Role Mining in the Presence of Noise

Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, Haibing Lu

Role Mining in the Presence of Noise

Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Haibing Lu

MSIS Department and CIMIC, Rutgers University, USA
{jsvaidya, atluri, qiguo, haibing}@cimic.rutgers.edu

Abstract. The problem of role mining, a bottom-up process of discovering roles from the user-permission assignments (UPA), has drawn increasing attention in recent years. The role mining problem (RMP) and several of its variants have been proposed in the literature. While the basic RMP discovers roles that exactly represent the UPA, the *inexact* variants, such as the δ-approx RMP and MinNoise-RMP, allow for some inexactness in the sense that the discovered roles do not have to exactly cover the entire UPA. However, since data in real life is never completely clean, the role mining process is only effective if it is robust to noise. This paper takes the first step towards addressing this issue. Our goal in this paper is to examine if the effect of noise in the UPA could be ameliorated due to the inexactness in the role mining process, thus having little negative impact on the discovered roles. Specifically, we define a formal model of noise and experimentally evaluate the previously proposed algorithm for δ-approx RMP against its robustness to noise. Essentially, this would allow one to come up with strategies to minimize the effect of noise while discovering roles. Our experiments on real data indicate that the role mining process can preferentially cover a lot of the real assignments and leave potentially noisy assignments for further examination. We explore the ramifications of noisy data and discuss next steps towards coming up with more effective algorithms for handling such data.

1 Introduction

Today, Role Based Access Control (RBAC) is the de facto model used for advanced access control, and is widely deployed in diverse enterprises of all sizes. Under RBAC, roles represent organizational agents that perform certain job functions within the organization. Users, in turn, are assigned appropriate roles based on their responsibilities and qualifications [1, 2]. Essentially, a role, can be viewed as a set of permissions. One main benefit of RBAC is simplified security administration as the role configuration need not be changed when users join or leave the organization.

Deploying RBAC requires first defining a complete and correct set of roles. This process, known as *role engineering* [3], has been identified as one of the costliest and most time consuming components in realizing RBAC [4]. The problem of role mining, a bottom-up process of discovering roles from the user-permission assignments (UPA), has drawn increasing attention in recent years.

Unlike the top-down approach where roles are defined by carefully analyzing and decomposing business processes into smaller units in a functionally independent manner, role mining has the advantage of automating the role engineering process. Role mining can be used as a tool, in conjunction with a top-down approach, to identify potential or candidate roles that can then be examined to determine if they are appropriate, given existing functions and business processes.

Work has been carried out on the top-down approach to role engineering[3, 5, 6], though it requires significant manpower effort to do correctly. To alleviate this, there have been several attempts to propose good bottom-up techniques to finding roles [7, 8]. More recently, researchers [9–15] have begun formally defining the role mining problem (RMP) and proposed a number of RMP variants. There has also been some work on hybrid mining of roles[16, 17]. The basic-RMP problem has been shown to be equivalent to a number of known problems including matrix decomposition and minimum biclique cover problem in graphs, among others. The basic-RMP as a matrix decomposition problem is as follows. Given m users, n permissions, then UPA can be represented as an $m \times n$ boolean matrix $M(UPA)$ where a 1 in cell $\{ij\}$ indicates the assignment of permission j to user i. If there are k roles, the user-to-role mapping (UA) can be represented as an $m \times k$ boolean matrix $M(UA)$ where a 1 in cell $\{ij\}$ indicates the assignment of role j to user i. Similarly, the role-to-permission mapping (PA) can be represented as a $k \times n$ boolean matrix $M(PA)$ where a 1 in cell $\{ij\}$ indicates the assignment of permission j to role i. The basic-RMP is to decompose $M(UPA)$ into a $m \times k$ matrix $M(UA)$ representing UA and a $k \times n$ matrix $M(PA)$ representing PA, such that k is minimal.

Apart from basic-RMP, other role mining problems have been defined with different minimization objectives. One objective is to discover roles in such a way that the total number of user-to-role assignments and role-to-permission assignments ($|UA| + |PA|$) is minimal. This, known as the Edge-RMP, has been studied by Vaidya et al. [9] and Ene et al. [12], among others. Other variants [14, 10] focus on discovering optimal roles as well as role hierarchies.

While the above RMP problems attempt to discover roles that *exactly* describe the original UPA, this may be unnecessary. With this in mind, some *inexact* variants of RMP have been identified, which do not exactly describe the original UPA, but allow for some inexactness. This is justified since mistakes can always be made, and asking for roles to match those mistakes would actually worsen the situation. Moreover, in some sense, it may be possible to find more "fundamental roles" by only trying to match a certain percentage of the original UPA. To describe this. Vaidya et al. [18] have proposed two inexact variants of RMP, called the δ-consistent RMP and MinNoise RMP.

Any given UA, PA, and UPA are considered to be δ-consistent, if and only if the UPA derived from UA and PA matches the original UPA within δ. Now, the δ-approx RMP can be defined [18] as the problem of finding the minimal set of roles such that the discovered UA and PA and the original UPA are δ-*consistent*. As a problem formulation, the δ-approx Role Mining Problem is *always* better than the Basic Role Mining Problem when noise is present. This

Table 1. User-Permission Assignment

(a) Without Noise

	p_1	p_2	p_3	p_4	p_5
u_1	1	1	1	0	0
u_2	0	0	1	0	1
u_3	1	1	1	0	1
u_4	1	1	1	0	0

(b) With Noise

	p_1	p_2	p_3	p_4	p_5
u_1	1	1	1	1	1
u_2	0	0	1	0	1
u_3	0	1	1	0	1
u_4	1	1	1	1	0

is so, because, setting a nonzero value of δ gives the leeway to correct for the present noise errors. In this sense, any UPA divergent within δ from the original UPA satisfies the δ-approx definition. Obviously, all such formulations are not equally good. A good algorithm gives a UPA' that preferentially removes noise rather than making errors.

Another approach for inexactness could be by bounding the number of roles, and minimizing the approximation. This is the Minimal Noise Role Mining Problem (MinNoise RMP). This fixes the number of roles that one would like to find, but finds those roles that incur minimal difference with respect to the original user-permission matrix (UPA).

Since data in real life is never completely clean, the role mining process is only effective if it is robust to noise. By noise, we mean a permission being recorded as a denial or a denial being recorded as a permission. Our goal in this paper is to examine if the effect of noise in the UPA could be ameliorated due to the inexactness in the role mining process, thus having little negative impact on the discovered roles.

In the following, we provide a concrete example to justify our argument. Table 1(a) shows a sample user-permission assignment (UPA), for 4 users and 5 permissions. Table 1(b) shows the UPA with noise. Specifically, user u_1 has the extra permissions p_4, p_5, user u_3 has lost the permission p_1, while user u_4 gains the extra permission p_4 (i.e., cells (u_1, p_4), (u_1, p_5), and (u_4, p_4) are flipped from 0 to 1, while cell (u_3, p_1) is flipped from 1 to 0). This may happen due to the security administrator forgetting to remove old permissions that are no longer necessary (in the 0 to 1 case), or by mistake temporarily revoking a necessary permission (in the 1 to 0 case). Tables 2(a) and 2(b) depict a user-role assignment (UA) and role-permission assignment (PA) that completely describe the given (noisy) user-permission assignment (i.e., $M(UA) \otimes M(PA) = M(UPA)$). Indeed, the given UA, PA, and $ROLES$ are optimal. It is not possible to completely describe the given UPA with less than 3 roles. As one can see, Table 2, representing the basic-RMP approach, actually matches exactly reconstructs the input noisy UPA and in effect, matches all 4 noisy bits as well.

On the other hand, Tables 3(a) and 3(b) depict the optimal UA and PA, which cover none of the noisy bits and *accurately* cover (without introducing errors) all of the 1's. In fact, as one can see, this is actually the optimal decomposition of the original (clean) UPA, which unfortunately is never seen. This clearly shows that the optimal solution for basic-RMP is not always a better choice when noise is present.

Table 2. Optimal role set for Basic-RMP over noisy data

(a) UA

	r_1	r_2	r_3
u_1	1	1	1
u_2	0	1	0
u_3	1	1	0
u_4	1	0	1

(b) PA

	p_1	p_2	p_3	p_4	p_5
r_1	0	1	1	0	0
r_2	0	0	1	0	1
r_3	1	0	0	1	0

Table 3. Optimal role set for Basic-RMP over Clean data

(a) UA

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

(b) PA

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

In fact, noise can cause multiple problems in the process of role mining. First, as shown above, we may get a suboptimal (i.e., larger) number of roles. Worse, the discovered role set reconstitutes noisy bits. Since the discovered roles are also contaminated with noise, they perpetuate the existing errors. For example, in Table 2 role r_3 consists of $\{p_1, p_4\}$. Since bit p_4 is actually noisy, in effect any user who will now be assigned r_3 will also have the same error. Thus, due to the noise, not only is the wrong UPA reconstituted, but also, the error affects future users who may be mistakenly given over and underpermissions right from the start. This can create great problems and reduce the benefits of using RBAC in the first place.

In this paper, first we define a formal model of noise and then experimentally evaluate the previously proposed algorithm for the δ-approx RMP. Essentially, this would allow one to come up with strategies to minimize the effect of noise while discovering roles. Our experimental results indicate that if one sets the level of δ equal to the level of noise, the impact of noise is minimal in discovering roles. We evaluate the robustness of the algorithm using the standard F-score measure. Our results indicate that, under the presence of noise, the F-score for δ-approximate RMP is fairly uniform with increasing noise levels, whereas the F-score for basic-RMP deteriorates.

The rest of this paper is organized as follows. In Section 2, we review the RBAC model and the formal definitions of δ-approx RMP, and MinNoise RMP. In Section 3, we present an appropriate noise model, which include the definition of noise, degree of noise and the robustness measure. In Section 4, we illustrate the δ-approx RMP. In Section 5 we discuss our noise robustness evaluation approach. In Section 6, we present the results of our experimental evaluation of the δ-approx RMP algorithm against noise robustness. In Section 7 we provide an

insight into alternative strategies for reducing the effect of noise. Finally, Section 8 concludes the paper.

2 Preliminaries

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

2.1 Role Based Access Control Model

We adopt the NIST standard of the Role Based Access Control (RBAC) model [2]. For the sake of simplicity, we do not consider sessions, role hierarchies or separation of duties constraints in this paper.

Definition 1 (RBAC).

- $U, ROLES, OPS$, *and* OBJ *are the set of users, roles, operations, and objects.*
- $UA \subseteq U \times ROLES$*, a many-to-many mapping user-to-role assignment relation.*
- $PRMS$ *(the set of permissions)* $\subseteq \{(op, obj) | op \in OPS \bigwedge obj \in OBJ\}$
- $PA \subseteq ROLES \times PRMS$*, a many-to-many mapping of role-to-permission assignments.* [1]
- $UPA \subseteq U \times PRMS$*, a many-to-many mapping of user-to-permission assignments.*
- $assigned\_users(R) = \{u \in U | (u, R) \in UA\}$*, the mapping of role* R *onto a set of users.*
- $assigned\_permissions(R) = \{p \in PRMS | (p, R) \in PA\}$*, the mapping of role* R *onto a set of permissions.*

2.2 The Inexact RMP Variants

We review the formal definitions [18] of the δ-approx RMP and the MinNoise RMP as these inexact RMP variants are the focus of this paper.

Definition 2 (δ-approx RMP). *Given a set of users* U*, a set of permissions* $PRMS$*, a user-permission assignment* UPA*, and a threshold* δ*, find a set of roles,* $ROLES$*, a user-to-role assignment* UA*, and a role-to-permission assignment* PA*,* δ*-consistent with* UPA *and minimizing the number of roles,* k*.*

The basic-RMP is the case where $\delta = 0$ [18]. Given the user-permission matrix (UPA), the basic-RMP is to find a user-to-role assignment UA and a role-to-permission assignment PA such that UA and PA describe UPA while minimizing the number of roles. The notion of δ-consistency is useful, since it helps to bound the degree of approximation. The MinNoise RMP bounds the number of roles, which is defined as follows.

Definition 3 (MinNoise RMP). *Given a set of users* U*, a set of permissions* $PRMS$*, a user-permission assignment* UPA*, and the number of roles* k*, find a set of* k *roles,* $ROLES$*, a user-to-role assignment* UA*, and a role-to-permission assignment* PA*, minimizing* $\| M(UA) \otimes M(PA) - M(UPA) \|_1$ *where* $M(UA), M(PA)$*, and* $M(UPA)$ *denote the matrix representation of* UA*,* PA *and* UPA*.*

[1] Note that in the original NIST standard [2], PA was defined as $PA \subseteq PRMS \times ROLES$, a many-to-many mapping of permission-to-role assignments.

3 Noise

In reality, no data is clean, and the user-permission-assignment is no exception. Permissions are accidentally assigned to those who do not need them, or are not revoked once the permission is no longer needed. In addition, some users may not have all the permissions that others performing similar job functions have. Since noise in general is random in nature, we believe that using inexact variants of the RMP in discovering roles could use this noise to its benefit and may have little impact on the outcome of the role discovery process. In this section, we first present our noise model [19] that helps in evaluating the δ-approx RMP algorithms against their robustness to noise. Second, we present the degree of noise [19]. We now discuss each in detail.

3.1 Noise Model

The presence of noise essentially means that errors have occurred in the data – i.e., the actual data does not match the real data. In our case, the data consists of access permissions in the user-permission-assignment matrix (UPA). In this boolean matrix, a 1 signifies that the subject-object access is permissible (we denote this as allowed permission), and a 0 denotes lack/denial of permission (we denote this as disallowed permission). In this case, noise means that the actual boolean matrix is different from the desired boolean matrix with the following three types of errors occurring in the matrix:

1. **General noise:** Such noise results in bit-flipping errors. Thus, a 1 gets flipped to a 0 and vice-versa. Effectively, this means that either a permission is incorrectly revoked or a permission is incorrectly given by the security administrator.
2. **Additive noise:** In this case, a permission can only be incorrectly given, not incorrectly revoked. Thus, a 0 can incorrectly be changed to a 1 but not vice-versa. This could happen if an administrator had first given a permission to a user to accomplish some task, but then forgotten to revoke it after the task/duration is complete.
3. **Subtractive noise:** In this case, a permission could be incorrectly revoked, though not incorrectly given. Thus, a 1 is incorrectly changed to a 0, but not vice-versa. This could happen when a user is only given a subset of the overall permissions he may ultimately need. For example, when someone new starts in the organization, he may be given a set of permissions for some initial assignments but not the full set he will ultimately need because accurate assignment is time consuming.

It is clear that general noise actually includes both additive noise and subtractive noise. Thus, the presence of general noise implies the presence of both additive as well as subtractive noise. However, their percentages are not equal. In actuality, the degree of additive and subtractive noise depends on the number of 0s and 1s. All else being equal, any general noise will result in additive noise proportional to the number of 0s and subtractive noise proportional to the number

of 1s. Typically, the access control matrix will be sparse with fewer 1s and many more 0s. Therefore, additive noise will be more likely to occur. This corresponds to real situations where users are more likely to have more permissions than they need (additive noise) than less permissions than they need (subtractive noise) because otherwise they could not perform their job functions. For now, we explore the robustness of our algorithm to additive noise since we believe it is the type of noise we are most concerned.

One may argue that errors do not happen completely at random, and are predetermined based on actual usage. However, when there is no RBAC in place or preexisting policies (which could easily happen when you are freshly deploying RBAC), the random model of noise does accurately reflect reality.

3.2 Degree of Noise

To effectively take noise into account, one must consider the degree of noise along with the different types of noise discussed above. An obvious way to consider the degree of noise is computing it as a percentage of the amount of data. However, considering this percentage of noise for the entire number of bits in UPA may not be an accurate representation of noise. For example, consider a system with 2000 users and 500 permissions. This results in the UPA of 1,000,000 bits. 1% of this dataset equates to 10000 bits. However, it does not mean that 10000 bits should be flipped. This is because of the fact that the UPA is typically very sparse, i.e., the number of allowed permissions (1s) is significantly smaller than the number of disallowed permissions (0s). For example, only 4000 of the subject-object accesses might be allowed. In this case, flipping 10000 bits would completely obviate the true data. One must realize that unlike digital communications, in access control data, the signal is characterized only by the 1s, and not by the 0s. This implies that when considering the degree of noise, it should be a percentage of the number of 1s in the data. Thus, when considering noise percentages, we take noise to be a percentage of the number of 1s.

Finally, we need to consider how to add noise to the data. Again, assume 2000 users, 500 permissions and 4000 allowed subject-object pairs (4000 1s). If we wanted to introduce 10% general noise into this data set, how should we proceed? A simple way to add noise is to pick 400 bits at random from the 1,000,000 bits and flip them. But is this correct – should exactly *400* bits be flipped? The key issue is whether by 10% noise we mean that the noise is exactly 10% of the data or whether we mean that the probability of error in the data is 10%. The first case corresponds to flipping 400 bits. However, in the second case, we should go through all 1,000,000 bits and flip each bit with a probability of 0.04% (This is correct since 400 is actually 0.04% of 1000000). Though either way is fine, we argue that the second way of introducing noise more closely approximates real life.

In our experiments we considered 1%, 5%, 10%, 20% and 30% noise and introduced it into the datasets as described in the second method above. As a result of the discussion in the prior two sections, we define noise as follows:

Definition 4. *[19] p% General Noise: Given users U and permissions $PRMS$ and a user to permission assignment $UPA \subseteq U \times PRMS$, a noisy dataset with p% general noise consists of a modified permission assignment $UPA' \subseteq U \times PRMS$ such that: (i) if $x \in UPA$, $x \in UPA'$ with probability $1-p$; (ii) if $x \notin UPA$, $x \in UPA'$ with probability p.*

4 Algorithm

As discussed earlier, Vaidya et al.[18] first formalized the Role Mining Problem and defined the δ-approx RMP. However, no solution was proposed for it. In [20], a heuristic solution has been proposed that can efficiently solve both the RMP as well as the δ-approx RMP. We now describe this solution.

The algorithm proceeds in two independent phases. In the first phase, a set of candidate roles is generated from the UPA. This is currently done using the FastMiner algorithm[19], though any other method could also be used instead. FastMiner generates candidate roles simply by intersecting all unique user pairs. Once the candidates are generated, the second phase greedily picks the best set of roles among them by picking the best candidate role from the remaining candidate roles until the original UPA can be reconstituted within the desired approximation factor (δ). Thus, in each iteration, for every remaining candidate role, the uncovered area of that role is computed by finding the number of 1s in $M(UPA)$ that are not already covered by any of the roles in $ROLES$ (the current minimum tiling). The best role is then selected and the algorithm reiterates until termination. Algorithm 1 gives the details.

Algorithm 1 δ-approx RMP(UPA, δ)

Require: User-Permission assignment, UPA
Require: the approximation threshold, δ
{Create candidate set of roles}
Create a candidate set of roles, $CROLES$, using the FastMiner [19] algorithm
$ROLES \leftarrow \phi$
while UPA is not covered within δ **do**
  $BestRole \leftarrow \phi$
  $BestArea \leftarrow 0$
  for each role C in $CROLES$ **do**
    $carea \leftarrow Uncovered\_Area(C, UPA, ROLES)$ {compute uncovered area of candidate role}
    if $carea > BestArea$ **then**
      $BestArea \leftarrow carea$
      $BestRole \leftarrow C$
    end if
  end for
  $ROLES \leftarrow ROLES \bigcup C$ {Add C to the set of roles, $ROLES$}
  Remove C from $CROLES$
end while
Return $ROLES$

It is important to note that the way in which candidate roles are generated (through subset enumeration), and the way the uncovered area is computed, plays a role in the kinds of the errors the algorithm can tolerate. Specifically, when reconstituting the UPA, the above algorithm can leave 1s uncovered

though it does not cover 0s with 1s. What this means is that the algorithm could potentially eliminate additive noise (by ignoring the added 1s), but cannot handle subtractive noise (since a 1 that has been turned into a 0 is never reconstituted). In general, since additive noise is much more of a vulnerability from the security standpoint, and typically the UPA is sparse, this still may be acceptable in real life. However, ideally one would prefer to have algorithms that are robust to both kinds of noise.

5 Noise Robustness Metric

Robustness, in essence, reflects the degree to which an algorithm is *not* affected by noise. First, we define a *noisy* bit as a bit whose value has been erroneously flipped. We also use the term non-noisy bit to describe those bits whose values are not changed by mistake. A *completely robust* algorithm will somehow be able to magically eliminate the effect of noise. Thus, the output of a "completely robust" algorithm would be the same regardless of the presence of noise. In effect, a perfect algorithm would cover all of the original 1s and cover none of the original 0s, regardless of their value after the introduction of noise. In order to compare noise robustness of different algorithms, and to devise better algorithms, it is extremely important to appropriately define a metric for noise robustness.

A naïve approach to define the degree of robustness is to define it as ratio of the number of appropriately reconstituted bits (noisy or non-noisy bits included) to the total number of bits in UPA. For example, from Table 1(a) and Table 1(b), we can see the total number of bits in UPA are 20, among which, 4 bits are noisy: (u_1, p_4), (u_1, p_5), (u_3, p_1), and (u_4, p_4). The solution in Tables 3(a) and 3(b) reconstitutes the original values for all 4 noisy bits and all 16 non-noisy bits. Therefore, the degree of robustness of the algorithm, as defined above, is 20/20=100 percent. On the contrary, the solution obtained for the Basic-RMP (depicted in Tables 2(a) and 2(b)) does not reconstitute the original values of any of the noisy bits, and therefore, has a degree of robustness of 16/20=0.8. The problem with this way of measurement is that it does not differentiate between good performance on the original data versus good performance with respect to noise. Thus, the solution for the Basic-RMP still has 0.8 robustness even though it does not fix any of the errors associated with noise. In effect, it has 0 tolerance to noise, though it reconstitutes the original non-noisy data perfectly.

Another naïve approach to measure the degree of noise robustness is to define it as the ratio of the number of noisy bits whose original (before-flipping) values are reconstituted to the total number of noisy bits. In other words, it measures the relative ratio of appropriately covered noisy bits. To use the same example as above, the solution in Tables 3(a) and 3(b) reconstitutes the original values for all 4 noisy bits. Therefore, the degree of robustness of the algorithm is 100 percent. On the contrary, The Basic-RMP algorithm in Tables 2(a) and 2(b), does not reconstitute the original values of any noisy bit, therefore, having a degree of robustness as 0.

While this accurately measures how much noise is fixed, it does not take errors made by the algorithm into consideration. For example, the algorithm may cover a bit with 1 when its real value is 0 (though the value of the bit has not changed due to noise). Similarly, the algorithm may leave a bit uncovered as 0 when its real value is 1. Taking our original example into consideration, assume that a candidate algorithm almost reconstitutes the original UPA, but with only one error, say the value of a non-noisy bit (u_2, p_4) is 1 in the reassembled UPA. Under the above definition of degree of robustness, the algorithm would be considered to be 100 percent robust, which is clearly incorrect.

Indeed, the term *noise* can be used to describe the defect of the dataset, while the term *error* can be used to indicate the misjudgement made by the algorithm. A good definition of degree of noise robustness should measure both noise fixed as well as errors made by an algorithm. For this we must measure two sets of numbers – the percentage of correctly reconstituted noisy bits, as well as the percentage of errors made by the algorithm. We must also devise a way of integrating these into a unifying measure.

For this, work from the field of data mining and statistics comes to our aid. In a general classification problem, every data item must be classified to belong to a certain class. In a statistical classification task[21], the *Precision* for a class is the number of true positives divided by the total number of elements labeled as belonging to the class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). *Recall* is correspondingly defined as the number of true positives divided by the total number of elements that actually belong to the class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to that class but should have been). In this sense, Precision can be seen as a measure of exactness or fidelity, whereas Recall is a measure of completeness. The F-measure or balanced F-score is the harmonic mean of precision and recall: $F = 2 \cdot (\text{precision} \cdot \text{recall})/(\text{precision} + \text{recall})$.

Assuming only two classes, every instance originally belongs to one of these two classes and is classified into one of the two classes. From our perspective, the amount of noise fixed approximately corresponds to the notion of *Recall*, while the number of errors made approximately corresponds to the notion of *Precision*. Thus, we simply use the notion of F-score to check the overall degree of noise robustness of the algorithm.

6 Experimental Evaluation

The purpose of the experimental evaluation performed in this section is to compare the robustness of various role mining algorithms towards the noise. In Section 6.1, we discuss the design of the experiments and the their results are analyzed in Section 6.2.

6.1 Experimental Design

The experimental design consists of four steps: Data Creation, Noise Insertion, roles generation and UPA reconstitution, finally the computation of the degree of noise robustness. We will illustrate each step in separate sections.

Data Creation The test data generator performs as follows: First a set of roles are created. For each role, a random number of permissions up to a certain maximum are chosen to form the role. The maximum number of permissions to be associated with a role is set as a parameter to the algorithm. Next, the users are created. For each user, a random number of roles are chosen. Again, the maximum number of concurrent roles a user can have is set as a parameter to the algorithm. finally, the user permissions are set according to the roles the user has been assigned. We will also consider certain cases where the number of roles randomly chosen is 0 indicating that the user has no roles and therefore no permissions.

Noise Insertion Just as described in the Section 3.2, we take noise to be a percentage of the number of 1s. In our experiments we considered 1%, 5%, 10%, and 20% noise and introduced it into the datasets.

Reconstitution of UPA To test noise robustness, we apply a role mining algorithm to the data contaminated with noise. As a result, a role set is generated followed by the reconstitution of the original UPA.

Computation of Robustness The final step to consider is the computation of the noise robustness of the algorithm. We will use the way to do this based on Section 5.

Another way to do this is simply compute how many of the original roles are found in the results. This approach seems fine but actually suffer two serious weakness. (1) when we match roles, the matches are exact. While this is fine when there is no noise, in the presence of noise there is a good possibility that we may find approximate roles rather than the real roles. In this case, we should also calculate the pseudo-accuracy. This could be an important factor affecting the overall accuracy of the algorithm. However, for now, we restrict ourselves to exact matches and report the results obtained. In the future, we plan to see if approximate matching can lead to better results. (2) the percentage of common roles between original role set and newly generated role set is not a good indicator of robustness towards noise. There are various factors which could affect this value. For example, different algorithms may result in different way of matrix decomposition. They can be right at the same time and totally disjoint. This does not mean that the algorithm is not robust or at least not 0 percent robust.

6.2 Experimental Results

For an initial set of experiments, we created two datasets, the first with 100 users and 200 permissions, while the second was composed of 200 users and 200 permissions. Since the test data was randomly generated and the noise was also randomly inserted, we actually created three versions of the datasets with the same parameters. For each of these datasets, a corresponding noisy dataset

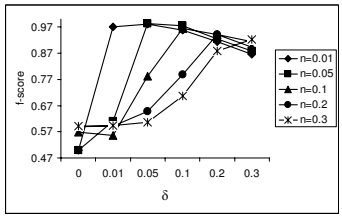

Fig. 1. Relative performance on dataset 1

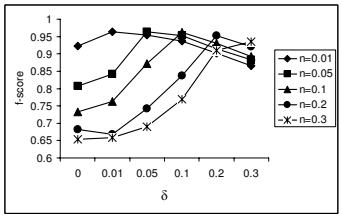

Fig. 2. Relative performance on dataset 2

was also created three times with the same set of noise parameters. The final results reported below were averaged over the 9 datasets generated for each set of parameters.

Figure 1 plots the f-score against different values of δ for different levels of noise for dataset 1. Figure 2 does the same for dataset 2. It can be observed that for every noise value, the f-score hits the peak at the corresponding value of δ. This implies that the δ-approx RMP algorithm does indeed perform very well with respect to noise. Furthermore, if the security administrator has some idea of the amount of noise present within the data, he can set the δ value to be that to provide great benefit during the role mining phase. Interestingly, when noise is present any $\delta > 0$ and up to that noise level is always beneficial. However, when the value of δ is set higher than the amount of noise present, the f-score decreases, since the algorithm starts making more errors without providing as much benefit.

Figures 3 and 4 plot the ratio of noise fixed / errors made by the algorithm for different δ values and different noise levels. For reasonable levels of noise, the fix/mistake ratio is always over 1, which implies that for any δ value more noise will be fixed as compared to the errors made. When the δ value is the same as the amount of noise, the fix/mistake ratio is very high, and using the inexact RMP has great benefit.

Finally, figure 5 plots the performance of the basic-RMP algorithm versus the δ-approx RMP algorithm for different levels of noise. This clearly shows the degradation of the basic-RMP w.r.t noise. However, the δ-approx RMP keeps its effectiveness even in the presence of noise. This supports our thesis that it makes great sense to use δ-approx RMP whenever noise may be present in the data. This works especially well when the degree of noise can be correctly estimated, but still gives some benefit when this cannot be reliably done. One of our future challenges is to come up with ways to estimate the degree of noise reliably which would greatly increase the effectiveness of noise-resistant algorithms.

6.3 Experiments with real data

We also carried out experiments with real data. For this, we utilized the 9 real datasets presented in [12]. Since it is impossible to find out whether any of the user-permission assignments was noisy, we instead simply ran δ-approx RMP

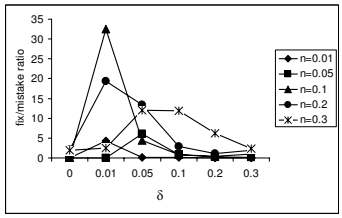

Fig. 3. Ratio of errors fixed to errors made for dataset 1

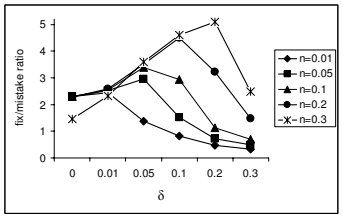

Fig. 4. Ratio of errors fixed to errors made for dataset 2

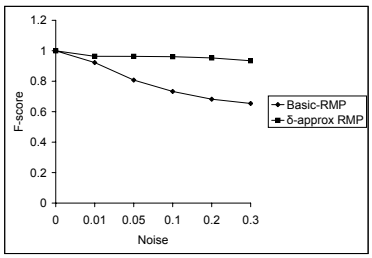

Fig. 5. Noise tolerance benefits of δ-approx RMP

algorithm on all of the datasets with different values of δ and counted the number of roles necessary to cover the UPA upto that threshold. We also measured the number of roles necessary to completely cover the entire UPA. Figures 6-7 plot the percentage of roles necessary to cover a certain percentage of the UPA. As can be seen, for almost all of the datasets (except *emea*), only 40% of the roles are necessary to cover 90% of the UPA. Indeed, the curve is non-linear, indicating that more roles are necessary to cover smaller parts of the UPA as the coverage increases. This can be clearly seen from Figures 8-9 which expand the portion depicting 80% to 100% coverage of the UPA. While we cannot confirm this, it is indeed an indication that the remaining UPA assignments are more likely to be noisy, or at the least, should be individually examined by the security administrator.

7 Discussion

So far, we have seen that using the inexact variants of the RMP problem can give signficantly better results in the presence of noise. Our experimental results show, that the δ-approx RMP algorithm of [20] can handle additive noise quite well, though it is unable to account for subtractive noise. Given the sparsity of the UPA in real life, this is still ok since noise is likely to be more additive than subtractive. Other solutions for δ-approx RMP may be able to give even better performance. In the following, we provide more insight into how to choose δ and discuss other noise removal strategies that we plan to explore in future work, which can further help in the role mining process.

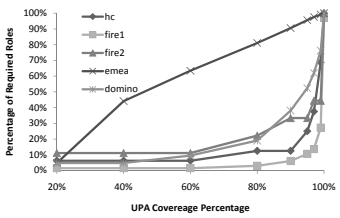

Fig. 6. Percentage of roles necessary to cover partial dataset

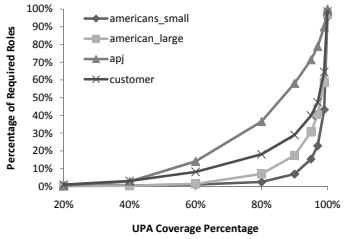

Fig. 7. Percentage of roles necessary to cover partial dataset

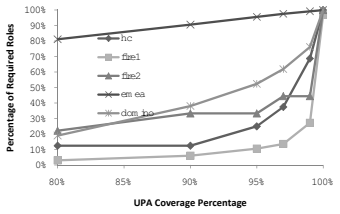

Fig. 8. Expanded part of Figure 6

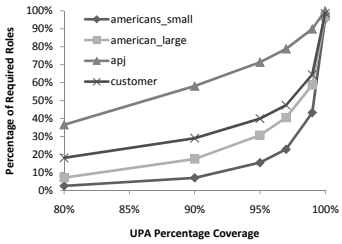

Fig. 9. Expanded part of Figure 7

7.1 Determining the right value for δ

So far, we have assumed that we could set the right value of δ for the δ-approx RMP algorithm. However, in reality, we may have little idea about this. Given that our δ-approx RMP uses a greedy strategy to pick roles, we could use that to adaptively define δ. Essentially, instead of setting δ from the start, the idea would be to iteratively pick the best role from among the candidates and then evaluate the quality of the role to determine whether to stop. Here, we use the role itself to help determine the right δ value. This is also possible because of the nature of access control data and the nature of noise. Access control data is tightly coupled whereas we assume that noise may occur at random. Thus, it is possible that later roles increasingly cover noisy bits as opposed to real data, and can be detected due to their heterogeneity. In any case, since using the inexact variant always does better in the presence of noise, an effective strategy could be to start with a fixed value for δ (say 0.7, since there is unlikely to be more than 30% noise in the data) and then incrementally increase the value of δ, while paying more attention to the roles obtained in latter phases. We will explore this further in the future.

7.2 Noise Removal by prefiltering

An alternative strategy to handling noise is to somehow remove it to generate a "clean" UPA. Now, some Basic-RMP algorithm can be run over this cleaned UPA to give the final roles. Then, the question is how can we detect and clean

noise. An answer may come from the field of data mining. It is possible to model each permission as a class and then to build a classification model for it based on the values of the other permissions for all of the users. We can then use all of the created models to predict whether each individual bit is truly correct or noisy. The performance of this inherently depends upon the quality of the classification algorithm, and having some user input would significantly help. We will also further explore this in the future.

7.3 Noise Removal by post-filtering

Another alternative strategy is to use the Basic-RMP algorithm on the given UPA. Now, we can use post-filtering strategies on the discovered UA and PA to somehow remove noise. This may again be done through use of classification strategies, or by evaluating how homogeneous each role is in terms of permissions or users. Domain semantics may vastly help to identify intelligent strategies to filter noise.

8 Conclusions

Given the noisy nature of data in the real world, deployment of role mining algorithms requires effective ways of addressing the problem of noise. In this paper, we take a first look at the problem of role mining in the presence of noise. We demonstrate the many problems noise can cause if it is not accounted for within the role mining process. We present a model for noise, devise metrics to evaluate noise robustness and investigate the effectiveness of inexact variants of the Role Mining Problem in terms of noise. Our preliminary experiments show that the algorithm developed for δ-approx RMP does indeed reduce the effect of noise. However, it is only able to handle additive noise. In the future, we plan to develop more complete noise aware algorithms that can effectively reduce or eliminate the problem of noise from the role engineering process. Algorithms for the MinNoise RMP can help with this. We will also examine whether approaches from data mining could be used to "clean" the noise from the user-permission data in the first place, thus allowing us to use any basic-RMP algorithm for discovering roles.

References

1. Sandhu, R.S., et al.: Role-based Access Control Models. IEEE Computer (February 1996) 38–47
2. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramouli, R.: Proposed NIST Standard for Role-Based Access Control. TISSEC (2001)
3. E.J.Coyne: Role-engineering. In: 1st ACM Workshop on Role-Based Access Control. (1995)
4. Gallagher, M.P., O'Connor, A., Kropp, B.: The economic impact of role-based access control. Planning report 02-1, National Institute of Standards and Technology (March 2002)

5. Roeckle, H., Schimpf, G., Weidinger, R.: Process-oriented approach for role-finding to implement role-based security administraiton in a large industrial organization. In ACM, ed.: RBAC. (2000)
6. Neumann, G., Strembeck, M.: A scenario-driven role engineering process for functional rbac roles. In: 7th ACM Symposium on Access Control Models and Technologies. (June 2002)
7. Schlegelmilch, J., Steffens, U.: Role mining with orca. In: Symposium on Access Control Models and Technologies (SACMAT), ACM (June 2005)
8. Vaidya, J., Atluri, V., Warner, J.: Roleminer: mining roles using subset enumeration. In: CCS '06: Proceedings of the 13th ACM conference on Computer and communications security. (2006) 144–153
9. Vaidya, J., Atluri, V., Guo, Q., Lu, H.: Edge-rmp: Minimizing administrative assignments for role-based access control. Journal of Computer Security **17** (2009) 211–235
10. Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S., Lobo, J.: Mining roles with semantic meanings. In: SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies, New York, NY, USA, ACM (2008) 21–30
11. Zhang, D., Ramamohanarao, K., Ebringer, T.: Role engineering using graph optimisation. In: SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies, New York, NY, USA, ACM (2007) 139–144
12. Ene, A., Horne, W., Milosavljevic, N., Rao, P., reiber, R.S., Tarjan, R.: Fast exact and heuristic methods for role minimization problems. In: The ACM Symposium on Access Control Models and Technologies. (June 2008)
13. Vaidya, J., Atluri, V., Guo, Q., Adam, N.: Migrating to optimal rbac with minimal perturbation. In: The ACM Symposium on Access Control Models and Technologies. (June 2008)
14. Guo, Q., Vaidya, J., Atluri, V.: The role hierarchy mining problem: Discovery of optimal role hierarchies. In: Proceedings of the 24th Annual Computer Security Applications Conference. (December8-12 2008) 237 – 246
15. Lu, H., Vaidya, J., Atluri, V.: Optimal boolean matrix decomposition: Application to role engineering. In: IEEE International Conference on Data Engineering. (April 2008)
16. Frank, M., Streich, A.P., Basin, D., Buhmann, J.M.: A probabilistic approach to hybrid role mining. In: CCS '09: Proceedings of the 16th ACM conference on Computer and communications security, New York, NY, USA, ACM (2009) 101–111
17. Fuchs, L., Pernul, G.: Hydro - hybrid development of roles. In Sekar, R., Pujari, A.K., eds.: ICISS. Volume 5352 of Lecture Notes in Computer Science. (2008) 287–302
18. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: Finding a minimal descriptive set of roles. In: The Twelth ACM Symposium on Access Control Models and Technologies, Sophia Antipolis, France (June20-22 2007) 175–184
19. Vaidya, J., Atluri, V., Warner, J., Guo, Q.: Role engineering via prioritized subset enumeration. IEEE Transactions on Dependable and Secure Computing (to appear)
20. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: A formal perspective. ACM Transactions on Information Systems Security (to appear)
21. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley-Interscience Publication (2000)