mathsf{minjac}(\gamma, R)$ (see Equation 5), where R is the role set of the original RBAC, while y is the outlier rate (see Equation 10) detected for this role. From Figure 5, it can be seen there is a major difference between the distributions of roles yielded by the algorithm with α set to 1 and 0.

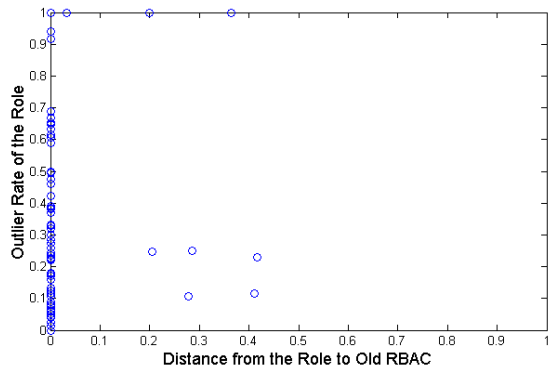
Moreover, we show the marginal distributions of $\mathsf{minjac}(\gamma, R)$ and the outlier rate under different α in Figure 6. The histogram in Figure 6(a) demonstrates that the roles generated by $\alpha = 1$ tend to have less outlying users than the roles generated by $\alpha = 0$. By contrast, the histogram in Figure 6(b) demonstrates that the roles generated by $\alpha = 0$ tend to be closer to the role set in the original RBAC than the roles generated by $\alpha = 1$. These observations further validate the effectiveness of the DDRE algorithm.

5. RELATED WORK

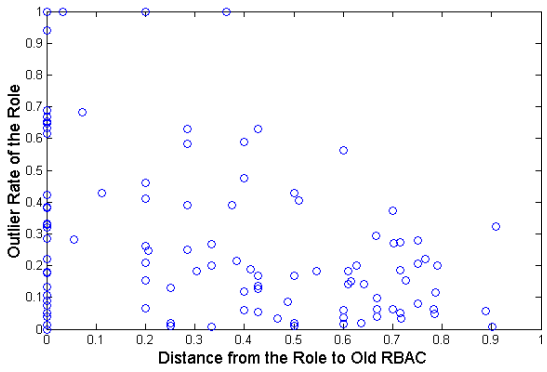
The concept of role engineering was first proposed by Coyne [5]. As mentioned earlier, the process of role engineering can be grossly categorized into top-down and bottom-up strategies. There have been various approaches proposed for top-down approaches (e.g., [20, 17, 15]), but given the time-consuming and costly nature of this approach, it has limited adoption in real settings [17].

Thus, over the past decade, there has been a growing interest in bottom-up approaches, which enables role engineering to be automated with significantly lower cost. Here, we highlight several approaches that are conceptually similar to our work in that they iteratively build larger permission sets

[7]The paratmeter g is determined by the grid search method and is not investigated.

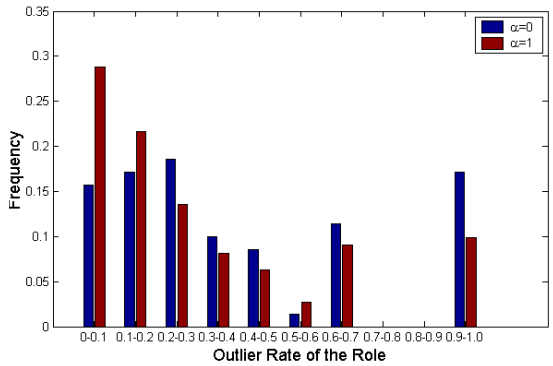


(a) Plot of $\alpha = 0$

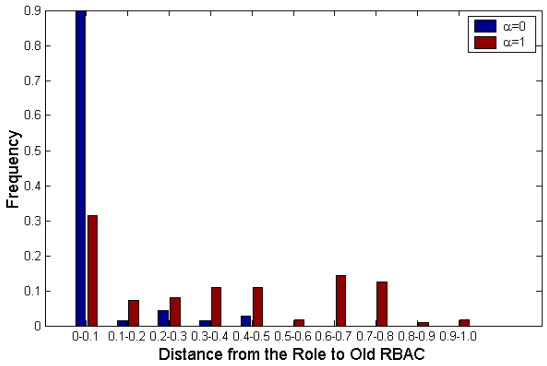


(b) Plot of $\alpha = 1$

Figure 5: Plots of roles denoted by corresponding distance to old RBAC and outlier rate



(a) Distribution of Outlier Rate



(b) Distribution of RBAC Distance

Figure 6: Frequency distributions of (a) outlier and (b) distance rates under $\alpha = 0$ and $\alpha = 1$

for roles. In [23], the goal is to minimize the number of roles and permissions per role. It was shown that this problem is computationally challenging and so a greedy heuristic-driven algorithm was proposed. The algorithm consists of two phases: in the first phase, *FastMiner* [22] produces a set of candidate roles by intersecting each pair of permission sets of users, and then, in the second phase, candidates with the greatest ability to cover the *UPA* (i.e., 1's in the matrix) are selected until coverage is complete. Alternatively, [18] proposes the ORCA algorithm, which generates roles by performing a hierarchical clustering on permission sets. In this process, the quality of a cluster (role) corresponds to the number of users associated with it. [26] use graphs to represent the relations among users, roles, and permissions, and then employs graph optimization to solve the role mining problem. The process begins with a set of possible roles, which is composed of the permission sets of all users. Next, pairs of roles are iteratively selected and are split or merged, to gain the largest improvement on the optimization measure of the resulting graph. While these strategies propose roles, they do not attempt to maximize homogeneity and minimize the distance to an existing set of roles. [4] proposes a role engineering method that leverages organizational information to generate a set of roles with clear business meaning. The method first partitions the data set (user-permission assignment) according to certain appropriate business information (e.g. organization unit). Next, they adopt a divide-and-conquer approach that is to perform role mining on each subset. This approach may produce a RBAC configuration that is close to that built by administrators or experts due to the use of business information that is often used in top-down role engineering. However, like other role mining algorithms, it does not leverage information in access logs, such that roles with high homogeneity could not be searched.

There have been several approaches proposed which attempt to revise roles and leverage permission utilization patterns (which we empirically compared to in the previous section). [24] defines the minimal perturbation role mining problem, whose objective is to find a set of roles that has both small distance to the original roles and a small number of roles in total. The objective function is defined as $f(R) = w \cdot k + (1-w) \cdot k \cdot D$, where k is the number of roles, D is the distance between old and new role sets, and w is a parameter used to control the balance between k and D. In this method, a role is constantly selected from the candidate role sets produced by *FastMiner* [22] according to its value on a heuristic function $f(r) = w \cdot a + (1-w) \cdot a \cdot d$, where a is the remaining 1's in *UPA* covered by this role and d is the distance between this role and the original role set. The

selection process terminates when *UPA* is covered by the selected roles. However, this work is limited in that it neither takes the users' behavior into consideration, nor does it measure the similarity of RBAC configurations. Rather, it only uses the similarity between two role sets. By contrast, [13] takes user behavior into consideration and proposes a simulated annealing approach to mine *URA* and *RPA* with the usage of privileges. This approach begins with a random initialization of *URA* and *RPA*, which is derived from the probability distribution of users over roles, and the probability distribution of roles over permissions calculated by the LDA model learned from the access log. It then iteratively decides if a new pair of *URA* and *RPA* matrices would be accepted to replace the old ones by a λ-distance (a measure of how well they explain the usage of permissions). For simplicity, the resulting *URA* and *RPA* does not necessarily have to be consistent with the original *UPA*. Thus, this work is significantly different than ours in that the resulting RBAC configuration is not necessarily subject to $UPA = URA \otimes RPA$.

6. DISCUSSION AND CONCLUSIONS

This paper proposed a novel role engineering algorithm that enables a controlled evolution of an RBAC configuration based on the utilization of permissions (as documented in access logs). We devised an objective function that balances an administrator's beliefs and actual permission utilization, and defined a role mining problem for finding an RBAC configuration that optimizes this objective. To solve this problem, we proposed a two-phase algorithm. In the first phase, a heuristic function related to the objective is applied to propose a set of candidate roles. In the second phase, each user is assigned roles in the candidate role set to minimize redundancy in role definitions. We then performed an empirical analysis with real and simulated datasets to show that our algorithm can generate appropriate RBAC configurations for various biases of the two competing goals of the objective function.

There are several limitations to our strategy, however, which we highlight to provide a roadmap for future work. First, our strategy is based on permission utilization patterns in an atemporal fashion. This is a simplification of the access logs and neglects that the order in which permissions are invoked may be correlated. Second, our approach is predicated on the hypothesis that there is only one pattern (in the form of a distribution of permission rates) associated with the underlying roles. Yet, it is possible there could be multiple patterns. We plan to extend our strategy to determine when such a situation arises and tease apart these patterns into distinct roles.

7. ACKNOWLEDGEMENTS

This research was supported by grants CCF-0424422 and CNS-0964063 from the National Science Foundation (NSF), R01-LM010207 from the National Institutes of Health (NIH), and HHS-90TR0003/01 from the Office of the National Coordinator for Health Information Technology at the Department of Health and Human Services (HHS). Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the HHS, NIH, or NSF. The authors also thank Nathan Sisterson for supplying the dataset used in this study, Ian Molloy of IBM Research for discussing the specification of the parameters for the simulated annealing algorithm for LDA and Jaideep Vaidya of Rutgers University for discussing the implementation of FastMiner.

8. REFERENCES

[1] E. Bertino, A. Kamra, E. Terzi, and A. Vakali. Intrusion detection in RBAC-administered databases. In *Proceedings of the* 21^{st} *ACSAC*, pages 170–182, 2003.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27, 2011.

[3] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[4] A. Colantonio, R. D. Pietro, A. Ocello, and N. V. Verde. A new role mining framework to elicit business roles and to mitigate enterprise risk. *Decision Support Systems*, 50:715–731, 2011.

[5] E. J. Coyne. Role engineering. In *Proceedings of the* 1^{st} *ACM Workshop on Role-Based Access Control*, 1995.

[6] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Proceedings of Applications of Data Mining in Computer Security*, pages 78–100. Kluwer, 2002.

[7] L. Freeman, A. Romney, and S. Freeman. Cognitive structure and informant accuracy. *American Anthropologist*, 89:310–325, 1987.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, NY, 1990.

[9] T. H. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Evaluating strategies for similarity search on the web. In *Proceedings of the* 11^{th} *International Conference on the World Wide Web*, pages 432–442, 2002.

[10] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. Technical report, Dept. of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2003.

[11] L. M. Manevitz and M. Yousef. One-class SVMs for document classification. *The Journal of Machine Learning Research*, 2:139–154, 2002.

[12] S. Mehrotra, C. Butts, D. V. Kalashnikov, N. Venkatasubramanian, R. Rao, and et al. Project RESCUE: challenges in responding to the unexpected. In *Proceedings of SPIE*, volume 5304, pages 179–192, 2004.

[13] I. Molloy, Y. Park, and S. Chari. Generative models for access control policies: applications to role mining over logs with attribution. In *Proceedings of the* 17^{th} *ACM Symposium on Access Control Models and Technologies*, pages 45–56, 2012.

[14] J. Park and J. Giorgano. Role-based profile analysis for scalable and accurate insider-anomaly detection. In *Proceedings of the* 25^{th} *IEEE International Performance, Computing, and Communications Conference*, pages 470–476, 2006.

[15] 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 *Proceedings of the* 5^{th} *ACM Workshop on Role-Based Access Control*, pages 103–110, 2000.

[16] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29:38–47, 1996.

[17] 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 the* 6^{th} *ACM Symposium on Access Control Models and Technologies*, pages 3–9, 2001.

[18] J. Schlegelmilch and U. Steffens. Role mining with ORCA. In *Proceedings of the* 10^{th} *ACM Symposium on Access Control Models and Technologies*, pages 168–176, 2005.

[19] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[20] D. Shin, G. Ahn, S. Cho, and S. Jin. On modeling system-centric information for role engineering. In *Proceedings of the* 8^{th} *ACM Symposium on Access Control Models and Technologies*, pages 169–178, 2003.

[21] M. Strembeck. Scenario-driven role engineering. *IEEE Security and Privacy Magazine*, 8:28–35, 2010.

[22] J. Vaidya and V. Atluri. Roleminer: mining roles using subset enumeration. In *Proceedings of the* 13^{th} *ACM Conference on Computer and Communications Security*, pages 144–153, 2006.

[23] J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: A formal perspective. *ACM Transactions on Information and System Security*, 13:27, 2010.

[24] J. Vaidya, V. Atluri, Q. Guo, and N. Adam. Migrating to optimal RBAC with minimal perturbation. In *Proceedings of the* 13^{th} *ACM symposium on Access Control Models and Technologies*, pages 11–20, 2008.

[25] G. Wu, S. Osborn, and X. Jin. Database intrusion detection using role profiling with role hierarchy. In *Proceedings of the* 6^{th} *VLDB Workshop on Secure Data Management*, pages 33–48, 2009.

[26] D. Zhang and T. E. K. Ramamohanarao. Role engineering using graph optimisation. In *Proceedings of the* 10^{th} *ACM Symposium on Access Control Models and Technologies*, pages 139–144, 2007.

APPENDIX

A. SYNTHETIC DATA SET GENERATION

We use the ideal roles in example of Figure 2 in this section. These roles are $r_1 = \{p_1, p_2\}$, $r_2 = \{p_3, p_4, p_5\}$ and $r_3 = \{p_6, p_7\}$ and their probability distributions are $\{0.2, 0.8\}$, $\{0.2, 0.5, 0.3\}$, and $\{0.7, 0.3\}$, respectively. When synthesizing the actual role γ_i, we first select a random set of ideal roles (i.e., each role has a 0.5 probability of being selected). Next, we merge all roles in the set to form γ_i. For instance, r_1 and r_2 are selected, then $\gamma_i = \{p_1, p_2, p_6, p_7\}$.

Then, we simulate users for the γ_i. Let us use u_j as an example. The row vector corresponding to u_j in $UPIM$ needs to be consistent with the probability distributions associated with r_1 and r_3. When we try to assign values to $UPIM_{j1}$ and $UPIM_{j2}$ (which correspond to r_1), we need to ensure that $UPIM_{j1} : UPIM_{j2} = 1 : 4$. To finish the assignment, we first generate a random integer (e.g., 100), and repeat the following procedure 100 times: we add one to $UPIM_{j1}$ with probability 0.2 and if we fail to add to $UPIM_{j1}$, we add one to $UPIM_{j2}$. Similarly, we assign values to $UPIM_{j6}$ and $UPIM_{j7}$. For $UPIM_{jk}$ where $k = 3, 4, 5$, the value is set to 0 because the role associated with u_j does not contain p_k.