Towards Automated Assistance for Mined Roles Analysis in Role Mining Applications

Safaà Hachana*‡, Frédéric Cuppens†, Nora Cuppens-Boulahia*†, and Joaquin Garcia-Alfaro†
*Swid Web Performance Service, Rennes, France, Email: safa@swid.fr
†Institut Telecom-Mines/Telecom Bretagne, Dépt. LUSSI, Rennes, France
Emails: {frederic.cuppens; nora.cuppens; joaquin.garcia}@telecom-bretagne.eu
‡ École Nationale Supérieure de Mécanique et d'Aérotechnique, LISI, Poitiers, France

***Abstract*—The use of role engineering has grown in importance with the expansion of highly abstracted access control frameworks in organizations. In particular, the use of role mining techniques for the discovery of roles from previously deployed authorizations has facilitated the configuration of such frameworks. However, the literature lacks from a clear basis for appraising and leveraging the learning outcomes of the role mining process. In this paper, we provide such a formal basis. We compare sets of roles by projecting roles from one set into the other set. This approach allows to measure how comparable the two configurations of roles are, and to interpret each role. We formally define the problem of comparing sets of roles, and prove that the problem is NP-complete. Then, we propose an algorithm to map the inherent relation among the sets based on algebraic expressions. We demonstrate the correctness and completeness of our solution, and investigate some further issues that may benefit from our approach, such as detection of unhandled perturbations or source misconfiguration.**

***Keywords*-Access Control; Role Mining; IT Security; Boolean Logic;**

I. Introduction

Access control is a security fundamental concern which covers a wide area of applications including operating systems, database systems, enterprise resource planning systems, and workflow systems. Role Based Access Control (RBAC) [1] [2] is the dominant model for access control in both commercial and research fields. It structures into roles the concrete agents who have access requirements to secured objects, such as employees, programs, or processors. A role can be viewed from two perspectives: a set of authorizations that are recurrently assigned together, or a set of agents that are granted the same set of authorizations. As RBAC platforms have become essential to large enterprises, the major difficulty is still the configuration and establishment of the RBAC model into a functional state in the organization.

Role engineering is the practical discipline of implementing the RBAC model into organizations. Essentially, role engineering is the process of structuring the different agents of an organization into roles, and the association of the corresponding set of authorizations to every role. Top-down and bottom-up are the two common strategies in order to apply role engineering. Under the top-down approach, the roles are defined by carefully analyzing the business process associated with the organization, and decomposing it into smaller units in a functionally independent manner. Under the bottom-up approach, roles emerge from existing configurations of authorizations already deployed over the organization. The role engineering task is extremely time consuming and costly [3]. For this reason, the use of 'automatic' bottom-up role engineering techniques is growing in importance. These techniques get inspiration from data mining techniques, hence, they are known as *role mining* (RM). Several efficient role mining algorithms have been proposed in the literature [4]. However, they still require intensive human interpretation before leading to appropriate results. To the best of our knowledge, no efficient solutions have been proposed to assist a security administrator in the task of leveraging the outcomes of a role mining process. Moreover, as the years pass since the wide adoption of RBAC in organizations, a new requirement is strongly emerging. In addition to the traditional application of role engineering upon organizations where there is only a direct user-permission assignment framework, we may also need to apply role engineering in organizations where a pre-existing RBAC configuration exists but is no longer optimized. This may happen because the RBAC configuration is getting old and recurrently updated, or does not fit the dynamic nature of the organization anymore. In such cases, keeping as close as possible to the original RBAC configuration is the proper optimization criteria of existing role mining strategies [5]. In this regard, the ability of mapping the outcomes of a role mining process to the set of original authorizations may be highly beneficial in order to understand and validate the discovery process, as well as to detect unhandled perturbations over the deployed configurations, or misconfiguration within the original ones. Similarly, we also rise the necessity of comparing the outcomes of two different role mining processes. Indeed, several role mining tools have been proposed in the literature. Each new role mining technique needs to be tested and evaluated before being commercialized and used in real configurations. The test stage usually consists of running the role mining algorithm

on real and synthetic concrete data, generated from already known consistent RBAC state. The RBAC state obtained from the role mining is then compared to the original RBAC state, in a reverse engineering manner. However, we do not find any analytic comparison tool between the two RBAC configurations of roles in the literature. Thus, an automatic matching tool between roles from two sets is obviously needed, either for the assessment of the role mining methods, or for the interpretation of the mined roles.

In this paper, we formally define the problem of comparing two sets of roles according to the underlined requirements extracted from the literature. We demonstrate that the problem is NP-complete. We present a greedy solution that handles the motivation problem. Then, we prove the correctness and completeness of the solution. We define a sufficient condition that guarantees preciseness of the comparison between the original set of roles and mined set of roles. The experimental results confirm the validity of our approach, as a beneficial solution to analyze and better understand the set of mined roles when an original set of roles exists.

Paper organization. Section II reviews some technical definitions about the RBAC model and role mining problem and its algebraic representation. Section III formally defines the problem to solve. Section IV presents our solution to the problem, underlines some of the properties of our solution, and elaborates on some heuristics to enhance our approach. Section V presents experimental results. Section VI provides a comparison with related work. Section VII concludes the paper.

II. Preliminaries

In this section, we present the background about the Role Based Access Control (RBAC) model. In addition, we state a formulation of the role mining problem, and introduce the algebraic presentation of the involved data in role mining.

A. Role Based Access Control Model

The RBAC model is a NIST standard. It introduces the notion of "role" in order to make the access control system more compact and comprehensive, compared to the direct user-permission assignments. Definition 1 presents the basic model RBAC_0 [2] without considering sessions.

Definition 1. *RBAC*
An RBAC configuration, denoted as $RC = (ROLES, UR, RP)$, is characterized by:

- *U, $ROLES$, OPS, and OBJ, which are the sets of users, roles, operations, and objects*
- *$UR \subseteq U \times ROLES$, a many-to-many mapping user-to-role assignment relation*
- *$PRMS \subseteq \{(op, obj) | op \in OPS \wedge obj \in OBJ\}$, a set of permissions, where a permission is an operation over an object*
- *$RP \subseteq ROLES \times PRMS$, a many-to-many mapping of user-to-permission assignment relation*[1]
- *$assigned\_users(R) = \{u \in U | (u, R) \in UR\}$, the mapping of role R onto a set of users*
- *$assigned\_permissions(R) = \{p \in PRMS | (R, p) \in RP\}$, the mapping of role R onto a set of permissions*

B. The Role Mining Problem

Intuitively, role mining is the process of extracting a configuration of roles from a discretionary user permission assignment relation. The users granted with similar permissions are structured into similar roles. Symmetrically, permissions with common users are assigned together, so that they belong to the same roles. Optionally, some information about the business process or the user attributes may be provided, and then it is a hybrid role mining. Nevertheless, there is no consensus about the formal role mining problem definition. Several role mining techniques are proposed in literature, with different assumptions and optimization criteria leading to different solutions. For instance, the first explicit formal definition for the *role mining problem* is given in [6]. Vaidya et al. suggest the minimization of the number of generated roles as an optimization criteria, and show that this problem is NP-complete. The *inference role mining problem* presented by Frank et al. [7] focuses on the problem from a general perspective. In all cases, role mining assumes implicitly that an underlying RBAC configuration exists in the deployed authorizations, and aims to reveal this underlying configuration.

Definition 2. *Inference Role Mining Problem*
Let U be a set of users, $PRMS$ a set of permissions, UPA a user-permission assignment relation, and, optionally, part of the top-down information TDI be given. Infer the unknown RBAC configuration $RC^* = (ROLES^*, UR^*, RP^*)$, under the following assumptions:

1) *An underlying RBAC configuration exists*
2) *Exceptions (may) exist*
3) *TDI (if given) influences RC^**

Definition 2 provides a unified view of both bottom-up and hybrid role mining. Unlike other definitions, it does not give the function to optimize, i.e. the precise way to solve the problem. Therefore, to validate that the problem is solved, we must know the underlying RBAC configuration RC^*. We can compare the set of mined roles with the set of original roles in experimental scenarios, as well as in application scenarios where an old RBAC configuration exists.

Algebraic Representation of the Problem: To unify the representation of the inputs and outputs of role mining, the involved entities are represented with boolean matrices as follows: given m users, n permissions and k roles (i.e.,

[1]In the original NIST standard *RP* is defined as *RP* $\subseteq$ *PRMS* $\times$ *ROLES* rather than $ROLES \times PRMS$.

$|U| = m$, $|PRMS| = n$, $|ROLES| = k$), the user-to-role mapping UR is represented as an $m \times k$ matrix where 1 in cell $\{ij\}$ indicates the assignment of role j to user i. Similarly, the role-to-permission mapping RP is represented as a $k \times n$ matrix where 1 in cell $\{ij\}$ indicates the assignment of permission j to role i. Finally, the user-to-permission mapping UPA is represented as an $m \times n$ matrix where 1 in cell $\{ij\}$ indicates the assignment of permission j to user i. The relationship between UPA, UR and RP can be expressed with the boolean matrix multiplication.

Definition 3. *Boolean Matrix Multiplication*
A Boolean matrix multiplication between Boolean matrices $A \in \{0,1\}^{n \times k}$ *and* $B \in \{0,1\}^{k \times m}$ *is* $A \otimes B = C$ *where* $C \in \{0,1\}^{n \times m}$ *and* $c_{ij} = \bigvee_{l=1}^{k} \wedge a_{il} b_{lj}$.

In an RBAC configuration, we have $UPA = UR \otimes RP$. A user i is granted permission j if, at least, one of its roles is assigned to this permission. The role mining process approximates the above decomposition by extracting from a given UPA the relations UR^* and RP^* such that: $UPA \approx UR^* \otimes RP^*$.

III. Role Set Comparison Problem

A. Motivation

The requirement of comparing two sets of roles occurs in several use cases related to role mining. The first use case is the assessment of role mining algorithms. An algorithm designer usually evaluates the performance of its RM algorithm with the reverse engineering technique: starting from an RBAC configuration, apply role mining over the resulting UPA, and compare the obtained roles with the original roles. Likewise, we can compare the outputs of several role mining algorithms to test their performance under different constraints. The second use case is the enforcement of role mining results. A security administrator can require assistance toward migrating to a new RBAC configuration from an old RBAC configuration. A similar application is migrating from discretionary UPA to an RBAC configuration. Considering the set of permissions of each user as a pseudo-role, the administrator may need assistance to assign each user to the appropriate roles which guarantee the required permissions. A mapping of the pseudo-roles with the obtained new RBAC roles may solve the problem.

In almost all the above mentioned cases, we have typically a set of roles of reference or original roles OR and a set of mined roles MR. The first set is a set of mastered roles since they are well known by the security administrator, in opposition to the set of new roles MR. The two configurations of roles are defined for the same set of discretionary security rules UPA. However, they may have been calculated with respect to different constraints. For instance, one set can correspond to a hierarchical structure of roles whereas the other is flat. Similarly, the roles of one set may be partially overlapping whereas the roles of the other set are orthogonal.

Role mining algorithms usually output a list of roles, possibly overlapping or even redundant which means that a role can be fully covered by a union of another subset of roles from the same list. When migrating to a new RBAC configuration, the security administrator is confronted to such a list of new roles, and he has to assign them to the users and manage them to suit the organization's evolutive requirements. The administrator has to respect three access control rules: provisioning, security and maintainability. For provisioning, each user should have access in the new configuration of roles to all its privileges in the old one. For that, the administrator needs to know how to optimally cover the permissions provided by each role from OR using roles from MR. Security consists of not allowing any user to access extra-privileges. For this purpose, the administrator needs to ensure that the new assigned roles to the users do not exceed the privileges provided by their old roles. Finally, the maintainability aims to make simple and safe the evolution of the structure of roles by adding and retrieving permissions and users to the roles, according to the evolving requirements of the organization. One key characteristic to ensure the maintainability is mastering the configuration of roles. Thus, the security administrator, who masters the old roles or the old UPA, would be very interested in leveraging his experience with the old roles in order to master the new configuration of roles more quickly. Unfortunately, mapping the new roles with the old ones manually is not a viable task. It is essential then to assist the security administrator with automated tools to analytically understand the new roles. In particular, assistance is needed to find where the permissions of an old role has been distributed in the new configuration of roles, especially when moving between a flat role structure and a hierarchical role structure. In addition, such tool may help to find how to optimally assign the permissions to the users without adding extra-permissions using the new roles.

B. Problem Statement

The problem addressed in this paper is formally stated as follows:

Definition 4. *Role Set Comparison Problem*
Given two sets of roles OR and MR, both defined onto the same set of permissions PRMS, find for each role R in OR a minimal Disjunctive Normal Form DNF of roles in MR, such that DNF is included in R, and DNF maximally covers the permissions of R. By minimal DNF we mean that the size then the number of conjunctive clauses in the DNF are minimized.

Definition 4 proposes to express each role from OR with an algebraic formula of mined roles. The formula is presented as a disjunction of conjunctions of mined roles, i.e.

a *Disjunctive Normal Form (DNF)* expression. The structure of a DNF fits naturally the requirement of projecting a role R from OR in MR. The permissions of the role R are divided into several roles in MR or gathered with other roles in a larger role, or both. Thus R may be expressed by a combination of unions of roles and/or intersections of roles. The conjunctions express hierarchical relationships: when the role R becomes an intersection of roles in the new configuration MR, this means that the projection in MR of R or of a subset of it is a super role. Moreover, the use of negation of roles in combination with conjunction aims to retrieve a role from a role expressing exceptions. The disjunctions accumulate and cover the permissions of R when they are spread in MR. The problem also states that the DNF should still be included in R. This condition guarantees the security rule, and avoid assigning extra-privileges to the users. Symmetrically, the problem states that the coverage of the permissions should be maximized since the purpose is to understand the projection of the role in the new configuration. This is an approximation problem since the exact DNF can not always be found because of role mining errors, exceptions and shadowed roles. This issue is handled in the next section more in depth. The existence of the exact DNF matching, and the complexity of the DNF, are good indications about whether the two sets of roles are comparable or not. Besides, there may be different possibilities of DNF from MR to maximally cover a given role R. In such case, we find the DNF which involves less conjunctions of roles in order to reduce the size of the obtained hierarchical relationships. In a second stage, we also minimize the number of clauses. Finally, we note that the comparison is intended to be from original roles to mined roles, but it obviously can be done in the opposite way.

C. Complexity

We demonstrate that the problem stated in Definition 4 is NP-complete. We first decompose the problem considering the comparison of one role to a set of roles, and then we reformulate it as a decision problem, in order to apply the NP Completeness theory.

Definition 5. *Role-to-Role Set Comparison Decision Problem*
Given a role R and a set of roles MR, defined onto the same set of permissions $PRMS$, and given two integers k and l, is there a Disjunctive Normal Form DNF of roles from MR, with less than l literals per clause and less than k clauses, such that DNF equals R?

To prove that the problem is NP-complete, we show that:

1) The problem is NP.
2) There exists another known NP-complete problem, any instance of which can be reduced in a polynomial time to an instance of our problem, such that resolving our problem infers resolving the other problem.

Definition 6. *Set Covering Problem*
Given a universe U, a family of subsets S of U and an integer k. A cover is a subfamily $C \subseteq S$ of sets whose union is U. Is there a cover of size at most k?

Theorem 1. The *Role-to-Role Set Comparison Decision Problem* is NP-complete.

Proof: First, the problem is NP since checking the validity of a solution, i.e. calculating the set of permissions of a DNF and comparing it with R and counting the number of clauses and the number of literals in each clause, can be done in a polynomial time. Second, the set covering problem can be reduced to our problem. The universe U is assimilated to the role R. The family of subsets S is projected to the set of roles MR, which are sets of permissions. The size of the cover k is the same as the size of the DNF k, and we set l to 1. We get a special case of the *Role-to-Role Set Comparison Problem*, where all the roles of MR are included in R. Resolving this instance of the *Role-to-Role Set Comparison Problem* implies finding a DNF in MR, that maximally covers R with the minimal number of clauses and no conjunction at all. It infers finding a minimal cover for the universe U in S if such a cover exists. The transformation is obviously polynomial since it is a one to one mapping.

IV. Role Set Comparison Solution

In this section we provide an algorithm to solve the *Role Set Comparison Problem*. Afterwards, we analyze some important properties of our algorithm, and show its relevance in the context of role mining.

A. The Algorithm

Algorithm 1 is a greedy algorithm which solves the *Role Set Comparison Problem*. The algorithm takes as input two sets of roles, and gives as output, for each role R in the first set setR1, an algebraic expression formulated as a generalized union of intersections of roles taken in the second set of roles setR2 and their negations. The obtained expression, denoted DNF, is equal to R if any possibility to construct such an expression exists among setR2. Otherwise, it is the best approximation of R included in it

The universe of literals which can be used to build the DNF is CandidateRoles, containing all the roles in setR2 in addition to the negation of each of them (line 1). The algorithm is structured in two main nested loops. The first one is a *for* loop which handles the roles of setR1 sequentially. For each role, we create a variable DNF which will contain the solution and a vector called UncoveredPerms initialized to the whole set of permissions of R (line 3–4). This vector indicates the current coverage of the permissions of the role R gradually with the temporary values of DNF during the progress of Algorithm 1. The second main loop is a *while* loop (line 9–33). It checks all the possibilities

Algorithm 1 Compare(setR1,setR2)

Input: nRoles1×nPerms reference role-permission relation setR1

Input: nRoles2×nPerms compared role-permission relation setR2

Output: a DNF of roles from setR2 for each role in setR1. The DNF is equal to the role, or the closest included possible one if equality is not possible.

```
1: CandidateRoles ← setR2 ∪¬setR2
2: for each role R in setR1 do
3:    UncoveredPerms ← R
4:    DNF ← {}
5:    CandidateClauses ← CandidateRoles
6:    DiscardedClauses ← {}
7:    RIsCovered ← False
8:    ConjunctiveClauseLevel ← 1
9:    while ¬RIsCovered and CandidateClauses ≠ ∅ do
10:      for each clause C in CandidateClauses do
11:         if ¬RIsCovered then
12:            if C ⊆ R then
13:               if C ∩ UncoveredPerms ≠ ∅ then
14:                  DNF ← DNF ∪ C
15:                  UncoveredPerms←UncoveredPerms∩¬C
16:                  if UncoveredPerms == ∅ then
17:                     RIsCovered ← True
18:                  end if
19:                  for each clause C' in DNF do
20:                     if C' ⊂ (DNF - C') then
21:                        remove C' from DNF
22:                     end if
23:                  end for
24:               end if
25:               DiscardedClauses←DiscardedClauses∪C
26:            end if
27:         end if
28:      end for
29:      if ¬RIsCovered then
30:         ConjunctiveClauseLevel ← +1
31:         CandidateClauses←GenerateConjunctiveClauses(
            CandidateRoles,DiscardedClauses,
            ConjunctiveClauseLevel)
32:      end if
33:   end while
34: end for
35: return DNF for role R
```

Algorithm 2 GenerateConjunctiveClauses(CandidateRoles, DiscardedClauses,k)

Input: CandidateRoles: a role-permission relation representing the literals from which we build conjunctive clauses

Input: DiscardedClauses: a role-permission relation representing the set of clauses of less than k literals to discard from the resulting clauses

Input: k: the number of literals in each resulting clause

Output: a set of conjunctive clauses of k literals each

```
1: GeneratedClauses ← all the combinations of k roles
   from the CandidateRoles
2: for each clause C in DiscardedClauses do
3:    for each clause C' in GeneratedClauses do
4:       if C ⊂ C' then
5:          remove C' from GeneratedClauses
6:       end if
7:    end for
8: end for
9: return GeneratedClauses
```

of disjunctive clauses from CandidateRoles until the role R is totally covered where the obtained DNF exactly matches R, or until all the combination of clauses are tested, where the obtained DNF is the best approximation of R. Since a DNF is a disjunction of conjunctions, the DNF is built by adding each time a new conjunctive clause to the main disjunction. A clause is a conjunction of 1 to nRoles2 literals from CandidateRoles. For this purpose, the clauses are tested one by one. The number of literals in the disjunctive clauses called ConjunctiveClauseLevel is increased gradually. If the role is not covered by the current ConjunctiveClauseLevel, Algorithm 2 is called to calculate all the combinations of conjunctive clauses of a higher ConjunctiveClauseLevel. DiscardedClauses are the clauses which are included in R and do not contribute in covering the UncoveredPerms more than the current DNF, or are already included in the DNF. The following running example illustrates the algorithm.

B. Running Example

We consider the following configurations of roles defined onto $PRMS$ = {p1, p2, p3, p4, p5, p6, p7}. The first set of roles contains only one role setR1= {R = {p1, p2, p5, p6, p7}}. The second set is setR2 ={r1= {p1, p2}, r2={p1, p3}, r3={p3, p5, p6, p7}}. Since setR1 contains only one role, the main *for* loop (line 2–35) runs only one time. So we focus on the rest of Algorithm 1. First of all, the candidate literals are CandidateRoles = {$r1$, $r2$, $r3$, $\neg r1$, $\neg r2$, $\neg r3$} (line 1). After the initialization step (line 3–8), we get: UncoveredPerms = R; DNF = {}; CandidateClauses = CandidateRoles; DiscardedClauses = {}; RIsCovered = False; and ConjunctiveClauseLevel = 1, which means that we will first try to cover R using disjunctions of roles from setR2. In the first iteration of the *while* loop (line 9–33) of Algorithm 1: We enter the *for* loop (line 10–28) which tests the clauses in CandidateClauses one by one. Only the clause r1 is included in R, and satisfies the conditions of line 12 and line 13. Thus, {r1} is added to DNF, and UncoveredPerms is updated to {p5, p6, p7} (line 14–15).

At the end of the *for* loop (line 10–28), since the value of RIsCovered is still False, the number of literals per clause ConjunctiveClauseLevel is increased to 2, meaning that we will test the clauses of disjunctions of two literals next. Algorithm 2 is called to generate a new set of clauses (line 30–31) . The result of this call is CandidateClauses=$\{r2 \cap r3$, $r2 \cap \neg r1$, $r2 \cap \neg r3$, $r3 \cap \neg r1$, $r3 \cap \neg r2$, $\neg r1 \cap \neg r2$, $\neg r1 \cap \neg r3$, $\neg r2 \cap \neg r3\}$. It is all the combinations of conjunction of two roles from CandidateRoles, minus the clauses involving DiscardedClauses. We notice that r1 does not take part in any of the candidate clauses, since it is a discarded clause now. We also exclude the clauses involving a role and its negation because it is the empty set. In the second iteration of the *while* loop (line 9–33) of Algorithm 1: the *for* loop (line 10–28) tests the new set of CandidateClauses one by one again. The first clause to pass the test in line 12 (inclusion in R) is $r3 \cap \neg r2$. This clause covers all the remaining UncoveredPerms. So DNF is updated to $\{\{r1\}, \{r3, \neg r2\}\}$ which is interpreted: DNF $= r1 \cup (r3 \cap \neg r2)$ i.e. R is the association of r1 and r3 minus r2. UncoveredPerms is updated to $\emptyset$ (line 15), and so RIsCovered is set to true (line 17). The *for* loop (line 19–23) checks that the new added clause does not cover the previously added clauses to DNF, in order to remove the obsolete clauses. There will be no further loops and no further generation of clauses of conjunction of higher number of literals because RIsCovered is set to true.

C. Properties of Algorithm 1

Theorem 2. Correctness. For each role R in OR, the returned DNF is included or equal to R.

Proof: Lines 12 and 13 of Algorithm 1 guarantee that the DNF is always a subset of or equal to R.

Theorem 3. Completeness. For each role R in OR, if a disjunctive normal form of roles from MR equal to R exists, Algorithm 1 returns an exact matching DNF for R. If no exact matching DNF exists for the role R, Algorithm 1 returns a maximal DNF of roles from MR included in R (with respect to set inclusion).

Proof: Algorithm 1 tests exhaustively all the possible configurations in the worst case, enhancing the covering of R gradually. Thus the returned DNF maximally covers R.

Theorem 4. Compactness. For each role R in OR, if several different disjunctive normal forms of roles from MR with the maximum coverage of R exist, Algorithm 1 returns a DNF with a minimal level of conjunctions and a minimal number of clauses with respect to that level.

Proof: The proof derives from the structure of the algorithm in two nested loops. The $ConjunctiveClauseLevel$ is increased gradually, after all the clauses in the lower conjunctive levels are tested. If no enhancement of the coverage can be achieved in a conjunctive level, no clauses are picked in that level. Besides, the number of clauses is minimal because a clause is added only if it enhances the coverage. Each time a new clause is added to the DNF, the for loop (line 19–23) in Algorithm 1) checks if any other clause previously added has became redundant and retrieves it from the DNF to keep the number of clauses minimal. A clause is redundant if all the permissions it covers are already covered by one or multiple other clauses.

Theorem 5. Constrained Completeness Guarantees. Given two sets of roles RP1 and RP2 which may lead to the same User Permission Assignment relation UPA, i.e., there exists two correspondent User-Role matrices UR1 and UR2 such that $UR1 \otimes RP1 = UR2 \otimes RP2 = UPA$;
For any role R from RP1, if there exists a subset of k users such that $\bigcap_{i=1}^{k} assigned\_perms(U_i) = R$ then there exists at least a Disjunctive Normal Form DNF of roles from RP2 which exactly matches R.

Proof: Since the two configurations of roles are equivalent, the k users which have exactly the permissions of the role R in common, must have exactly the same permissions in the two RBAC configurations. Let the roles assigned to user U_i in RP2 be referred as r_{ij}, j from 1 to l_i, where l_i is the number of roles assigned to the user U_i in RP2. Then we have: $assigned\_perms(U_i) = \bigcup_{j=1}^{l_i} r_{ij}$. R is the intersection of the permissions of the users U_i, i from 1 to k, thus it can be denoted $R = \bigcap_{i=1}^{k} \bigcup_{j=1}^{l_i} r_{ij}$, which is a Conjunctive Normal Form (CNF) of roles from RP2. Using the generalization of the Morgan's law [8], we can transform the obtained CNF to a DNF.

***Shadowing Detection in Role Configuration**:* We combine the results from Theorems 3 and 5 and conclude that in the case of role mining without errors, Algorithm 1 finds for each role from one set (OR or MR) a DNF of roles from the other set which exactly matches with it, Unless the condition mentioned in Theorem 5 (there exists a subset of k users such that $\bigcap_{i=1}^{k} assigned\_perms(U_i) = R$) is not satisfied. Moreover, this condition is closely related with the concept of role. Indeed, a role can be viewed as a set of permissions assigned together to a set of users. Thus, if Algorithm 1 does not find an exact matching DNF for a given role in a context of role mining, this can be interpreted as a possible misconfiguration of that role. The first possibility is that the User_Role assignment is incomplete, which means that this role did not have been assigned to any user yet. For instance, a job position has been created but the affected employee is not hired yet. The second possibility is that some permissions of the role are shadowed by other roles. That is to say, the role overlaps some permissions with other roles, and is always assigned to a user together with at least

one of these overlapping roles. The common permissions can underline then a misconfiguration of the role or a non optimal hierarchization in the structure of roles. In both cases, it is relevant to warn the security administrator. We note that this condition is independent of the used role mining technique.

A Similarity Metric: The result of Algorithm 1 can be leveraged to obtain a similarity metric between two sets of roles. Indeed, we can sum the distances between each role from setR1 and its DNF, with distance(R,DNF) = $\frac{uncoveredPerms(R,DNF)}{Assigned\_Perms(R)}$. The obtained distance, divided by the number of roles nRoles1 is an indicator of how well the set setR1 is represented in setR2.

D. Time Complexity of Algorithm 1

The time complexity of Algorithm 1 depends on both the size and the nature of the input data. If the inputs are two sets of roles of size n roles and m roles respectively, then the worst case complexity is O($n \times 4^m$). The worst case is when the two sets of roles are not comparable, which means that for each role in setR1 there is no DNF of roles from setR2 which exactly matches it. The algorithm will test all the possible configurations of disjunction of conjunction of roles in setR2, and the output will be only an approximation of the roles. The best case complexity is O($n \times 2m$), when the two sets of roles are identical. We note that the complexity is independent of the number of permissions and users.

E. Heuristic to Reduce the Complexity

Algorithm 1 solves an NP-complete problem. The worst case complexity is high. However, since we intend to use the algorithm in a context of role mining where the roles are comparable, we expect that the exact matching for each role will be found at a low level of ConjunctiveClauseLevel. Otherwise, we can guess that no matching DNF can be found for the handled role. Thus, we can limit the Conjunctive-ClauseLevel by setting a threshold,in order to avoid that the algorithm explores all the further combinations.

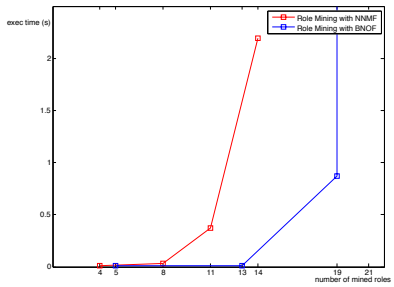
V. Experimental Results

We have implemented a random RBAC configuration generator to get synthetic data for our experiments. This tool takes as input the characteristics of an RBAC configuration, notably the number of users, the number of permissions and the number of roles. In addition, it uses two parameters: the density of the user-role relation UR, and the density of the role-permission relation RP. The outputs are the three boolean matrices UR, RP, and UPA where $UPA = UR \otimes RP$. We generate eight data sets with different sizes described in Table I with density set to 0.1. We implement the test plateform in MATLAB_R2011a. All the experiments have been made with a 2.26 GHz Intel Core 2 Duo processor on Mac OS X V 10.7.2. The results of the experiments are summarized in Figure 1.

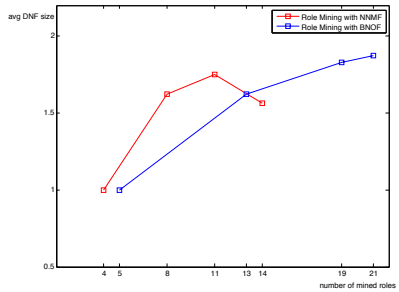
n	nUsers	nPerms	nOR
1	10	15	4
2	18	25	8
3	25	30	12
4	35	40	16
5	200	350	20
6	400	500	30
7	500	800	40
8	600	1000	50

Table I
SIZE OF RBAC DATA SETS.

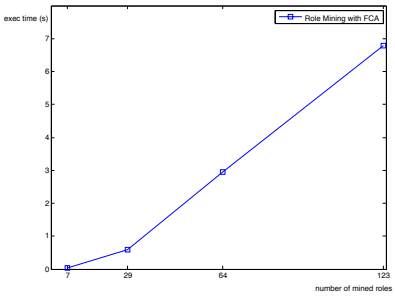
First, we use the data sets 1, 2, 3 and 4 in Table I for role mining tests. We provide the UPA matrix of each data set as input to a given role mining algorithm, and we compare the original roles in UR with the mined roles in UR^* using the Algorithm 1. We test three different role mining algorithms. We adapt two techniques of matrix compression usually used in data mining, namely the *Non Negative Matrix Factorization (NNMF)* [9] and the *Binary Non-Orthogonal Decomposition (BNOF)* [10]. Both techniques take an UPA matrix from a data set and give an approximative decomposition of the matrix into the product of two matrices of lower rank, namely UR and RP. This is done by emphasizing the frequent patterns in the original matrix, which can be interpreted as roles in the context of role mining. However, the two techniques differ in their constraints and optimization objective. NNMF gives a factorization into two integer non negative matrices, while minimizing the root-mean-squared. We transform the obtained positive matrices into boolean matrices. BNOF uses another decomposition technique which gives a boolean decomposition of overlapping roles, but a user can be assigned to only one role. The results of this first series of tests with NNMF and BNOF are given in Figure 1 (a) and (b). The purpose is to observe the behavior of Algorithm 1 when comparing two sets of roles which have been configured with different objectives. All role mining processes presented in this paper have finished without approximation errors. The number of original roles rise from 4 to 16, but the number of mined roles with NNMF and BNOF is slightly different from the number of original roles. In Figure 1 (a), we plot the execution time of Algorithm 1 in function of the number of mined roles because, as stated in the section IV-D, the complexity of the algorithm mainly depends on this latter parameter. The execution time increases with the number of mined roles. Besides, the curve of the role mined with BNOF shows an exponential increase in the 4th data set in point (19 mined roles, 253 s). This is because there is an original role for which no exact DNF can be calculated. This role is shadowed in the original configuration of roles. Thus, Algorithm 1 reaches the worst complexity case for this data set. The execution time does not only depend on the number of mined roles, but it is mostly dependent of



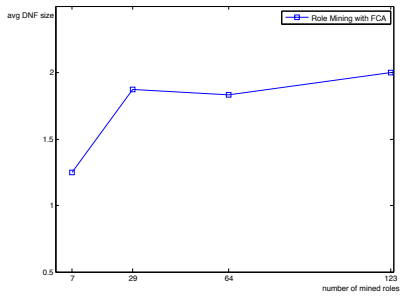
(a) Execution time when comparing roles of the RBAC data sets 1, 2, 3 and 4 with the mined roles by NNMF and BNOF.



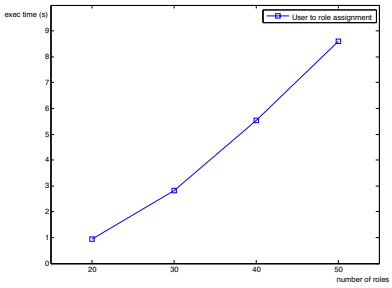
(b) Average DNF size with NNMF and BNOF.



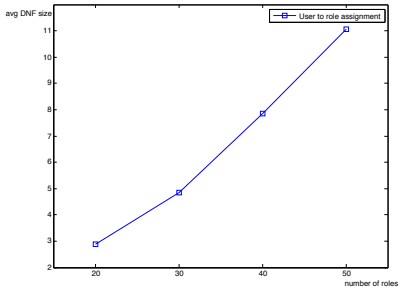
(c) Execution time when comparing roles of the RBAC data sets 1, 2, 3 and 4 with the mined roles by FCA.



(d) Average DNF size with FCA.



(e) Execution time when comparing UPA with RP in RBAC data sets 5, 6, 7 and 8.



(f) Average number of assigned roles to each user.

Figure 1. Experimental Results

the nature of the compared roles and how resembling they are. In Figure 1 (b), we plot the average size of the obtained DNFs for each data set. By size of DNF we mean here the number of involved roles, no matter if in conjunctive or disjunctive clauses. The figure shows that the average size of the DNFs also increases with the number of mined roles. We notice a correlation between the size of the DNFs and the execution time. In particular, the execution time grows with the maximum number of conjunctions in the DNFs, which rises from one to three in this series of tests.

The third role mining algorithm we use is a *Formal Concept Analysis (FCA)* algorithm [11]. Considering the users as objects, and the permissions as their attributes, FCA calculates a lattice of all the possible formal concepts. We consider each Formal concept as a mined role. Role Mining algorithms in literature usually process to a pruning of the lattice to discard a subset of the concepts with regard to some optimization criteria, but in this paper, we keep all the concepts. This explains the high number of mined roles rising from 7 to 123 in Figure 1 (c) and (d), corresponding

to sets of original roles of only 4 to 16 roles. Our purpose is to test the ability of Algorithm 1 to compare a hierarchical configuration of roles with a flat configuration of roles. We aim also at testing the scalability of Algorithm 1 in such cases. The results in Figure 1 (c) and (d) show that the algorithm execution time scales well with the number of mined roles, where all the roles can match their exact DNFs. This is also explained by the fact that the average size of the obtained DNFs is still between 1 and 2, meaning that Algorithm 1 succeeds in matching the original roles with the appropriate roles in the hierarchical configuration of roles. Actually, the size of the DNFs varies between 1 to 6, and the number of conjunctions does not exceed three roles, which is an intelligible amount of data that can be managed by a security administrator.

Finally, we use the remaining last four sets of data in Table I, to test the ability of Algorithm 1 to perform a user to role assignment. We provide as input both the UPA (which may be considered as pseudo-roles as explained previously), and the set of roles generated by the random generator. The results are given in Figure 1 (e) and (f). Figure 1 (e) shows that the algorithm scales well with large data sets, since the execution time does not exceed 9 seconds. All the users are assigned to their appropriate roles, and the average size of the DNFs presented in Figure 1 (f) going up to 11 roles is simply the average number of roles assigned to each user, since no conjunctions has been involved in the generated DNFs in all the four tests.

VI. Related Work

Research on role mining has revealed different requirements for the comparison of roles. Several propositions in this field have been presented.

In [12], Kuhlmann et al. address the problem by counting how many of the roles lying in the original reference set are exactly the same to those discovered by their proposed role mining method. Similarly, Vaidya et al. in [13] consider the average, instead of the total number, of original roles that are exactly discovered by their role mining technique, as an evaluation metric for role mining. Thus, their metric provides means to compare different role mining approaches, being not influenced by the number of roles, and giving a similarity of one for total coincidence, and zero for total difference. The main drawback of these two first metrics is that they discard roles that are very similar but not exactly the same than roles in the original role set.

Hassan Takabi et al. in [14], consider the case in which an initial RBAC configuration is upgraded during the role mining process. The similarity between roles is one of the two optimization criteria used by their role mining approach, the minimality of the resulting number of roles being the second criteria. In order to address similar issues, Vaidya et al. propose in [5] a series of similarity and dissimilarity metrics based on the Jaccard Coefficient and Jaccard Distance. Their metrics have been recurrently used in the literature to compare roles and sets of roles; and to evaluate role mining algorithms [4]. However, these Jaccard-based measurements still suffer from an important limitation: it computes role similarity with respect to only one role in the compared set, and the same role may be used to be compared with several roles from the other set. Moreover, as a quality performance criteria, it may artificially be affected by reference sets with a high number of closely similar roles.

Alternatively, P. Streich et al. [15] propose a one-to-one matching for the comparison of two sets, based on the average Hamming distance. Their method outputs a single permutation of all roles for the second set, that gives a one-to-one mapping between the same number of roles kept in the first set. This avoids the drawback of the previous proposition of mapping independently many discovered roles to the same original role. However, the method is not specifically designed for role mining applications, but for clustering in general. We think that the Hamming distance is not appropriate for role-to-role distance measure. In fact, it does not take into account the similarity of the roles, but only the difference. More recently, Molloy et al. [16] has proposed a one-to-one mapping between the roles from two sets, also based on the Jaccard Coefficient. In order to evaluate the role stability in the presence of noise as defined in [17] , they compare the two sets of roles generated by role mining over noisy data, and clean data. They assimilate the role mapping problem to a bipartite matching optimization problem known as *The Minimum Weighted Bipartite Matching Problem* (MWBM). The solution contains the closest pairwise matching by maximizing the global similarity metric, with means that each role in SR1 is matched with only one role in SR2, such that the sum of the distances is minimized. But this similarity measure is still straightforward. In general, we may wish to define similarity between sets of roles in a much more sophisticated fashion. For example, note that all the above measures still assume that a role can only be mapped to another role. In general, this is not true. For example, let us assume that one set has the role R1 = (p1,p2,p3) and the other set has the roles {R1 =(p1), R2=(p2,p3)}. While the Jaccard measure will give a score of 0.33 and 0.66 respectively for the roles R2 and R3, and while the bi-partite mapping will choose to match R1 to R3 and discard R2, it is clear that taken together, the sets of roles are quite similar.

The previous work only focuses on the issue of comparing two sets of roles from the perspective of finding a similarity metric and reduces the problem to a simple pairwise role to role mapping. Morover, it fails to provide an accurate comparison of two sets of roles synthetically. In this paper, we consider the problem of comparing two sets of roles from a new perspective. Our approach allows a semantic analysis of the resulting set of mined roles in comparison with the preexisting set of roles, and the detection of misconfiguration

of roles. We provide a tool that, beyond being indicative about how comparable two sets of roles are, can assist a security administrator in the comprehension of the mined roles and their enforcement.

VII. Conclusion

The comparison of two configurations of roles is an important issue in the research and the application of Role Based Access Control, particularly with the accession of the role mining techniques. In this paper, we have provided a formal approach to handle the problem of comparing two sets of roles. We have stated the Role Set Comparison Problem (RSCP) as the problem of finding for each role in one set an expression of a disjunctive normal form of roles from the other set. We have demonstrated that RSCP is NP-complete. Afterwards, we have presented a greedy algorithm which solves it. We have proved the correctness and the completeness of the comparison algorithm. We have also defined a relevant condition (Constrained Completeness Guarantee) to compare sets of roles representing the same access control relation and detect some role misconfiguration issues. We have evaluated the time complexity of our algorithm. Finally, we have presented some experimental results which validate the efficiency of our algorithm in comparing configurations of roles defined with different criteria, such as structured and non structured hierarchies of roles. We have also tested the algorithm skills in deploying an RBAC policy by assigning the mined roles to the users.

As perspective of this work, we consider investigating some techniques of preprocessing the compared sets of roles, in order to lower the complexity of the solution.

Acknowledgment

This research was partially supported by the European Commission, in the framework of the ITEA2 Role-ID (Grant agreement no.08007) for Safaà Hachana and in the framework of the ITEA2 Predykot project (Grant agreement no.10035) for Nora Cuppens-Boulahia, Frédéric Cuppens and Joaquin Garcia-Alfaro.

References

[1] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control model," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, February 1996.

[2] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed *NIST* standard for role-based access control," NIST, standard, 2001.

[3] M. P. Gallaher, A. C. O'Connor, and B. Kropp, "The economic impact of role-based access control," RIT for NIST (National Institute of Standards and Technology), Tech. Rep., March 2002.

[4] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo, "Evaluating role mining algorithms," in *Proceedings of the 14th ACM symposium on Access control models and technologies SACMAT '09*. ACM, June 2009, pp. 95–104.

[5] J. Vaidya, V. Atluri, Q. Guo, and N. R. Adam, "Migrating to optimal *RBAC* with minimal perturbation," in *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies Proceedings SACMAT '08*. ACM, June 2008, pp. 11–20.

[6] J. Vaidya, V. Atluri, and Q. Guo, "The role mining problem: finding a minimal descriptive set of roles," in *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies SACMAT'07*. ACM, June 2007, pp. 175–184.

[7] M. Frank, J. M. Buhmann, and D. Basin, "On the definition of role mining," in *Proceeding of the 15th ACM symposium on Access control models and technologies SACMAT '10*. ACM, June 2010, pp. 35–44.

[8] R. L. Goodstein, *"Boolean Algebra"*. Dover Publications, 2007.

[9] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Neural Information Processing Systems (NIPS)*, 2000, pp. 556–562.

[10] M. Koyuturk, A. Grama, and N. Ramakrishnan, "Nonorthogonal decomposition of binary matrices for bounded-error data compression and analysis," *ACM Transactions on Mathematical Software*, vol. 32, no. 1, pp. 33–69, March 2006.

[11] A. Colantonio, R. D. Pietro, and A. Ocello, "Leveraging lattices to improve role mining," in *Proceedings of the 23rd International Information Security Conference IFIP SEC'08*. Springer, September 2008, pp. 333–347.

[12] M. Kuhlmann, D. Shohat, and G. Schimpf, "Role mining - revealing business roles for security administration using data mining technology," in *Proceedings of the 8th ACM symposium on Access control models and technologies SACMAT '03*. ACM, 2003, pp. 179–186.

[13] J. Vaidya, V. Atluri, and J. Warner, "Roleminer: mining roles using subset enumeration," in *Proceedings of the 13th ACM Conference on Computer and Communications Security CCS'06*. ACM, November 2006, pp. 144–153.

[14] H. Takabi and J. B. Joshi, "Stateminer: An efficient similarity-based approach for optimal mining of role hierarchy," in *Proceedings of the 15th ACM symposium on Access control models and technologies SACMAT '10*. ACM, June 2010, pp. 55–64.

[15] A. P. Streich, M. Frank, and J. M. Buhmann, "Multi-assignment clustering for boolean data," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 969–976.

[16] I. Molloy, N. Li, Y. A. Qi, J. Lobo, and L. Dickens, "Mining roles with noisy data," in *Proceedings of the 15th ACM symposium on Access control models and technologies SACMAT '10*. ACM, June 2010, pp. 45–54.

[17] A. Colantonio, R. D. Pietro, A. Ocello, and N. V. Verde, "Mining stable roles in *RBAC*," in *Proceedings of the 24th International Information Security Conference IFIP SEC'09*. Springer, 2009.