Towards User-Oriented RBAC Model

Haibing Lu, Yuan Hong, Yanjiang Yang, Lian Duan, Nazia Badar

Towards User-Oriented RBAC Model

Haibing Lu[1], Yuan Hong[2], Yanjiang Yang[3], Lian Duan[4], Nazia Badar[5]

Santa Clara University[1], Rutgers University[2,5], I2R Singapore[3], New Jersey Institute of Technology[4]

hlu@scu.edu[1], yhong@cimic.rutgers.edu[2], yyang@i2r.a-star.edu.sg[3], lian.duan@njit.edu[4], nbadar@pegasus.rutgers.edu[5]

Abstract. Role mining recently has attracted much attention from the role-based access control (RBAC) research community as it provides a machine-operated means of discovering roles from existing permission assignments. While there is a rich body of literature on role mining, we find that user experience/perception - one ultimate goal for any information system - is surprisingly ignored by the existing works. This work is the first to study role mining from the end-user perspective. Specifically, based on the observation that end-users prefer simple role assignments, we propose to incorporate to the role mining process a user-role assignment sparseness constraint that mandates the maximum number of roles each user can have. Under this rationale, we formulate user-oriented role mining as two specific problems: one is user-oriented exact role mining problem (RMP), which is obliged to completely reconstruct the given permission assignments, and the other is user-oriented approximate RMP, which tolerates a certain amount of deviation from the complete reconstruction. The extra sparseness constraint poses a great challenge to role mining, which in general is already a hard problem. We examine some typical existing role mining methods to see their applicability to our problems. In light of their insufficiency, we present a new algorithm, which is based on a novel dynamic candidate role generation strategy, tailored to our problems. Experiments on benchmark datasets demonstrate the effectiveness of our proposed algorithm.

Keywords: Access Control, Role Mining, Sparseness, Binary, Optimization

1 Introduction

Role-based access control (RBAC) restricts system access to authorized users by assigning permissions to roles and then assigning roles to users. RBAC has become a de facto access control model, due to its many advantages, including the convenience of authorization allocation and the reduction of the system administrative workload. Enterprises still employing their old access control systems want to migrate to RBAC. To accomplish the migration, the first phase is to define a good role set. While the role defining problem is seemingly straightforward, it has been recognized as one of the costliest phases in the implementation of RBAC and poses a great challenge to the system engineers. The difficulty comes from the fact that a RBAC system engineer usually has little knowledge on the semantic meanings of user responsibilities and business processes within an enterprise.

Role mining has proven to be an effective (machine-operated) means of discovering a good role set. Its key idea is to utilize data mining technologies to extract patterns from existing permission assignments of the old access control system, which are then used to establish roles. This greatly facilitates the implementation of RBAC (by migrating from the old access control system). In the literature, role mining has been extensively studied. In a nutshell, the existing literature investigates role mining with different objectives, including minimization of the number of roles, minimization of the administration cost, minimization of the complexity of the role hierarchy structure, and others. However, we find that none of the existing works has ever considered to improve end-user experience (of the underlying RBAC system), which should be one ultimate goal for any practical information system. Needless to say, users' experience/perception of a system represents system usability and directly affects the eventual success of the system in practice. As such, we argue that user-friendliness should be an essential criterion for evaluating the quality of role mining.

In this paper, we study user-oriented role mining, being the first to explore the role mining problem from the end-user perspective. Our daily experiences tell us that end users often prefer fewer role assignments; as long as a user acquires all the needed permissions, the fewer roles she has to bear, the better usability she may feel upon the system. That is, from the end-user perspective, a good RBAC system should have as sparse user-role assignments as possible. This coincides with an advantage of RBAC: recall that one reason accounting for the wide acceptance of RBAC is that it allows users to carry very few roles while enjoying their (potentially many) access rights. However, on the flip side, if we create a unique role for every user, in which case user-role assignments are trivially the most sparse, then the resultant RBAC system would contain too many roles. This absolutely contradicts to a premise of RBAC, which is to map permission roles with functional roles within an organization. As such, a user-oriented RBAC solution should not compromise other advantages of RBAC. To this end, we propose to limit the maximum number of roles a user can take on top of regular role mining. Such a strategy would well balance user friendliness and other system quality factors such as administrative overhead. While the idea is clear, the added constraint poses extra challenges to role mining, considering that role mining in general has already been a hard problem. Towards tackling the obstacle, we make the following contributions: (1) we formulate user-oriented role mining as two specific problems, i.e., user-oriented exact RMP (Role Mining Problem) and user-oriented approximate RMP; (2) in searching for efficient solutions to the formulated problems, we examine several typical role mining algorithms and reveal that they do not meet our needs; (3) in view of the weaknesses of the existing algorithms, we present an efficient algorithm, tailored to the user-oriented role mining problems; (4) to investigate the effectiveness of our algorithm, we conduct experiments on benchmark datasets and obtain promising experimental results.

The remainder of the paper is organized as follows. Section 2 reviews existing role mining works in the literature. Section 3 presents the user-oriented role mining problem. Section 4 presents optimization models. The heuristic algorithm is provided in Section 5. Experimental results on benchmark access control data sets are reported in Section 6. Section 7 concludes the paper.

2 Related Work

The concept of role engineering was introduced in 1995 by Coyne [1]. It aims to define an architectural structure that is complete, correct and efficient to specify the organization's security policies and business functions. Coyne's approach is a top-down process oriented strategy for role definition. With the top-down approach, one starts from requirements and successively refine the definitions to reflect the business functions [13].

Top-down approaches are only suitable for small size enterprises. For medium or large size cases, the bottom-up approach that utilizes the existing user-permission assignments to formulate roles is preferred. In particular, data mining techniques are often employed in the bottom-up approach to identify promising roles. Kuhlmann et al. [5] coined the concept of role mining using data mining. In [14], an algorithm ORCA is proposed to build a hierarchy of permission clusters using data mining technologies. However, overlap between roles is not allowed in ORCA, which contradicts to normal practice in real applications. Vaidya et al. [18] propose a subset enumeration approach, which can effectively overcome this limitation.

An inherent issue with all of the above approaches is that there is no formal notion for goodness of role. Vaidya et al. [15] propose to use the number of roles to evaluate the goodness of a role set. They [16] also introduce to use the administrative task as an evaluative criterion. Role hierarchy is another important evaluative criterion, as it is closely related to the semantic meanings of roles. Related works on role hierarchy include [10]. Ma et al. [9] also use weights to combine multiple objectives. Our work in this paper strengthens this line of research by being the first to incorporate user experience/perception as an extra evaluative criterion in role mining.

Beside the above works on finding a role set with respect to different criteria, there are other interesting works with different flavors. Lu et al. [6, 8] present an optimization framework for role engineering. They even extend their work to incorporate negative authorizations in [7]. Frank et al. [3] provide a probabilistic model to analyze the relevance of different kinds of business information for defining roles that can both explain the given user-permission assignments and describe the meanings from the business perspective. They [4] also introduce a model to take the attributes of users into account. Studies on role mining in the presence of noisy data are presented in [17].

3 Conflict Resolution

In this section, we study and formulate user-oriented role mining. As we have discussed in the introduction, from users' perspective a user-friendly RBAC system should assign as few roles as possible to each user; no user is happy with being overwhelmed by assuming too many roles (titles). Ideally, a user would wish to carry only one role given that the role provides with him all the necessary access privileges for him to work and function smoothly. Indeed, in reality most organization's systems are designed that way. For example, in a school system, the majority of people carry only one role among STUDENT, FACULTY, STAFF, and VISITOR. In a software company, most employees are either ACCOUNTANT, ENGINEER, or MANAGER. Thus, user-oriented role mining is characterized with the fact that user-permission assignments should be sparse. This gives rise to the definition of user-oriented role mining, as stated below.

Definition 1. *(**User-Oriented Role Mining**) User-Oriented Role Mining is to discover a RBAC solution with respect to some evaluation criterions, such that after role assignments users get the same permissions as before and the resultant user-role assignments are sparse.*

To concretize user-oriented role mining, we have two questions to answer: (1) what evaluation criterion should be used to evaluate the goodness of a RBAC solution? (2) what level of spareness of user-role assignments is appropriate? Towards answering these questions, we examine existing role evaluation criteria for some insights.

3.1 Existing Evaluation Criteria

Role mining is typically formulated as certain optimization problems with objectives and constraints. As summarized by Molloy et al. at [10], there are five main factors which can be used to evaluate the goodness of a RBAC solution. They are the number of roles $|R|$, the complexity of user-role assignments $|UA|$, the complexity of role-permission assignments $|PA|$, the number of direct user-permission assignments $|DUPA|$, and the complexity of reduced role hierarchy $|t\_reduce(RH)|$. Among them, $|R|$, $|UA|$, and $|PA|$ are routine notations in RBAC, and no further exposition is needed on them. Direct user-permission assignments $DUPA$ imply that roles of one single user are treated as special roles. $|DUPA|$ is the amount of such direct user-permission assignments in a deployed RBAC solution. Role hierarchy $RH \subseteq R \times R$ represents the partial order over roles R. $t\_reduce(RH)$ denotes the transitive reduction of the role hierarchy.

Almost all role mining evaluative criteria can be generally described with the weighted structural complexity measure introduced in [10], which sums up the above five factors, with possibly different weights for each factor.

Definition 2. *(**Weighted Structural Complexity Measure**) Given a weight vector $W =< w_r, w_u, w_p, w_d, w_h >$, where $w_r, w_u, w_p, w_d, w_h \in \mathcal{Q}^+ \cup \{0\}$ [1], the weighted structural complexity of an RBAC state γ, which is denoted as $wsc(\gamma, W)$ is computed as:*

$$\begin{aligned} wsc(\gamma, W) = w_r * |R| + w_u * |UA| + w_p * |PA| \\ + w_d * |DUPA| + w_h * |t\_reduce(RH)| \end{aligned}$$

Role mining in general involves minimizing $wsc(\gamma, W)$. However, a minimization implicating all factors not only is too complex, but may not lead to a good RBAC system, as they may counteract with each other in the minimization. Depending on the objective to achieve, a specific role mining task often chooses to minimize a subset of factors relevant to the underlying objective. In particular, minimizing the number of roles $|R|$ might be the most studied role mining problem, where in the weighted structural complexity measure, w_r is a positive number while others are all 0. Such a specific role mining problem is referred to as basic RMP (Role Mining Problem), and it has been proven equivalent to the classic set cover problem. It is true that the number of roles can describe the complexity of a RBAC solution in a certain way. However, sometimes a RBAC solution minimizing the number of roles might not be able to capture

[1] $\mathcal{Q}^+$ is the set of all non-negative rational numbers

the internal organizational structure within an company. We shall show this by a toy example with six users, three permissions, and user-permission assignments such that $u1 : \{p1, p2, p3\}$, $u2 : \{p1, p2, p3\}$, $u3 : \{p1, p2, p3\}$, $u4 : \{p1\}$, $u5 : \{p2\}$, $u6 : \{p3\}$.

Under the objective of minimizing $|R|$, it gives a solution such that each individual permission is a role. As a result, there are three roles in total and $u1$, $u2$ and $u3$ are assigned to three roles to cover their permissions. However, even without the semantic information of permissions and the knowledge of user responsibilities, by simply observing the data set, one would conjecture that the permission set of $\{p1, p2, p3\}$ should be one role, as the permission set is shared by three out of six users.

However, if we incorporate the consideration of the size of the total user-role assignments, it can lead us to the right track. With the goal of minimizing $|UA|$, each user will get one role and each unique set of user permissions is treated as a role. As a result, the role of $\{u1, u2, u3\}$ is discovered. This example suggests the importance of incorporating the complexity of $|UA|$ as a part of the role mining goal.

The sum of user-role assignments and role-permission assignments of $|UA| + |PA|$ is commonly viewed as the representation of the system administrative cost. The minimization of $|UA| + |PA|$ is called edge RMP [16]. However, as far as an end-user is concerned, $|UA|$ is the only part that she can experience of a RBAC system. A user would not care how complex PA is. For example, a student would not care about how many permissions her STUDENT role actually contains, and all she cares is the simplicity of executing her role (or roles).

Other evaluative criteria, such as minimizing the complexity of the resultant role hierarchy [10], are also more from the system administrator perspective, rather than the end-user perspective.

By examining existing evaluative criterion, we found that the only factor in the Weighted Structural Complexity Measure that matters to end-users is the size of user-role assignments $|UA|$. However, if the user-oriented RMP is defined as the minimization of $|UA|$, the trivially optimized solution is to create a unique role for every user. That absolutely contradicts to the premise of role mining, which is to map permission roles with functional roles. As such, a user-oriented RMP solution should balance the user-friendliness and the overall quality of the system. Among five factors in the weighted structural complexity measure, the number of roles $|R|$ would be the best representative of the succinctness and goodness of a RBAC solution and is also the most studied criterion. So we propose to define the user-oriented RMP as the minimization of the combination of the number of roles and the size of user-role assignments, $w_r * |R| + w_u * |UA|$.

3.2 User-Oriented RMP

User-Oriented Exact RMP Given m users, n permissions, user-permission assignments $UPA_{m\times n}$, and the evaluation criteria of $w_r * |R| + w_u * |UA|$, the user-oriented exact RMP is to find $UA_{m\times k}$ and $PA_{k\times n}$ to completely reconstruct UPA while minimizing the evaluation criterion. It can be formulated as below:

$$\begin{aligned} &min \;\; w_r * |R| + w_u * |UA_{m\times k}| \\ s.t. &\Big\{ UPA_{m\times n} = UA_{m\times k} \otimes PA_{k\times n} \end{aligned} \tag{1}$$

where $\otimes$ is the Boolean product operator [6].

However, directly working on this formulation has two difficulties. First, it is not easy to determine weights of w_r and w_u in practice, as a role mining engineer may not have a global sense on the importance of $|R|$ and $|UA|$. Second, it is difficult to solve an optimization problem with a complex objective function. It would be relatively easier to solve an optimization problem with an objective of either $|R|$ or $|UA|$. In light of them, we redefine the user-oriented exact RMP as the following.

Problem 1 (User-Oriented Exact RMP). Given m users, n permissions, user-permission assignments $UPA_{m\times n}$ and a positive number t, it is to discover a role set $PA_{k\times n}$ and the user-role assignments $UA_{m\times k}$ such that: (1) the number of roles k is minimized, (2) the role assignments UA and the permission-role assignments PA accurately and completely reconstruct the existing user-permission assignments UPA, and (3) no user gets more than t roles.

Mathematically, it can be described in an optimization form as follows.

$$min\ \ k$$
$$s.t. \begin{cases} UA_{m\times k} \otimes PA_{k\times n} = UPA_{m\times n} \\ \sum_j UA(i,j) \leq t,\ \forall i \\ UA \in \{0,1\}^{m\times k}, PA \in \{0,1\}^{k\times n} \end{cases} \tag{2}$$

It is not difficult for a role mining engineer to find out the maximum roles a user can have. For example, it could be achieved through discussions with company operators and an investigation of the general organizational structure of the company. When the maximum roles each user can have is limited to a small number, $|UA|$ is naturally enforced to be small.

Another property is that Equation (2) can be easily converted to Equation (1) with the method of Lagrange multipliers. If we move the constraint $\sum_j UA(i,j) \leq t$ to the objective function by adding $\sum_j UA(i,j) - t$ as a penalty component, Equation 2 becomes

$$min\ \ |R| + \sum_i \lambda_i (\sum_j UA(i,j) - t)$$
$$s.t. \left\{ UPA_{m\times n} = UA_{m\times k} \otimes PA_{k\times n} \right. \tag{3}$$

where λ_i is the Lagrange multiplier for the constraint of $\sum_j UA(i,j) \leq t$. Further, we could assume that all Lagrange multipliers have the same value λ. Then the equation is changed to the following.

$$min\ \ w_r * |R| + \lambda |UA| - \lambda * t$$
$$s.t. \left\{ UPA_{m\times n} = UA_{m\times k} \otimes PA_{k\times n} \right. \tag{4}$$

Since $\lambda * t$ is a constant, it can be dropped from the objective function. Now the resultant optimization problem is the same as Equation 1. The effect of adjusting the Lagrange multiplier (penalty parameter) λ is equivalent to adjusting k, the maximum roles a user can have.

User-Oriented Approximate RMP Role mining with exact coverage of permission assignments is only suitable when the given permission assignments contain no error. The recent research results on the role mining on noisy data[12, 17]. suggest that when the given user-permission assignments contain noise, it is not necessary to enforce a complete reconstruction, as it causes the over-fitting problem. In such cases, approximate role mining may return better results. So in this paper, we also consider the user-oriented approximate RMP, which is defined as below.

Problem 2 (User-Oriented Approximate RMP). Given m users, n permissions, user-permission assignments $UPA_{m\times n}$, a positive integer number t and a positive fractional number δ, it is to discover a role set $PA_{k\times n}$ and the user-role assignments $UA_{m\times k}$ such that: (1) the number of roles k is minimized, (2) the role assignments UA and the role set PA reconstruct the existing user-permission assignments UPA with the error rate less than δ, and (3) no user gets more than t roles.

The problem can be roughly described in the following optimization form.

$$min\ \ k$$
$$s.t. \begin{cases} ||UA_{m\times k} \otimes PA_{k\times n} - UPA_{m\times n}||_1 \leq \delta \cdot \sum_{ij} UPA_{ij} \\ \sum_j UA(i,j) \leq t,\ \forall i \\ UA \in \{0,1\}^{m\times k}, PA \in \{0,1\}^{k\times n} \end{cases} \tag{5}$$

3.3 NP-hardness

Recall that the basic RMP is to minimize the number of role while the resultant RBAC solution completely reconstructs the given user-permission assignments. The user-oriented exact RMP is a generalization of the basic RMP. If we make the number of the maximum roles each user can have be a large enough number, so that the sparseness constraint does not take effect, then the user-oriented exact RMP becomes the basic RMP. The basic RMP is known to be NP-hard, as it can be reduced to the classic NP-hard set cover problem [15]. Therefore, the user-oriented exact RMP is NP-hard. Similarly, the user-oriented approximate RMP is a generalization of the approximate RMP, which is NP-hard. Thus, it is also NP-hard.

4 Optimization Model

Among many existing role mining approaches, the optimization approach has been favored by researchers, due to the existence of many public and commercial optimization software packages. The user-oriented RMP problems can be formulated by optimization models as well, which enables an engineer to directly adopt an existing software package.

We formulate the user-oriented exact RMP first, which can be viewed as a variant of the basic RMP with a constraint that each user cannot have more than t roles. Suppose we have located a set of q candidate roles, represented by a binary matrix

$CR \in \{0,1\}^{q \times n}$, where $CR_{kj} = 1$ means candidate role k contains permission j. Then the user-oriented exact RMP is reduced to finding the minimum roles from CR to completely reconstruct existing user-permission assignments while no one can have more than t roles. The problem can be formulated as the following ILP.

$$minimize \sum_k d_k$$
$$\begin{cases} \sum_{k=1}^{q} UA_{ik} CR_{kj} \geq 1, if\ UPA_{ij} = 1 \\ \sum_{k=1}^{q} UA_{ik} CR_{kj} = 0, if\ UPA_{ij} = 0 \\ d_k \geq UA_{ij}, \qquad \forall i,j \\ \sum_j UA_{ij} \leq t\ \forall i \\ d_k, UA_{ij} \in \{0,1\} \end{cases} \tag{6}$$

In the model, UA_{ik} and d_k are variables. The detailed description of the model is given as follows:

- Binary variable UA_{ik} determines whether candidate role k is assigned to user i and binary variable d_k determines whether candidate role k is selected. So the objective function $\sum_k d_k$ represents the number of selected roles.
- The first constraint enforces that if user i has permission j, at least one role containing permission j has to be assigned to user i.
- The second constraint enforces that if user i has no permission j, no role containing permission j can be assigned to user i.
- The third constraint $d_k \geq UA_{ij}$ ensures d_k to be 1 as long as one user has role k.
- $\sum_j UA_{ij} \leq t$ enforces that a user cannot have more than t role assignments.

The user-oriented approximate RMP can be viewed as a variant of the approximate RMP with the constraint that no user can have more than t roles. Similarly, we simplify the problem by locating a candidate role set CR. At the basis of the ILP formulation for the approximate RMP, an ILP formulation for the user-oriented approximate RMP is presented as follows.

$$minimize \sum_t d_k$$
$$\begin{cases} \sum_{k=1}^{q} UA_{ik} CR_{kj} + V_{ij} \geq 1, if\ UPA_{ij} = 1 \\ \sum_{k=1}^{q} UA_{ik} CR_{kj} - V_{ij} = 0, if\ UPA_{ij} = 0 \\ MU_{ij} - V_{ij} \geq 0, \forall i,j \\ U_{ij} \leq V_{ij}, \forall i,j \\ \sum_i \sum_j U_{ij} \leq \delta \cdot \sum_{ij} UPA_{ij} \\ d_k \geq UA_{ik}, \qquad \forall i,k \\ \sum_j UA_{ij} \leq t\ \forall i \\ d_j, UA_{ik}, U_{ij} \in \{0,1\}, V_{ij} \geq 0 \end{cases} \tag{7}$$

In the model, UA_{ik}, V_{ij}, U_{ij} and d_k are variables and M is a large enough constant. The detailed descriptions of the model are given as follows:

- In the first two constraints, V_{ij} acts as an auxiliary variable. Without V_{ij}, the constraints would enforce the exact coverage as the ILP model for the user-oriented

exact RMP. With the existence of V_{ij}, the exact coverage constraint is relaxed. The value of V_{ij} indicates whether the constraint for element (i, j) is violated.
- The third and fourth constraints convert V_{ij} to a binary value U_{ij}. If V_{ij} is 1, which means the constraint for element (i, j) is violated, U_{ij} has to be 1; otherwise U_{ij} is 0. The fifth constraint $\sum_{i,j} U_{ij} \leq \delta \cdot \sum_{ij} UPA_{ij}$ enforces the error rate to be less than δ.
- The constraint of $d_k \geq UA_{ik}\ \forall i, k$ enforces d_k to be 1 as long as a user is assigned to role K. So the objective function represents the number of roles being selected.
- $\sum_j UA_{ij} \leq t$ ensures no user gets more than t role assignments.

Although the optimization framework allows us to directly adopt fruitful optimization research results, the ILP in general is NP-hard. Existing algorithms and software packages for general ILP problems only work for small-scale problems. For mid to large size RMP problems, specially designed efficient heuristics are still required.

5 Heuristic Algorithm

In this section, we propose a tailored algorithm for the two user-oriented RMP variants formulated above. It is a heuristic solution, employing an iterative approach to discover roles. The key of our algorithm is a dynamic role generation strategy. Lately, we happened to notice that the idea of dynamic role generation was briefly mentioned in [11], but no further details were seen.

5.1 User-Oriented Exact RMP

The user-oriented exact RMP is to find a minimum set of roles to accurately and completely reconstruct the existing user-permission assignments with the constraint that no user can have more than t roles. Before coming to the details of our algorithm, we start by introducing a preprocessing stage that helps to reduce the problem complexity.

In the preprocessing stage, there are two steps. The first step is to reduce the data size by removing users with the same permission assignments. This step is also employed in other role mining methods, such as [18]. To do so, we group all users who have the exact same set of permissions, which can be done in a single pass over the data by maintaining a hash table of the sets of permissions gradually discovered.

The second step identifies a subset of users U' who have permissions that no other people have. These user-permissions assignments, $\{UA_{i:} | i \in U'\}$, will be included into the final role set. In other words, these users only get one role, which are themselves. Our argument is that if a user has a permission that is exclusively for herself, she must have at least one role containing that permission, and that role is not shared by other people. As such, from the end-user perspective, why not simply package all permissions of that user as one role and assign the only role to her? Therefore, the number of role assignments is significantly reduced while without increasing the total role number. This preprocessing step can significantly reduce the data size as well and simplify the subsequent role mining task.

Note that after the two preprocessing steps, in the remaining data, all user permissions assignments are unique and every permission is assigned to at least two users.

The general structure of our algorithm is to iteratively choose a candidate role and assign it to users until all existing permission assignments are covered while the constraint that no user gets more than t roles is carefully respected. We mention that such an idea of iterative role assignment has also been used in many other role mining methods such as the Lattice Model [2], the FastMinder [18] and the optimization-based heuristic [6]. The distinguishing element of our algorithm is the way of generating candidate roles.

The core of our algorithm is a dynamic role generation strategy. All of the other role mining algorithms generate a static set of candidate roles. Given n permissions, there are 2^n possible roles. If we consider too many candidate roles, the computing time is expensive. Conversely, if we consider only a very limited set of candidate roles, we might not be able to find a good role set. To avoid the extreme cases, our strategy is dynamic candidate roles generation. Specifically, rather than generating a static set of roles at the start of the algorithm, we generate a small set of promising roles at each iteration of the algorithm and the role set is updated according to the remaining user-permission assignments as the algorithm proceeds. There are two advantages: (i) we do not need to maintain and consider a large candidate role set all the time; (ii) the candidate role pool always keeps the potentially interesting roles.

In particular, we always consider the remaining user-permission assignments as potentially interesting roles. For instance, consider Table 1 as the existing user-permission assignments. Our algorithm treats the permission assignments for each user as a candidate role. So in this case, there are three candidate roles: cr1 (0 0 1 1 1 1), cr2 (0 0 1 1 0 0), and cr3 (1 1 1 1 0 0).

	p1	p2	p3	p4	p5	p6
u1	0	0	1	1	1	1
u2	0	0	1	1	0	0
u3	1	1	1	1	0	0

Table 1. Existing Assignments

	p1	p2	p3	p4	p5	p6
u1	0	0	0	0	1	1
u2	0	0	0	0	0	0
u3	1	1	0	0	0	0

Table 2. Remaining Assignments

Suppose now $cr2$ is chosen and it is assigned to all of the three users. Then the remaining permission assignments become Table 2, and they are treated as candidate roles for the next step in the algorithm. So the candidate roles are updated to be the following: cr1 (0 0 0 0 1 1) and cr2 (1 1 0 0 0 0).

With the candidate roles being defined, we need to figure out two things: (1) how to select a candidate role at each step? (2) how to enforce the constraint that no user can have more than t roles. For the first question, there are some studies and discussions in the literature. Here are some well known strategies. Vaidya et al. [18] chooses the candidate role which covers the most remaining permission assignments. Ene et al. [2] selects the candidate role with the least number of permissions. We have tested both of them and found out they do not work well in our case. As such, we use an alternative strategy: we choose the candidate role which covers the most users. In other words, the selected candidate role can be assigned to the most users. In fact, our strategy is justified by that the need for a permission set to become a role comes from the fact that they are

Algorithm 1 User-Oriented Exact RMP

Input: UPA, t
Output: UA, PA
1: $UA \leftarrow \emptyset, PA \leftarrow \emptyset, UPA' \leftarrow \emptyset$;
2: $CRoles \leftarrow UPA$;
3: **while** $UPA' \neq UPA$ **do**
4: Call RSelector;
5: Call CGenerator;
6: **end while**

shared by many people. Suppose that $\{p1, p2\}$ are possessed by three people, while $\{p1, p2, ..., p10\}$ are possed by only one person. It is more reasonable to make $\{p1, p2\}$ as a role than $\{p1, p2, ..., p10\}$. To illustrate this candidate role selection strategy, look at Table 1 again. Among those three candidate roles, (0 0 1 1 0 0) can be assigned to three people, so it is chosen.

To enforce the constraint that no user gets more than t roles, we make some special arrangement, when a user U_i has been covered by $t - 1$ roles and still has uncovered permissions. In such a case, we either need to create a role to cover all remaining permissions of U_i or revoke roles that have been assigned to U_i. Suppose we create a new role which consists of all remaining permissions of U_i and assign it to U_i. Then, we may need to check if the new role can be repetitively used by other users, otherwise it is costly. If no one else can take the new role, we make all permissions of U_i as a single role. Thus we can revoke all roles that have been assigned to U_i and assign the sole role to the user. In this way, at the same cost of adding one role, the role assignments for the user are significantly reduced, which is exactly the goal of this work. Based on this idea, the following steps are implemented to enforce that no user gets more than t roles.

When a user U_i has been covered by $t-1$ roles and still has uncovered permissions, we stop choosing roles from the candidate role set. Instead, we first treat the uncovered permissions of U_i as a candidate role, and evaluate its suitability by checking if some other user who has been assigned less than $t - 1$ roles will take this candidate role. If so, it means that the role can be repetitively used, then we include the role into the final role set and assign it to users. Otherwise, we discard it, and then make all permissions of the user U_i as a role, assign the role to U_i and delete all of the other role assignments to U_i. The complete algorithm is stated in Algorithms 1-3 ($UA_{i:}$ denotes the ith row of UA, which represents the role assignments to user i; $UPA_{i:}$ denotes the ith row of UPA, which represents the permissions assigned to user i).

5.2 User-Oriented Approximate RMP

The user-oriented approximate RMP is the same as the user-oriented exact RMP, except that the complete reconstruction is not required. The above algorithm for the user-oriented exact RMP can be easily modified for the the user-oriented approximate RMP by changing the termination condition from $UPA \neq UPA'$ to $||UPA - UPA'|| > \delta$. Consequently, the algorithm stops early and avoids covering too much noisy information.

Algorithm 2 RSelector

Input: $UPA, UPA', CRoles, UA, t$
Output: r, UA, PA, UPA'
if $\exists i\ s.t.\ |UA_{i:}| = t - 1$ **then**
  $temp \leftarrow UPA_{i:} \setminus UPA'_{i:}$
  if $\exists j\ s.t.\ UPA_{j:} \supseteq temp, j \neq i$ **then**
    $r \leftarrow temp$
  else
    $r \leftarrow UPA_{i:}$
    $UA_{i:} \leftarrow \emptyset$
  end if
else
  $r \leftarrow argmax_{r \in CRoles} |UPA_{i:}\ s.t.\ UPA_i \supseteq r|$
end if
Update PA by including r;
Update UA by assigning r to all valid users;
Update UPA';

Algorithm 3 CGenerator

Input: UPA, UPA'
Output: $CRoles$
$CRoles \leftarrow \emptyset$;
for $i = 1 \rightarrow |UPA|$ **do**
  if $UPA_i \setminus UPA'_i \neq \emptyset$ **then**
    $CRoles \leftarrow \{CRoles, UPA_i \setminus UPA'_i\}$
  end if
end for

5.3 Computational Complexity Analysis

The key of the above user-oriented role mining algorithm is the continuous updating of candidate roles. At each iteration of the algorithm, a candidate role is chosen and the role coverage is determined. The total computations then depend on the number of iterations. Consider a user-permission dataset with m users and n permissions. According to our algorithm, at each iteration, at least one user's permissions are completely covered. So the maximum required iterations are m. At each iteration step, each candidate role is compared against each remaining user. As the number of candidate roles is less than m, the number of remaining users is less than m and each user (or role) has up to n permission, so the incurred computations at each iteration cannot be over m^2n. Therefore the computation complexity of our algorithm is upper bounded by m^3n.

6 Experiments and Results

Experiments are conducted on benchmark access control datasets. They are **americas_small**, **apj**, **healthcare**, **domino**, **firewall1** and **firewall2**, which can be found at the

HP website [2]. **americas_small** and **apj** are user profiles from Cisco firewalls. **healthcare** was obtained from the US Veteran's Administration. The **domino** graph is from a set of user and access profiles for a Lotus Domino server. **firewall1** and **firewall2** are results of running an analysis algorithm on Checkpoint firewalls. Descriptions on the data sets including the number of users, the number of permissions, and the size of user-permission assignments are given in Table 3. More detailed descriptions can be found in [2].

The first experiment evaluates the user-oriented exact RMP. We want to know whether our **Dynamic** algorithm can effectively enforce the sparseness constraint and whether the output of the algorithm is comparable to the optimal RBAC solution without the sparsity constraint. To find the answers, we run the **Dynamic** algorithm on those real data sets with different sparsity constraints. We compare our results with the benchmark role mining algorithm, **Lattice** [2]. As far as we know, **Lattice** has the best reported result with respect to the minimization of the number of roles and the minimization of the system administrative cost. The experimental results are reported in Tables 4 - 7. In these tables, δ denotes the error rate. The exact RMP requires the error rate to be 0. So we only look at the portion of the results with $\delta = 0$. Other parameters are: t denotes the maximum number of role assignments enforced in our algorithm, $|UA|$ denotes the size of user-permission assignments and $|PA|$ denoting the size of permission-role assignments. Note that δ and t has no effect on the **Lattice** algorithm, as **Lattice** returns an exact RBAC solution and the solution is unique. In the results, the value at the row of **Lattice** and the column of t is the maximum number of roles that a user has in the RBAC solution returned by the **Lattice** algorithm.

Data Set	$\|U\|$	$\|P\|$	$\|UPA\|$
healthcare	46	46	1,486
domino	79	231	730
firewall1	365	709	31,951
firewall2	325	590	36,428
apj	2,044	1,164	6,841
americas small	3,477	1,587	105,205

Table 3. Data Description

In the results, when t decreases, the size of UA decreases accordingly. However, the value of $|UA| + |PA|$ changes in an opposite direction. This matches our expectation. Specifically, when t is a small value, each user gets few role assignments. Thus, we need roles with more permissions, so each user can still get enough permission assignments. When the sparseness constraint becomes more strict, the number of required roles increases. As a result, the value of $|PA|$ increases accordingly.

Furthermore, we are pleased to see that even with the sparseness constraint being enforced, the complexity of the RBAC solution returned by **Dynamic** is still comparable to that of **Lattice**. For example, in Table 8, when t is 2, **Dynamic** returns a RBAC solution with only 18 roles, while **Lattice** returns a solution with 15 roles and the maxi-

[2] http://www.hpl.hp.com/personal/Robert_Schreiber/

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	0.00	2	90	365	7100
		4	85	454	6890
		6	84	600	6897
		8	80	1516	6638
	0.05	2	37	250	5688
		4	48	361	5685
		6	41	529	5523
		8	39	1416	5671
	0.10	2	30	250	4619
		4	27	330	4297
		6	26	439	3868
		8	14	1464	2970
	0.15	2	17	250	2661
		4	9	422	1762
		6	10	563	1810
		8	7	1334	2055
	0.20	2	11	250	1881
		4	8	426	1655
		6	6	1131	1839
		8	6	1131	1839
Lattice	0.00	9	66	874	1953

Table 4. fire1

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	0.00	2	259	3477	25229
		4	256	3722	23610
		6	249	3890	22194
		8	246	4269	20119
	0.05	2	224	3283	23174
		4	184	3566	21349
		6	183	3780	20109
		8	185	4160	18406
	0.10	2	205	3220	21421
		4	154	3453	19870
		6	157	3715	18451
		8	147	4042	16318
	0.15	2	180	3096	19929
		4	138	3383	17966
		6	137	3604	16630
		8	127	4036	14698
	0.20	2	171	3096	18505
		4	130	3384	16530
		6	123	3673	14971
		8	117	3915	14238
Lattice	0.00	10	192	4782	9830

Table 5. americas small

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	0.00	2	11	325	1499
	0.05	2	11	325	1499
	0.10	2	7	285	1092
	0.15	2	7	285	1092
	0.20	2	7	285	1092
Lattice	0.00	3	10	434	1110

Table 6. fire2

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	0.00	2	23	79	716
	0.05	2	17	71	695
	0.10	2	14	64	680
	0.20	2	10	53	657
Lattice	0.00	3	20	110	713

Table 7. domino

mum role assignments of 4. Another observation is that the $|UA|$ value of the solutions returned by **Dynamic** can be much less than that of the solutions returned by **Lattice**. For instance, in Table 5, the value of $|UA|$ for **Dynamic** with δ of 0 and t of 2 is 3477, while that for **Lattice** is 4782.

The second experiment is to study the user-oriented approximate RMP. We want to know how the RBAC solution varies with the error rate. We run the **Dynamic** algorithm by varying the value of δ from 0.05 to 0.20. Results are reported in Tables 4 - 7. We observe that when the complexity of the RBAC solutions decrease drastically when δ increases. For instance, in Table 4, with t of 8, only 39 roles are required to cover the 95 percent of permission assignments (i.e., $\delta = 0.05$), while 80 roles are required for the complete coverage (i.e., $\delta = 0$). In terms of the coverage of permission assignments, those 39 roles appear more promising than the remaining 41 roles. In cases where data noise is believed to exist, the approximate version of **Dynamic** appears to be more useful.

To summarize, the two experiments have demonstrated the effectiveness of our user-oriented RMP approach. We highlight that one primary advantage of **Dynamic** is that it allows a RBAC engineer to tune the sparsity constraint to reflect the real need. This feature is not supported by any existing role mining method. More importantly, the overall system complexity of the resultant solution is comparable to that of the optimal solution without any sparsity constraint.

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	0.00	2	18	46	545
		3	18	53	468
	0.05	2	5	46	191
		3	6	54	201
	0.10	2	5	45	191
		3	6	54	201
	0.15	2	5	45	191
		3	6	54	201
	0.20	2	5	45	191
		3	3	72	126
Lattice	0	4	15	106	209

Table 8. healthcare

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	0.00	2	564	2044	5565
		3	497	2218	5221
		4	485	2277	5096
	0.05	2	477	1664	4995
		3	394	1538	4362
		4	381	1571	4215
	0.10	2	407	1292	4449
		3	358	1449	4122
		4	352	1501	4043
	0.15	2	348	876	3878
		3	322	1258	3831
		4	317	1313	3766
	0.20	2	315	816	3609
		3	290	1064	3533
		4	289	1169	3550
Lattice	0.00	6	454	2437	4117

Table 9. apj

7 Conclusion

In this paper, we studied the role mining problem from the end-user perspective. Unlike other existing role mining approaches which primarily aim to reduce the administrative workload, our approach strives to incorporate better user experience into the role decision process. As end-users prefer simple role assignments, we add a sparseness constraint that mandates the maximum number of roles a user can have to the role mining process. The number usually can be determined in practice by a brief study on the general business processes of an organization. Basing on this rationale, we formulated user-oriented role mining as two specific problems. One is the user-oriented exact RMP, which is obliged to completely reconstruct given permission assignments while obeying the sparseness constraint. It is applicable for scenarios where the given dataset has no noise. The other is the user-oriented approximate RMP, which tolerates a certain amount of deviation from the complete reconstruction. It suits for datasets containing noises. We studied existing role mining methods, and found that some of them can be applied to our problems with simple modification. For better efficiency, we also designed new algorithms tailored to our problems, which are based on a dynamic candidate role generation strategy. Experimental results demonstrate the effectiveness of our approach in discovering a user-oriented RBAC solution while without increasing the overall administrative workload too much.

Future work can go along two directions. One is to study the feasibility of employing some statistical measures such as Bayesian information criterion to facilitate the role mining process. The motivation is that sometimes the accurate sparseness constraint (the maximum role that a user can have) is not available. We could employ some statistical criteria to choose the RBAC model with a good balance of model complexity and describability. The other direction is to consider the dynamic sparseness constraint. In this work, we assume that the same sparseness constraint is enforced to everyone. However, it might be the case that some user requires many role assignments due to some need. In such cases, a more practical role mining approach is to minimize the sparsity of the whole user-role assignments rather than enforcing a sparseness constraint for every user.

References

1. E. J. Coyne. Role engineering. In *RBAC '95: Proceedings of the first ACM Workshop on Role-based access control*, page 4, New York, NY, USA, 1996. ACM.
2. A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan. Fast exact and heuristic methods for role minimization problems. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 1–10, New York, NY, USA, 2008. ACM.
3. M. Frank, D. Basin, and J. M. Buhmann. A class of probabilistic models for role engineering. In *Proceedings of the 15th ACM conference on Computer and communications security*, 2008.
4. M. Frank, A. P. Streich, D. Basin, and J. M. Buhmann. A probabilistic approach to hybrid role mining. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 101–111, New York, NY, USA, 2009. ACM.
5. M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 179–186, New York, NY, USA, 2003. ACM.
6. H. Lu, J. Vaidya, and V. Atluri. Optimal boolean matrix decomposition: Application to role engineering. *IEEE 24th International Conference on Data Engineering*, pages 297–306, 2008.
7. H. Lu, J. Vaidya, V. Atluri, and Y. Hong. Extended boolean matrix decomposition. In *IEEE International Conference on Data Mining*, 2009.
8. H. Lu, J. Vaidya, V. Atluri, and Y. Hong. Constraint-aware role mining via extended boolean matrix decomposition. *Dependable and Secure Computing, IEEE Transactions on*, 9(5):655 –669, sept.-oct. 2012.
9. X. Ma, R. Li, and Z. Lu. Role mining based on weights. In *SACMAT*, pages 65–74, 2010.
10. I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining roles with semantic meanings. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 21–30, New York, NY, USA, 2008. ACM.
11. I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. B. Calo, and J. Lobo. Mining roles with multiple objectives. *ACM Trans. Inf. Syst. Secur.*, 13(4):36, 2010.
12. I. Molloy, N. Li, Y. A. Qi, J. Lobo, and L. Dickens. Mining roles with noisy data. In *Proceeding of the 15th ACM symposium on Access control models and technologies*, SACMAT '10, pages 45–54, New York, NY, USA, 2010. ACM.
13. G. Neumann and M. Strembeck. A scenario-driven role engineering process for functional rbac roles. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, SACMAT '02, pages 33–42, New York, NY, USA, 2002. ACM.
14. J. Schlegelmilch and U. Steffens. Role mining with orca. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 168–176, 2005.
15. J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: finding a minimal descriptive set of roles. In *SACMAT*, pages 175–184, 2007.
16. J. Vaidya, V. Atluri, Q. Guo, and H. Lu. Edge-rmp: Minimizing administrative assignments for role-based access control. *Journal of Computer Security*, 17(2):211–235, 2009.
17. J. Vaidya, V. Atluri, Q. Guo, and H. Lu. Role mining in the presence of noise. In *DBSec*, pages 97–112, 2010.
18. J. Vaidya, V. Atluri, and J. Warner. Roleminer: mining roles using subset enumeration. In *The 13th ACM conference on Computer and communications security*, pages 144–153, 2006.