Policy Mining : a Bottom-Up Approach Toward Network Security Management

Safaà Hachana

▶ **To cite this version:**

Safaà Hachana. Policy Mining : a Bottom-Up Approach Toward Network Security Management. Other [cs.OH]. ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique - Poitiers, 2014. English. NNT : 2014ESMA0017 . tel-01129078

HAL Id: tel-01129078

https://theses.hal.science/tel-01129078

Submitted on 10 Mar 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour l'obtention du Grade de
DOCTEUR DE L'ÉCOLE NATIONALE SUPÉRIEURE DE MÉCANIQUE ET D'AÉROTECHNIQUE
(Diplôme National – Arrêté du 7 août 2006)

École Doctorale :
Sciences et Ingénierie pour l'Information

Secteur de Recherche : Informatique
Présentée par:

Safaà Hachana

****

Policy Mining : A Bottom-Up Approach for Network Security Management

****

Directeur de thèse : Yamine Ait-Ameur
Co-directeur : Frédéric Cuppens
Co-encadrant : Nora Cuppens-Boulahia

****

Soutenue le 4 juillet 2014

Devant la Commission d'Examen

****

JURY

Radu State, Assistant-Chercheur du corps intermédiaire, Université du Luxembourg, Président
Yamine Ait-Ameur, Professeur-CNU-27, INPT-ENSEEIHT/IRIT, Examinateur
Ladjel Bellatreche, Professeur-CNU-27, LIAS/ISAE-ENSMA, Examinateur
Frédéric Cuppens, Professeur, Institut Mines Télécom/ Télécom Bretagne, Examinateur
Nora Cuppens-Boulahia, Directeur de recherche, Institut Mines Télécom/Télécom Bretagne, Examinateur
Refik Molva, Professeur, Eurocom, Rapporteur
Stéphane Morucci, PDG de la société SWID, Examinateur
Mohand-Said Hacid, Professeur-CNU-27, LIRIS-Université Claude Bernard Lyon I, Rapporteur

Abstract

Today's corporations rely entirely on their information systems, usually connected to the Internet. Network access control, mainly ensured by firewalls, has become a paramount necessity. Yet, the management of manually configured firewall rules is complex, error prone, and costly for large networks. Using high abstract models such as the *Role Based Access Control (RBAC)* model has proved to be efficient in the definition and management of access control policies. Recent interest in *role mining*, which is the bottom-up approach for automatic RBAC configuration from the already deployed authorizations, has promoted further the development of this model.

This thesis is devoted to a bottom-up approach for the management of network security policies from high abstraction level with low cost and high confidence. Thus we show that the *Network Role Based Access Control (Net-RBAC)* model is more adapted to the specification of network access control policies than the RBAC model. We propose *policy mining*, a bottom-up approach that extracts from the deployed rules on a firewall the corresponding policy modeled with Net-RBAC. We devise a generic algorithm based on matrix factorization, that could adapt most of the existing role mining techniques to extract instances of Net-RBAC. Furthermore, knowing that the large and medium networks are usually protected by more than one firewall, we aim to provide a complete automatic bottom-up framework for network policy mining. We handle the problem of integration of Net-RBAC policies resulting from policy mining over several firewalls. We demonstrate how to verify security properties related to the deployment consistency over the firewalls in the meantime.

Besides, our comprehensive analysis of research axes around role mining, enables us to note that literature lacks a clear basis for appraising and leveraging the learning outcomes of role mining process. In this thesis, we provide assistance tools for administrators to analyze role mining and policy mining results as well. We formally define the problem of comparing sets of roles and evidence that the problem is NP-complete. Then, we devise an algorithm that maps the inherent relationship between the sets based on Boolean expressions, and projects roles from one set into the other set. This

approach is useful to measure how comparable the two configurations of roles are, and to interpret each role. We investigate some further related issues such as the detection of unhandled perturbations or source misconfiguration. In particular, emphasis on the presence of shadowed roles in the role configuration will be put as it increases the time complexity of sets of roles comparison. We provide a solution to detect different cases of role shadowing.

Each of the above contributions is rooted on a sound theoretical framework, illustrated by real data examples, and supported by experiments.

Résumé

Les Systèmes d'Information (SI) se sont diffusés massivement dans les organisations contemporaines pour soutenir les processus de gestion, de production et de commercialisation. Dans ce contexte, le contrôle d'accès aux réseaux informatiques, assuré essentiellement par des pare-feu, revêt une importance capitale. Cependant, la gestion manuelle des pare-feu s'avère une tâche complexe, coûteuse et sujette à l'erreur.

L'utilisation de modèles à haut niveau d'abstraction comme le modèle *RBAC (Role Based Access Control)* a prouvé son efficacité dans la définition et la gestion des politiques de contrôle d'accès. L'adoption de RBAC a bénéficié de l'intérêt porté plus récemment à la fouille de rôles ascendante pour une configuration automatique du modèle à partir des autorisations préalablement déployées. Cette discipline est communément appelée *role mining*.

Cette thèse est consacrée à une approche ascendante pour l'administration de la sécurité des réseaux informatiques d'un haut niveau d'abstraction avec l'assurance d'un coût bas et d'un niveau de confiance élevé. Nous montrons que le modèle *Net-RBAC (Network Role Based Access Control)* est plus adapté à la spécification des politiques de contrôle d'accès des réseaux que le modèle RBAC.

Nous proposons une approche baptisée *policy mining* qui extrait de manière ascendante et automatique la politique de contrôle d'accès modélisée par Net-RBAC à partir des règles de sécurité déployées sur un pare-feu. Nous définissons un algorithme basé sur la factorisation matricielle, capable d'adapter la plupart des techniques de *role mining* existantes afin d'extraire des instances du modèle Net-RBAC.

En plus, comme les réseaux des entreprises sont souvent protégés par plusieurs pare-feu, dans notre quête d'une solution complète pour la fouille automatique de la politique de contrôle d'accès du réseau entier, nous traitons le problème d'intégration de politiques modélisées par Net-RBAC et résultant de l'application de la technique de *policy mining* sur plusieurs pare-feu chacun à part. Par la même occasion, nous

vérifions les propriétés de sécurité reliées à la cohérence du déploiement de la politique sur plusieurs dispositifs.

En outre, nous avons noté pendant notre étude des axes de recherche autour du *role mining* un manque de base claire quant'à l'exploitation des résultats émanant du processus ascendant. Nous proposons dans cette thèse des outils destinés à assister un administrateur de sécurité réseau dans l'analyse des résultats aussi bien du *role mining* que du *policy mining*. Nous proposons une formalisation du problème de comparaison de deux configurations de rôles et prouvons que le problème est NP-Complet. Nous définissons un algorithme qui identifie des relations pertinentes entre les rôles respectifs de deux configurations équivalentes, et fournit la projection des rôles d'une configuration dans une autre, en se basant sur des expressions Booléennes. Cette approche est appropriée pour mesurer la comparabilité de deux ensembles de rôles et aussi pour interpréter chaque rôle. Nous étudions d'autres questions connexes comme la détection des perturbations de configuration à la source. En particulier, nous soulignons que la présence de rôles ombragés dans une configuration de rôles se manifeste par une augmentation de la complexité de la comparaison avec une autre configuration. Nous présentons une solution pour détecter différentes anomalies d'ombrage dans une configuration de rôles.

Chacune des contributions sus-mentionnées se base sur un cadre théorique solide, est illustrée par des exemples réels, et est appuyéé par des résultats expérimentaux.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Nowadays, the majority of companies is completely dependent on their information systems, either for their internal functioning, or for their commercial web interfaces. Protecting these networks from harmful intrusions has become a priority. A reliable access control policy is essential to ensure good and continuous working of these systems. The access control policy is mainly deployed on firewalls. A small firm usually has at least two firewalls. The configuration and the management of these firewalls represent hard tasks. Filtering rules are enforced in a vendor specific low level language, the number of rules is usually high, and the rules are order sensitive. Performed manually, the configuration of access control often results in misconfigured firewalls. Constructors mainly focusing on performance problems such as enhancing the capacity and the speed of the firewalls have achieved important advances in this field. However, management issues have not still received enough attention. The firewall management problem becomes more and more urgent due to increasing complexity in modern security policies.

One of the most important approach proposed to handle firewall management problems is to configure firewalls starting from high level natural language, down to device specific low level language, going through high abstraction models [18, 44, 62]. This methodology is likely to overcome problems encountered when using low level vendor specific languages. It would decrease the probability of human error, allow dynamic firewall configuration, and provide a reliable configuration of a system of firewalls to work in unison [70]. Safe automatic methods to ensure the translation and the deployment of a high abstraction level policy on firewalls exist [54, 55]. The greatest stumbling block to the wholesale adoption of such an approach lies with the initial definition of the high level policy, especially when a configuration of the firewalls already exists. For most security administrators, it is not worth throwing away the deployed filtering rules and starting a new policy definition from scratch. Providing a bottom-up approach that automatically extracts instances of a high level policy from the deployed rules on

a firewall is likely to highly promote the usage of this model based policy management approach.

Recently, the emergence of *role mining* [32] has become a new catalyst of the expansion of the *Role Based Access Control (RBAC)* [23] model in organizations. Role mining is a discipline that treats automating the extraction of RBAC roles from the already deployed set of direct authorizations in a system, by leveraging data mining tools. Prior to role mining, the task of *role engineering* consisting in structuring the different organization actors into roles and assigning authorizations to them, has been fully manual for years. This approach to role engineering relies on top-down information and defines roles by decomposition. Security experts have to consider the different use cases, and conduct interviews with business experts and users in order to deeply understand the semantics of business processes. Then, they define the roles by carefully analyzing the business processes and decomposing them into smaller units in a functionally independent manner. Manual approach to role engineering suffers from several limits. First, it is very expensive. According to a NIST report [29], role engineering is estimated to consume 60% of RBAC framework set up and exploitation costs. Second, it is a long process that may last several months. Third, it usually requires the involvement of security access control advisors in the internal business process, which may raise serious security issues. Forth, it does not fully leverage the existing access control framework, since the usage of the bottom-up information remains limited. Finally, it suffers from scalability limits. In a context of dozens of business processes, tens of thousands of users and millions of authorizations, the operation may become unfeasible, and has seldom been achieved successfully. Thus, relying only on manual role engineering has revealed to be insufficient, not viable, constituting a limit to the deployment of the RBAC model in enterprises. In this context, *role mining* is regarded as the best alternative to traditional role engineering approaches. When Kuhlmann et al. [40] have first suggested automating the bottom-up role engineering approach by using existing data mining techniques to extract the roles from the deployed user permission assignments, this was considered a very attractive idea. It promises to drastically reduce the process cost and complexity. The time required to perform role engineering is likely to decrease from several months with the manual approach to few seconds/hours with the automatic role mining approach. Moreover, the guarantee to take into consideration the existing user-to-permission assignments is an argument that could encourage organizations to move to the usage of RBAC more confidently. Hence the beginning of intensive research work on role mining.

Nevertheless, applying role mining solutions proposed in literature directly to firewall rules cannot provide interesting outcomes. Indeed, though *RBAC* model has

imposed itself as the standard high abstraction level access control policy model, and has proved to be very efficient in a wide area of access control applications such as physical security package, environmental security, operating system security, and staff security [28], it still does not fully capture the specificities of network access control policies. From the perspective of the RBAC model, the access control security rules are considered to follow the pattern: [*user u is allowed access to permission p*], with p an operation over an object. Users are the central entities, and RBAC introduces the concept of *role* to structure them. However, when we focus on the structure of a network access control rule, it is shaped [*allow source_host sh to send service of type s to destination_host dh*]. In this pattern, the three involved entities are semantically on the same level of importance from the network access control perspective. Cuppens et al. have shown that a model that captures this ternary relation allows to define a network security policy efficiently [18]. The *Network RBAC (Net-RBAC)* model contains and generalizes the concept of *role* from RBAC by adding the concepts of *activity* to structure the services, and *view* to structure the destination_hosts, the same as *role* structures the source_hosts into higher level of abstraction entities. Role mining techniques do not fit the Net-RBAC model since they structure only one of the three entities. Applying role mining to structure each of the three entities separately is not feasible since it would output unconnected abstract entities that cannot be related with security rules to express the original access control policy.

1.1 Contribution

In this thesis, we bridge the gap between recent achievements in the role mining field and the application of high abstraction access control models to the management of the network security policies.

To this purpose, we survey the research field of role mining. We define a complete and realistic business process that gathers all the issues related to role mining and are necessary to leverage it in practice. This perspective permits to have a better visibility on this research field, to discern concurrent and complementary solutions, and to capture the missing links in order to bring role mining from theory and academic research into practice and industry. We analyze the different components of the process, starting from collecting the input data to the enforcement of the output roles, passing through the role mining algorithms. We underline the unhandled issues, though essential to fully benefit from role mining in the different access control applications in each step.

Following our analysis two strands of work emerge. Under the first strand, we propose a bottom-up approach that extracts instances of the Net-RBAC model from a set of firewall rules. We intentionally reuse existing role mining techniques to leverage the achievements realized in this field. *Policy mining* is an extension of role mining that calculates roles, activities and views to meet network security requirements. Due to the fact that the three concrete entities in a rule are consistently structured, policy mining also generates a set of *abstract rules* that constitute the high level policy. Policy mining handles the problem of factorization of a three-dimensional matrix where traditional role mining is usually assimilated to a problem of factorization of a two-dimensional matrix. Furthermore, since large and medium networks are usually protected by more than one firewall, and policy mining algorithm designed for a single firewall, we still need a further processing of policy mining performed on each firewall to obtain a global network policy. In order to provide a complete automatic bottom-up framework for network policy mining, we handle the problem of integration of Net-RBAC policies resulting from policy mining over several firewalls. Proceeding in this modular way has several benefits. We can examine and analyze individually the configuration of each firewall apart, and detect intra-firewall misconfigurations. This could be performed on a regular basis to check the current configuration, without necessarily running the bottom-up process for the whole network. We propose a two-staged process. In the first stage, we unify the hierarchies of the abstract entities namely the roles, activities and views of all the firewalls. We assimilate the problem to the general mathematic problem of partially ordered set merging. In the second stage, we build the effective deployed network policy rules. We integrate the highly abstracted rules from the firewalls while checking several security properties. We detect irrelevance anomalies consisting of rules that never apply because they are enforced in a firewall not on the path between the source and the destination. We also detect inaccessibility anomalies due to inconsistency between the configuration of firewalls on the path of the same flow. The correctly deployed rules are aggregated into the network policy, whereas the detected anomalies are reported.

Under the second strand of work, survey of the post processing role mining tools reveals that assisting security administrator to leverage role mining results in actual applications has not received enough attention despite its crucial importance, and remains a weakness in the role mining contributions. Several efficient role mining algorithms have been proposed in literature [27, 32, 51]. However, their outcomes still require intensive human interpretation before leading to appropriate results. As far as we know no efficient solutions have been proposed to assist a security administrator in the task of leveraging the outcomes of a role mining process. In this thesis, we aim to provide automatic tools for the analysis, assessment and enforcement of the

generated roles. In particular, an automatic matching tool between roles from two sets is obviously needed for the assessment of role mining methods, as well as for the interpretation of the mined roles. 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 and to detect unhandled perturbations over the deployed configurations. We formally define the problem of comparing two sets of roles. We demonstrate that the problem is NP-complete. We present a greedy solution that tackels the 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.

Moreover, further role misconfiguration features may require new techniques to be detected automatically. For instance, the distribution of the permissions over the roles may become inconsistent through the evolvement of the assignment of users to the roles, leading to a set of roles that suffers from misconfigurations like unassigned roles, or hidden permissions in overlapping roles. We refer to these anomalies as shadowed roles. Through investigating the input data characteristics under realistic assumptions related to the role mining context, we demonstrate that there is a correlation between the misconfiguration problem of shadowed roles and the complexity of execution of our role set comparison approach. Then, we address the problem of shadowed roles detection by formally stating the problem and providing an algorithm that detects and reports shadowed roles in a given configuration of roles.

1.2 Outline of the Thesis

This thesis is organized as follows:

Chapter 2 surveys and classifies the approaches that address firewall management problems in literature, and underlines their limits. Then it motivates for a model based framework for firewall management based on two complementary top-down and bottom-up approaches that link the deployment level and the policy definition level. It reviews some technical definitions about the RBAC model and presents the Net-RBAC model. It explains the limits of RBAC in network security policy modeling and shows how Net-RBAC model overcomes them.

Chapter 3 is a comprehensive analysis of the role mining research field. It surveys the provided formal definitions of the role mining problem, and summarizes the most important solutions classified by their foundation techniques. It covers the addressed issues in the related literature and emphasizes the remaining unhandled issues.

Chapter 4 presents policy mining, an extension of role mining that applies to firewall rule and extracts the corresponding high level access control policy formalized with Net-RBAC model. Conceived to be flexible enough, it may extend almost any existing role mining technique to policy mining and Net-RBAC target model.

Chapter 5 completes the bottom-up approach for network security management by showing how to integrate the policy mining results from multiple firewalls protecting the same network in order to mine a high level policy for the network, while verifying some relevant security properties.

Chapter 6 aims to provide a security administrator with automatic assistance tools to leverage role mining and policy mining results. It formalizes and handles the problem of comparison of two sets of roles.

Chapter 7 deals with another issue related with role mining outcome analysis, namely shadowed roles detection problem. The chapter links this problem with the role set comparison problem, and proposes an algorithm to solve shadowing detection.

Chapter 8 presents the implemented proof of concept platform and provides experimental assessments of the proposed solutions in the four preceding chapters. The results show the feasibly of the bottom-up policy mining approach for a model based network security management. Experiments also evaluate the role set comparison tool and the shadowing detection tool and show how they could be used.

Chapter 9 closes the thesis and discusses some perspectives.

CHAPTER 2

Toward A Model Based Approach For Network Access Control Management

The size and complexity of many firewall policies along with manual configuration and management of the rule set generate tediousness and errors. Complex interaction of conflicting rules can conceal serious errors that compromise the security of the network or interrupt the delivery of important services. Current firewall configuration languages have no well-founded semantics. Each firewall implements its own algorithm that parses specific proprietary languages. The main consequence is that network access control policies are difficult to manage and most firewalls wrongly configured. Most existing tools dealing with firewall management problems focus on testing and debugging the policy, and require from the user the provision of a detailed set of test cases or queries, which can sometimes be as difficult as verifying the policy by hand and demands detailed knowledge of the potential vulnerabilities in addition to a significant investment of time and resources.

In this chapter, we survey the different firewall management approaches in literature and show their limitations. We emphasize the high potential of the high abstraction model based approaches for access control policies definition and management in networks. We propose a new approach that conceals two complementary top-down and bottom-up processes for a viable and efficient use of the high level models.We recall some technical definitions of the RBAC model and present the Net-RBAC model to be recommended for network security policies expression.

Chapter organization. Section 2.1, surveys the different approaches that address the firewall management issue in literature and summarizes the most important solutions

classified by their foundation techniques. Section 2.2 proposes the adoption of a model based approach for network security management within a framework that integrates top-down and bottom-up processes. Then it reminds readers of the RBAC model definition and shows how the Net-RBAC model fits better the requirements to network access control policy expression. Section 2.3 concludes the chapter.

2.1 Firewall Management Approaches in Literature

The management of firewalls is problematic due to the way their configuration develops. Usually, multiple parties are involved in the firewall management task including security administrators and project managers. Project managers request new authorizations according to project needs and rarely notify no longer needed authorizations such as an out of production web service. The network security administrator implements these updates on the firewalls, while trying to guarantee consistency with some security standards. She usually encounters several difficulties in this task: it is difficult for the administrator to know wether the necessary authorizations are already allowed or not, wether she has to update an existing rule or to create a new one and which firewalls have to be updated if there are many. The firewalls have to work in unison, and their configuration must take into consideration the topology of the network, which may also change. Thus the rule base grows uncontrolled and inconsistent. The administrator often uses many exceptions, which makes the configuration of the firewall error and conflict prone, or she uses too generic and permissive rules, which threatens the security of the organization resources. The situation is worse, given that the firewall configuration languages are vendor specific at a very low level and order sensitive. The order of the rules has an important impact on the access control policy and also on the filtering performance. Under these conditions, migration, updates, and delegation to a new administrator become real challenges. Each new request requires extensive investigation. When a problem such as an application which is supposed to work and does not arises it is difficult to identify why.

While the constructors focus mainly on enhancing the performance of the firewalls in term of speed, transparency, and capacity of filtering, enhancing the management of firewalls is still an academic concern. Different approaches that address firewall management problems exist.

2.1.1 Intrusive Approaches

Most practice-oriented papers provide methodologies to control and analyze the deployed access control policy in a given system by performing *penetration tests* against a firewall system. This intrusive approach consists in playing the role of a potential hacker and performing active tests by injecting packets and scanning ports to watch the behavior of the firewalls and evaluate the implemented rules. These tests could be conducted as *blind tests*, which are based on a vulnerability check list without requiring the specific environment description and rely only on the attacker's point of view so collaboration with the internal hosts is not allowed. Or they can be *design-oriented* tests, where the *pentester* interviews the engineers who have configured the firewalls about what they think their firewalls eventually protect. Then, he or she formulates a set of tests accordingly to verify the claimed protections.

There is a number of active firewall testing tools. For example, *SATAN* (software penetration tests tool) [22, 26] is a software that exploits the known flaws in widely deployed protocols and operating systems that can be blocked by appropriate firewall policies. Likewise, some firewall constructors propose dedicated hardware boxes that connect to the Intranet and probe the network to test the deployed routing and firewall policies (e.g. *NetSonar* [2] from Cisco). More recent, *Metasploit* Project [1, 36] offers a popular framework among the developers' community. It is a comprehensive computer security project that provides updated information about security vulnerabilities and aids in penetration testing. The open source Framework Metasploit is a tool for developing and executing exploit code against a remote target machine.

In practice, all the important government and corporation IT systems are pentested. Actually, this is the industrial way of handling the firewall management problem. However, this approach suffers from several limitations. It is difficult to conduct and it usually requires the services of exterior expertise. The envisaged scenarii can never be exhaustive, and the intrusive character may be harmful to the network normal functioning. Intrusive testing tools also suffer from scalability limitations and provide only statistical results in large Intranets. Besides, they can only detect the over assignment misconfiguration by finding packets that should have been unauthorized according to the policy. They can not detect the under-assignment misconfiguration, i.e. packets haphazardly blocked until users complain about the induced dis-functioning. Similarly, active tools are unable to handle spoofing attacks since they send packets and wait for responses to conclude if there is an anomaly or not. Moreover, misconfigurations are detected after the fact, which means that the IT System had previously remained vulnerable. Finally, these tools are dependent on the physical location of the tester in the topology so problems related to specific untested paths may remain undetected.

2.1.2 Offline Query Tools

Offline query based tools aim to help at understanding the deployed configuration of rules in a firewall or a system of firewalls by simulation. They take firewalls rules as input, and a description of the topology to design a simulated model that can answer a query such as [*can point a send service s to point b?*]. Mayer et al. [48] have presented one of the most important query based engines. In *FANG*, the network is modeled as a graph where the vertices are both the firewalls and the topology zones. Each vertex owns attributes and among them its nature and the IP addresses assigned to it. An edge relates two vertices if they have adjacent IP addresses. Firewall rules are modeled as k-tuples and attached to the firewalls that enforces them. This allows to support configurations of firewalls coming from different constructors in the same model. The queries are also modeled as attributes assigned dynamically to the vertices. When running a simulation of a query, the query is attached first to the vertex that has the addresses of the host group source of the query. Then, the query propagates through the graph. Each time it crosses a firewall, the corresponding rules are applied. A query may be divided into several subqueries during the simulation if for example a crossed firewall filters a subset of the concerned packets. The model is not based on the actual routing information and tests all the possible paths for a given query. This tool can simulate spoofing attacks by allowing to give a query an optional attribute *true-source* different from the host source. More recently, Marmorstein et al. has also proposed a query based engine tool in [46] that supports stateful firewalls also. The set of rules is represented as a *Multi-way Decision Diagram (MDD)*. In the DDM, each path represents a rule. The number of levels is variable and can be very high in a complex firewall policy.

Having the ability to cold-test the policy before deploying it reflects an improvement. But as this approach uses a simulation of the model and not the firewall and the network configuration themselves, its validity can be questioned. Assuming that the modeling is good, the difficulty to formulate the good queries still remains. While it is often easier to construct query tests than to inspect the rules manually, it can be difficult to create queries that test enough interesting behaviors and produce useful output.

2.1.3 Rule Structure Analysis for Misconfiguration Detection

This approach aims at discovering structural configuration errors in a firewall or a system of firewalls. It is grounded on the comparison and the analysis of the relationships between the filtering rules. It may detect conflicting rules and useless rules

in the initial firewalls configuration. Usually, the administrator is notified about the potential misconfigurations, and decides interactively if they are relevant or if they are false positives. Then from an initial firewall setup potentially misconfigured, she or he rewrites the rules subsequently to obtain an equivalent set of rules free from structural errors.

Al-Shaer et al have proposed such an approach for intra-firewall [6] and inter-firewalls [5] conflicts detection. They have provided a tool called *Firewall Policy Advisor* intended to assist the security administrator to ensure that a rule insertion, removal or modification does not engender conflicts with the other preexisting rules. They have modeled a firewall rule as a 5-tuple [*protocol, source_IP, destionation_IP, source_port, destination_port*] with two attributes: the order of the rule in the configuration, and the decision (accept/reject) taken by the firewall. They have classified and formally modeled the relationships between two rules as: disjoint if they never match the same packet, exactly matching, inclusively matching, partially disjoint, and correlated. They have assumed that any pair of rules is related by one and only one of these relationships. Nevertheless, we have noted that some rules may match both categories *partially disjoint* and *correlated* definitions. Then, they have defined a set of anomalies which are: shadowing (a rule never applies because all the packets it matches are matched by previous rules), correlation (conflicting firewall(s) decision for two rules that may match the same packets), generalization (a rule preceding another rule matches at least all its packets, and get a different decision), redundancy (a rule is not shadowed but removing it does not change the policy because rules preceding and following it match all its packets), and irrelevance (a rule is irrelevant if it does not match any traffic that may flow through the firewall where it is enforced). They have presented an algorithm for rule insertion that runs a state machine over a branch of the policy tree to discover if the insertion of the rule could generate any of the defined anomalies. This approach suffers from high computational time complexity. Moreover it is likely to generate a high rate of false positives due to the fact that the definitions of the considered misconfigurations are too generic.

The tool Mirage (MIsconfiguRAtion manaGEment of network security components) [7, 17] applies on firewalls as well as on IDS (Intrusion detection systems). It detects shadowing and redundancy anomalies of rules within the same device. The information about the topology and the routing is involved for inter-firewalls misconfiguration detection. This allows making more targeted comparisons, detecting redundancy and partial and total shadowing between rules along the path crossed by the packets. It allows also detecting misconnection anomalies when the decision taken by firewalls on the same path of a packet are conflicting, and irrelevance anomaly when a

rule handles packets that do not pass through the firewall of enforcement. Mirage assumes that the policy is closed, so implicit rules of rejection are also taken into account in the misconfiguration detection.

Yuan et al have presented the tool FIREMAN (FIREwall Modeling and ANalysis) in [69] to discover anomalies in individual and distributed firewalls. It implements an algorithm based on the *binary decision diagrams (BDDs)*. In addition to discovering structural inconsistencies and inefficiency problems, FIREMAN can also find violations of user-specified security policies.

Misconfiguration detection approach by firewall rule structure analysis could be useful to detect the conflicts when inserting, modifying or removing a rule from a firewall. However it is not a reliable solution for network security definition and analysis. It provides structural analysis of the implemented rules but no semantic analysis of the policy. It does not allow to understand the behavior of the firewalls whereas simulation query tools does. Besides, the definition of the anomalies to be detected is a sensitive problem and should avoid high amount of false positives. This approach requires greedy comparisons of rules, which arises scalability issues with the growing number of firewall rules. Finally, management of the notifications about misconfigured rules is still fulfilled at the implementation level.

2.1.4 Refactoring and Aggregation

Another approach tries to refactor the set of rules deployed on a firewall by rewriting the rules in a more compact and optimized way in order to enhance their readability. In most cases, refactoring involves aggregating rules. For instance, in [47], Marmorstein et al. have proposed a solution of firewall rules aggregation by classifying each of the source_hosts and destination_hosts into homogeneous groups. The primary objective is to rewrite the policy in such a way as to make the identification of anomalies easier. To achieve this purpose, their methodology extracts an *equivalence structure* from the deployed firewall(s) rules for the source hosts and the destination hosts, then classifies the hosts according to this structure, and last rewrites the rules with this structure. Finally, the security administrator would look at the obtained simplified policy and try to detect anomalies. To perform the host classification, they have defined an equivalence relation. Two hosts are *equivalent* if and only if they are *source equivalent* and *destination equivalent*. Two hosts are *source equivalent* (respec. *destination equivalent*) if and only if all the packets that have one of them as source host (respec. destination host) and differ only by the source host (respec. destination host) get the same firewall filtering decision. The firewall rule set is modeled by a *multi-way decision diagram*

(MDD). The *reduction* property of an MDD is used to compute the equivalent classes. We note that overlapping classes of hosts is not supported in this model. If a source or a destination host range of a rule overlaps with another host range in another rule, the rules are divided into three disjoint rules. After performing the host classification, the classes obtained are analyzed to detect anomalies. For example, if there are weird classes grouping improbable hosts together, this may come from type errors in the firewall configuration, which may lead to serious vulnerabilities. If some expected host classes are missing, this may inform about a shadowing problem in the configuration. This host classification approach may help rewrite the filtering rules in a more readable way. Moreover, by using routing information to classify hosts into groups, this approach could be useful with query based tools. The obtained classes may help in the definition of the queries. The commercial tool *Lumeta Firewall Analyzer* [3, 68] combines query based simulation engines with the host classification approach. It generates a comprehensive set of queries that reduce the amount of output data and makes the queries and their results easier to formulate and to understand. Although the authors state that it is fully automatic, the solution presented in [47] remains very subjective and lacks precision regarding its anomalies detection application. Moreover, the hosts with very different properties, but similar addresses, are gathered together since the host classification is based on the routing information and does not take into consideration the applications and services.

In [63], Tongaonkar et al. have proposed an aggregation approach that classifies source hosts, destination hosts and services too. Their methodology aims at generating equivalent flattened rules from the firewall configured rules. Then, they merge rules that have similar effects: first they group the services, hosts and protocols into various potentially overlapping classes. Then, they construct a merge graph, with weights for each rule or combination of rules and merge the rules containing the same class of objects. Finally they apply heuristics to choose the equivalent set of rules with the lowest complexity according a new metric for the complexity of a set of rules. The performed experiments show that their method reduces the size and the complexity of rules to get a debuggable and understandable policy.

Abedin et al. [4] have proposed to apply data mining techniques combined within tailored algorithms to perform a similar work of aggregation but in a different usage context. They do not process the firewall rules but the logs of the firewalls. The logs are collected on off mode or on streaming mode. The effective firewall rules are extracted from the logs in order to compare them to the original rules. Differences between the original rules and the rules mined from the logs mean that: either the policy is not well defined (as a result the effective rules are different from the policy rules), or the

firewall does not filter in the way required by the policy(default from the constructor), or there a problem appears in the logs (e.g. packets are usually malformed in the iptable linux logs), or the traffic in the log is not representative and does not cover all the rules. In this latter case, reordering the rules in the policy according to the traffic frequency should enhance the firewall performance. The paper handles stateful firewalls. Several tailored algorithms are used, each to extract a specific datum by parsing the logs and searching for its related packets (such as SYN-ACK packets that open a TCP connection). Thus, the solution lacks genericity and scalability.

The refactoring and aggregation approach offers automatic solutions and often rewrites the policy in a more efficient and readable manner. This may be of great help for administrators. However, these solutions do not target a well defined model of higher level policy, thus they may produce too much unstructured output to be useful. Moreover, the surveyed solutions does not leverage the existing data mining algorithms that may bring to them more scalability and efficiency.

2.1.5 Policy Management from High Abstraction Level

Most of the network management problems come from the difficulty of manipulating their low level languages. Network access control policy requirements are formulated in a human language first, then they are manually interpreted and implemented on the firewalls. Afterwards, the policy is updated and managed from the implementation level. Several contributions in literature suggest to perform all the definition and management tasks from a high level language, near to human language, and to translate the policy specifications via automatic tools to the deployment level in a top-down manner.

Haixin et al. [33] have proposed a top down *policy access control framework (PACF)* to manage automatically and dynamically the firewalls and screen routers configuration in *transit networks*. In these networks, the firewalls handle a high number of rules and have to be particularly efficient so that the loss of throughput coming from firewall treatments does not become more important than the loss coming from harmful and malicious sites. The firewalls have to protect the network resources and not the host machines in this particular application. The framework PACF consists in three levels of abstraction. The organizational access control policy (OACP) corresponds to the policy statement in natural language. The global policy (GACP) illustrates the refinement of the human language rules into a machine intelligible rules modeled as [*source destination action*] where the action is accept or reject. This policy model relies only on routing and does not take into consideration the kind of the service. At this level,

the order of rules is indifferent, the same as on which firewall each rule will be enforced. The third level of the framework is the local access control policy (LACP) that results from the automatic distribution of the GACP rules onto the firewall interfaces. The provided examples in [33] show only two types of rules *sources prohibited to send to anyone* and *destinations prohibited to receive from anyone*.

Zhang et al. [71] have proposed a high level firewall policy language *FLIP*. Unlike rule-oriented languages that impose to provide traffic details in the configuration, FLIP allows the administrator to define its high-level service oriented security goals by the specification of a set of domains, services, and policies related to each domain. The order of the rules does not matter at the high level policy definition. A *domain* is a set of users and/or machines and/or networks that have some access control rules in common. FLIP allows inheritance between sub-policies and gives the possibility to implement exceptions by overwriting parent rules by child rules. This tool is provided with a compiler to detect *domain definition* conflicts in the specified policy since domain overlapping is considered as a conflict. An algorithm is provided to translate FLIP to an Intermediate Language (IL) - a rule-oriented and order sensitive set of rules in a common packet format. This language is not much simpler than the deployment language. It demands high expertise to master the language, and a good knowledge of topology, IP addresses and services. Moreover the automatic translation supplies algorithms to distribute the rules on the appropriate firewalls.

Cuppens et al. [18] have provided a formal approach to specify and deploy a network security policy from a high level of abstraction. They have used an access control language based on XML syntax whose semantics are interpreted in the access control model *Organization Based Access Control (OrBAC)* [34]. They have shown how to use this language to specify high-level network access control policies and then to automatically derive concrete access control rules to configure the firewalls. To specify a network security using OrBAC semantics, the involved actors are assimilated to the model concepts. An entity that manages a set of security rules may be considered as an *organization* in the OrBAC model, thus, each firewall is modeled as an organization. Each host machine is considered as a *subject*, any implementation of a network service is an *action*, and an *object* is the content of a message and the related host destination. So on the concrete level, a permission is a triplet (subject, action, object) interpreted as the following: a host machine can use a service to send a message to another host-machine. The OrBAC model structures the concrete entities: subjects, actions, and objects into structures of respectively roles, activities, and views. At the policy definition stage, the administrator may define his own abstract entities by manually assigning concrete entities to them in order to get a more compact policy. Automatic assignment of

concrete entities to their relevant structures basing on their attributes gives another facility. For example, the administrator can define a role and provide a list of automatic inclusion and exclusion conditions for it. All the source-hosts whose attributes verify these conditions will be automatically assigned to that role. This approach considers that using the duality of permission and prohibition in network security policies induces sorting problems, so they define policies based only on permissions within a closed default policy. Derivation of the concrete firewall rules falls in two steps. A first XSL transformation translates the OrBAC policy to an intermediary multi-target language using an XML syntax. This step also specifies which rule will be managed by which firewall(s). A second XSL transformation translates the intermediary language to the specific language(s) of the firewalls (e.g. NetFilter).

Similarly, Zaborovsky et al. [62, 70] have proposed a top-down approach that adopts the OrBAC model for network policy specification. They have focused on the integration of an access policy with a network environment within the top down enforcement process. The network environment is modeled in the paper in term of data channels. A data channel is a distinguishable part of interaction between two initially independent subsystems of the network. A *network channel* relates two interacting systems characterized by their IP addresses. An *application channel* ensures interaction between two systems characterized by the application parameters. For example, a HTTP application channel could be characterized by an URL in a HTTP request, a user name, a file name, or a pair of users and of files. All the requests and responses with these characteristics belong to the same applicative channel. The refinement process assigns a state of open or close to each channel according to the policy specification.

Preda et al. [54, 55] have established a formal frame for the validation of the deployment of an access control security policy specified using the OrBAC model. They have developed a set of algorithms that realize the translation of an OrBAC set of rules into specific device configurations. The translation brings about the deployed rules by introducing the routing and the topology information within successive refinements. Each transition is proved and validated formally by the B-method. Routes are calculated from the topology information assuming an OSPF routing algorithm. The output results from the concrete rules directly enforced on the appropriate devices. In addition to the filtering functionalities, this tool leverages the concept of *context* from the OrBAC model to allow the enforcement of IPSec tunnels and authentication procedures. To that aim, the security capabilities of each device in the network must be provided as input. Moreover, this approach tells which security properties are ultimately verified. A security property is generally expressed on a more abstract level than the security requirements. It may still not be verified after the deployment of all

the security rules. Indeed, since topology is provided independently from the specified policy, then the deployment of the policy may reveal unfeasible for reasons such as insufficient firewalls number or bad positioning of the firewalls in the topology, or nodes that do not support the IPSec functionality, etc. Hence, Preda et al. have shown how to verify several security properties during the deployment process. In particular, some of the most interesting application-independent properties the policy deployment process should verify are:

- Completeness of deployment: if a network path from a subject to an object of a rule exists and the security devices belonging to this path have the right functionalities regarding the context, then the security rule may and will be deployed.
- Accessibility and Inaccessibility: for each rule, all the firewalls on the path between a subject and an object must allow the action the subject is supposed to realize on the object. The inaccessibility property lies in the fact that if no permission between a subject and an object for an action is specified by the policy, then there will be no open path.
- All the traffics are regulated by firewalls.
- Integrity and confidentiality property: this property is related to the establishment of IPsec tunnels. It ensures that extremities of the IPsec tunnel have the encoding capability. Moreover, particular IPsec configurations may include recursive encapsulation of traffic on a path.
- Authentication: in the OrBAC policy, some actions may require an authentication context to be achieved. This property verifies these cases by providing a variable that records the actions realized by the subject concerned with the authentication and by imposing a workflow constraint.

Managing the policy from high abstraction level provides the administrator with the necessary hindsight to fully control the network security and makes the policy definition and configuration easier. The approaches based on the OrBAC model offer an intermediary abstract level between the policy requirements formulated in a human language and its equivalent set of firewall rules. They provide well defined semantics to network security policy specification, which guarantees the portability of the policies. Moreover, the definition of a global security policy ensures that a system of firewalls will be configured in a consistent manner. Furthermore, by using a high level model such as OrBAC, the network access control policy could be directly integrated as a part of the organizational access control policy. The existence of reliable automatic

tools that handle the automatic deployment of the policies increases the potential of such an approach.

Nevertheless, the adoption of a high level model in the network access control policies management may imply to throw away the already deployed rules in the firewalls and start from scratch the specification of the whole access control policy in a top-down manner.

2.2 Top-Down and Bottom-Up Framework for a Model Driven Network Security Policy Management

Enhancing the management of firewalls is an important academic concern that has been addressed from different perspectives. Most of the proposed approaches aim at testing, debugging and analyzing the behavior of the already configured firewalls. They are usually difficult to conduct and require from the user the provision of a detailed set of test cases, which can be sometimes as difficult as verifying the policy by hand. The rules refactoring approach offers automatic tools that rewrite the configured firewall rules in a more readable and optimized way, but the existing solutions do not target a well defined model of higher level policy and do not leverage generic data mining techniques. Adopting a high abstraction model for the policy definition and management seems to be the best alternative. It has several benefits. First, a standard model specifies the network security policy with clear semantics to guarantee portability. Second, by structuring the data in a syntax close to the natural language, the policy becomes more compact and easier to manipulate, so a security administrator can benefit from the necessary hindsight over the global policy to control it and update it safely. Moreover the structure is usually less variable than the concrete entities, so the policy becomes more stable. Finally, defining the policy at a high level released from topology constraints and then deploying it through an automatic top-down process allows a consistent configuration of multiple firewalls. We have shown in the last section 2.1.5 that reliable tools exist to ensure the top-down translation of a high level policy to the deployment level. Yet, the greatest stumbling block to the adoption of high level models in network security rests in the fact that no solutions have been proposed to address the rules already deployed on the firewalls. This implies to throw them away and start from scratch the specification of the whole access control policy.

To bypass this problem, we advocate an automatic bottom-up approach that parses the configured rules in the firewalls and leverages data mining techniques to automatically reach an instance of the high level model corresponding to the deployed policy. The bottom-up and the top-down approaches should be used together in a cyclic manner. The bottom-up approach is used to reach a general high level programming language from the already configured firewall rules. Then the policy is updated or corrected at the abstract level before being deployed to the concrete level again with the top-down approach.

All the surveyed papers that specifically deal with the application of access control models in network security have chosen the OrBAC model. There are no preexisting approaches that deal with the configuration of OrBAC model in a bottom-up manner, and in particular, no solutions that aim at extracting an OrBAC policy from the configuration of firewalls. Meanwhile, a bottom-up approach so called *role mining* has been proposed for the RBAC model configuration and has gained interest in the last few years. This approach automates the role engineering task by using existing data mining techniques to extract the roles from the deployed access control lists (ACL) of direct user to permission assignment.

2.2.1 The Role Based Access Control Model

In the last decades, the *Role-Based Access Control (RBAC) model* [23, 56] has become the dominant model for access control in both commercial and research fields. The following definition is extracted from the NIST standard for RBAC [23]. We present the basic model $RBAC_0$ without considering sessions. The standard defines also other RBAC model versions with extended features such as the role hierarchy, the separation of duties functionality and the sessions. These features have been seldom taken into consideration in the bottom-up configuration approach which questions as in this thesis.

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

- *U, ROLES, OPS, and OBJ, 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 role-to-permission assignment relation*

- $assigned\_users(R) = \{u \in U | (u, R) \in UR\}$, *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.*

In the RBAC model, the organization access control system is viewed as a set of users U who need access to some resources to perform their tasks. For that, the users need to be granted permissions from the $PRMS$ set. A permission is characterized by an operation over an object, but it is usually considered as an indivisible entity. Users are dynamic entities, so it is safer and more efficient to structure them into higher level of abstraction entities called roles. Intuitively, a role can be viewed from two perspectives: a set of permissions frequently assigned together, or a set of users granted a shared set of permissions. The concept of role makes the access control system more compact, structured and stable, compared to the direct user-permission assignments. Roles are defined by security experts according to the organization functioning, a task called *role engineering*. They may represent job functions for example. The RP relation represents the link between the $ROLES$ and the $PRMS$, and the UR relation the link between the $ROLES$ and the U. The permissions are granted to the roles and no longer directly to the users. Users are assigned to roles and gain permissions through these roles.

In some cases, supplementary direct user-permission assignment is allowed to express exceptional authorizations, inspire of the adoption of an RBAC configuration in the organization. Instead of creating an artificial role for each remaining single rule, the RBAC configuration is usually denoted thus: $RC = (ROLES, UR, RP, DUPA)$ with $DUPA$ being the relation of the direct and unstructured user-permission assignments. Moreover, if a hierarchy relationship exists between the roles, it is denoted $RH \subseteq ROLES \times ROLES$, RH being an order relation between $ROLES$. The RBAC configuration with hierarchical structure of roles is denoted: $RC = (ROLES, UR, RP, RH, DUPA)$.

2.2.2 The Net-RBAC Model

The RBAC model has proven its ability to bring a substantial enhancement of performance and productivity in a wide area of access control applications such as physical security package, environmental security, database systems, enterprise resource planning systems, workflow systems and operating system security [28]. However, when it comes to network security, we find that RBAC is not well suited to express network filtering rules. From the perspective of the basic RBAC model, security rules are considered to follow the pattern: [*a user u is allowed access to permission p*]. The users,

structured into roles, are the central entities in RBAC. They are expected to be the active entities that require authorizations in the system.

When we focus on the structure of a network access control rule, it is fashioned as follows: [*allow source_host sh to send service of type s to destination_host dh*] where *sh is an IP address or a panel of addresses* that send packets of service *s defined by protocol, source_port, and destination_port*, to *dh which is also an IP address or a panel of addresses*. If we assimilate the *source_host* to user, the *service* to operation and the *destination_host* to object according to the terminology of definition 1, then we note that in this pattern, the three entities are semantically on the same level of importance from the network access control perspective. Thus it is not appropriate to structure only the users and not the operations and the objects.

The OrBAC model has proved that it fits the network policy requirements inherently [18], mainly because it handles ternary rules and structures all the three concrete entities involved the security rules. However, OrBAC is a sophisticated and expressive model that defines several concepts such as the organization and the context. These concepts may be useful in the top-down network policy definition but risk to introduce too much complexity to the bottom-up approach.

In this view, we advocate the Net-RBAC model, an extension of the basic RBAC model to structure the three concrete entities into higher level of abstraction entities equally and a simplification of the OrBAC model at the same time.

Definition 2. *Network RBAC: Net-RBAC*
A Net-RBAC configuration denoted $NSRC = (ROLES, UR, ACTIVITIES, OPA, VIEWS, OBV, RAV)$ *is characterized by:*

- U, $ROLES$, OPS, $ACTIVITIES$, OBJ, *and* $VIEWS$ *which are the sets of users, roles, operations, activities, objects and views*
- $UR \subseteq U \times ROLES$, *a many-to-many mapping user-to-role assignment relation*
- $assigned\_users(R) = \{u \in U | (u, R) \in UR\}$, *the mapping of role* R *onto a set of users*
- $OPA \subseteq OPS \times ACTIVITIES$, *a many-to-many mapping operation-to-activity assignment relation*
- $assigned\_operations(A) = \{op \in OPS | (op, A) \in AD\}$, *the mapping of activity* A *onto a set of operations*
- $OBV \subseteq OBJ \times VIEWS$, *a many-to-many mapping object-to-view assignment relation*
- $assigned\_objects(V) = \{obj \in OBJ | (obj, V) \in OBV\}$, *the mapping of view* V *onto a set of objects*

- *$RAV \subseteq ROLES \times ACTIVITIES \times VIEWS$, a many-to-many-to-many mapping of role-to-activity-to-view assignment relation.*

Definition 2 adds two new abstract entities: *activity* which structures the operations, and *view* the objects, in the same way as *role* structures the users. More importantly, the fact that the three concrete entities are structured induces the definition of *abstract rules* represented by the *RAV* relation. An abstract rule takes on the form [*role r is granted to perform activity a over view v*]. The concrete access control rule involving a given user u, operation *op* and object *ob* exists if u is assigned to a R role, *op* to an A activity, and *ob* to a V view, and the triplet (R, A, V) belongs to the *RAV* relation.

2.3 Conclusion & Key Unhandled Issues

In this chapter, we have surveyed the existing approaches that address firewall management problems, including penetration testing, offline simulation based query tools, and misconfiguration detection. All these approaches aim at analyzing the behavior of already configured firewalls but they are usually difficult to conduct. Some approaches proceed by refactoring the rules deployed on a firewall by rewriting them in a more readable and optimized way. But they do not target a well defined model of higher level policy. Adopting a standard high abstraction model is likely to make firewalls configuration and management much simpler and safer.

We have recommended an approach combining cyclic top-down and bottom-up processes for network security management grounded on the Net-RBAC model. This model guarantees portability, interoperability, integration of the network security policy in the global organization policy, and facilitates the information transmission to new security administrators. While some interesting work has been done for the top-down approach, there are no preexisting bottom-up approaches for the configuration of a Net-RBAC policy from the deployed rules on firewalls. In this thesis, we tackle with this problem. We are motivated by recent achievement in role mining field for RBAC model configuration.

CHAPTER 3

Role Mining

Role mining points to a vivid research field that has inspired a number of solutions and shown important advances in a relatively short time. In this chapter, we survey this area of research. We provide the readership with keys and enough technical background to understand the context and the prevailing contributions achieved during the last few years. We define a complete and realistic business process that gathers all the issues related to role mining and necessary to leverage it in practice. We analyze the different process components from collecting the input data to the enforcement of the output roles, passing through role mining algorithms. For each step, we explain the problem and its constraints, then we provide keys to the chief solutions in literature with enough technical details to understand contributions, and we underline the unhandled issues however necessary to fully benefit from role mining in access control applications. We also survey the new market of role mining. We end up by questioning current achievements along with chronological evolution of the topics of interest, research perspectives and future trends.

Chapter organization. In section 3.1, we present the basic role mining process that aims to extract structured roles from a given user to permission assignment relation. Afterwards, we present an extended role mining process that covers more exhaustively the different issues addressed in the related literature. The following sections deal with the different steps of this process; Section 3.2 raises the issues related to the input data and preprocessing this data for role mining. Section 3.3 deals with the main role mining process. It surveys the provided formal definitions of the role mining problem, and summarizes the most important solutions classified by their foundation techniques. Section 3.4 is dedicated to the survey of the different assessment methods of role mining results in addition to the post processing tools. Section 3.5 summarizes the different research interests and identifies the future trends. Section 3.6 concludes the chapter and proposes further research perspectives. It emphasizes some key research directions that will be treated in the remaining part of this thesis.

3.1 Overview of Role Mining Process

This section provides technical background to understand the context of role mining problem and the notations usually used to schematize the problem.

3.1.1 Basic Role Mining Process

Role mining is the discipline of automating the definition of roles from the deployed set of access control rules assigned to the users, in order to configure an RBAC state in a given organization. Starting from a set of users, a set of permissions, and a direct User-Permission-Assignment relation *UPA*, the role mining process calculates a set of roles and a compatible assignment to users and permissions to these roles. This assignment must be done with regard to the initial *UPA*, and should at least guarantee two fundamental principles: provisioning and security. Provisioning means providing each user with access to the necessary resources to carry out his tasks in the organization. Security means forbidding unauthorized users to access to extra-permissions. Moreover, the target RBAC state should present some intuitive properties to provide the expected enhancement to the business process. The configuration of roles should be simpler to manage than the initial direct user-permission assignment. Minimality and interpretability are keys to simplicity [24]. In addition, it should provide generalization facility to support the evolution of users and permissions.

Boolean matrices are generally used to represent the data manipulated in the role mining process. Thus we introduce an algebraic representation of the entities involved and the relationships between them.

Algebraic Representation of the Role Mining Problem

For the sake of simplicity, and in order to unify the representation of the inputs and outputs of role mining, the entities involved are represented with Boolean matrices as follows: Given m users, n permissions and k roles (i.e.,$|U| = m$, $|PRMS| = n$, $|ROLES| = k$), the *UR* user-to-role mapping can be represented as an $m \times k$ Boolean matrix where a 1 in cell $\{ij\}$ indicates the assignment of role j to user i. Similarly, the *RP* role-to-permission mapping can be represented as a $k \times n$ Boolean matrix where a 1 in cell $\{ij\}$ indicates the assignment of permission j to role i. Finally, the *UPA* user-to-permission mapping can be represented as an $m \times n$ Boolean matrix where a 1 in cell $\{ij\}$ indicates the assignment of permission j to user i. If a hierarchy relation *RH* exists, then it is represented as a $k \times k$ matrix where a 1 in cell $\{ij\}$ means $role_i$

is a subrole of $role_j$. The relationship between the *UPA* and the *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 denoted* $A \otimes B = C$ *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 should have $UPA = UR \otimes RP$. The interest of this representation lies in the fact that a user i will be assigned to a permission j only if at least one of the roles to which he is assigned gets permission j. The role mining process tries to approximate the above decomposition: $UPA \approx UR \otimes RP$ with *UR* and *RP*, the user-to-role and the role-to-permission assignment relations resulting from role mining, and *UPA* the deployed user-to-permission assignment relation given as input to the role mining process. The deviation may come from different causes: the initial *UPA* not fully structured because of noise or exceptions, or the role mining process relaxing the decomposition constraints and allowing a controlled deviation to enhance the performance of the algorithm used for technical considerations. We refer to such errors as *false negative* if we have 0 instead of 1 and *false positive* if we have 1 instead of 0 in the resulting $UPA' = UR \otimes RP$ in comparison with the initial *UPA* matrix. Finally, in cases where the hierarchy of roles is supported by role mining process, decomposition becomes $UPA \approx UR \otimes RH \otimes RP$.

To illustrate this point, we imagine the example in figure 3.1 with *The Simpsons* characters. In the example, the initial *UPA* is modeled as a 6 users × 9 permissions Boolean matrix. We propose a decomposition of *UPA* into two matrices: a user to role assignment matrix and a role to permission assignment matrix then we define five roles.

Even with such a small set of data, we notice that there are different ways to define the roles. For instance, we could create a special role for the permission *watch TV* and assign it to every user. We could also represent explicitly the inferring hierarchy in this configuration of roles: since the role *Cool People* is always assigned to users who have also the role *Kid*, we could define it with both its current permissions and the permissions of *Kid* and consider it as a sub_role, or a specialized_role of the role *Kid*. However, this does not function if, in practice, this role has not been created only for *Kid*. For instance, the organization could intend to hire the new user *Marge Simpsons* and assign it to the role *Cool People* but not to *Kid*. In a real application, with hundreds or thousands of users and permissions, the number of possibilities of role definition would be much higher. This is why role mining optimization criteria and objectives are very important, since they are keys to get intelligible roles, adequate

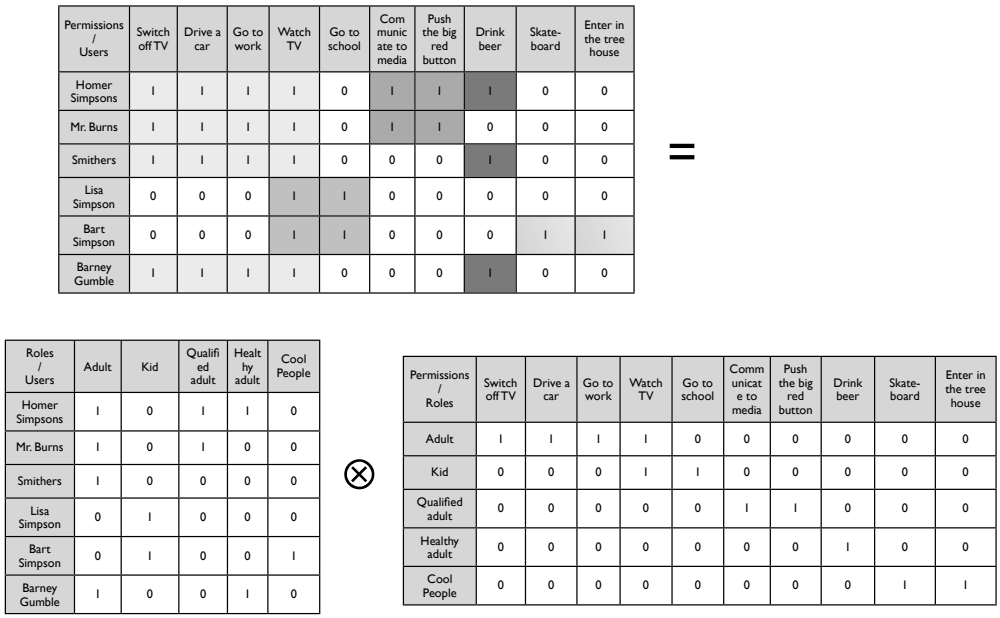


Figure 3.1: Example of Algebraic Representation of an Access Control Relation

for the targeted application. Besides, human involvement is still required to leverage the roles because they know what every role is intended for, whereas a role mining algorithm handles only the current deployed rules.

3.1.2 Extended Role Mining Process

The initial idea driving role mining was to automate the generation of the roles from the existing *UPA* only, and this by leveraging existing data mining techniques. The targeted output was considered to be an exact match $UPA = UR \otimes RP$. Several years of research in this field along with trials to move role mining techniques from theory to practice, with the ambition to satisfy different contexts of application with different requirements, has raised multiple related issues. Complexity is no longer concentrated on the sole aggregation of users and permissions into roles. Pre-processing and post-processing modules appeared to be necessary in a multitude of cases. The role mining process itself should be tunable and respond to evolving requirements. In this chapter, role mining process is modeled as presented in figure 3.2 to cover the various issues related to role mining and dealt with in literature.

Optional elements, usually dependent on the application case are drawn with dashed lines. For instance, the input data to the role mining process may involve additional *Top Down Information* (*TDI*). Several preprocessing operations are conceivable to

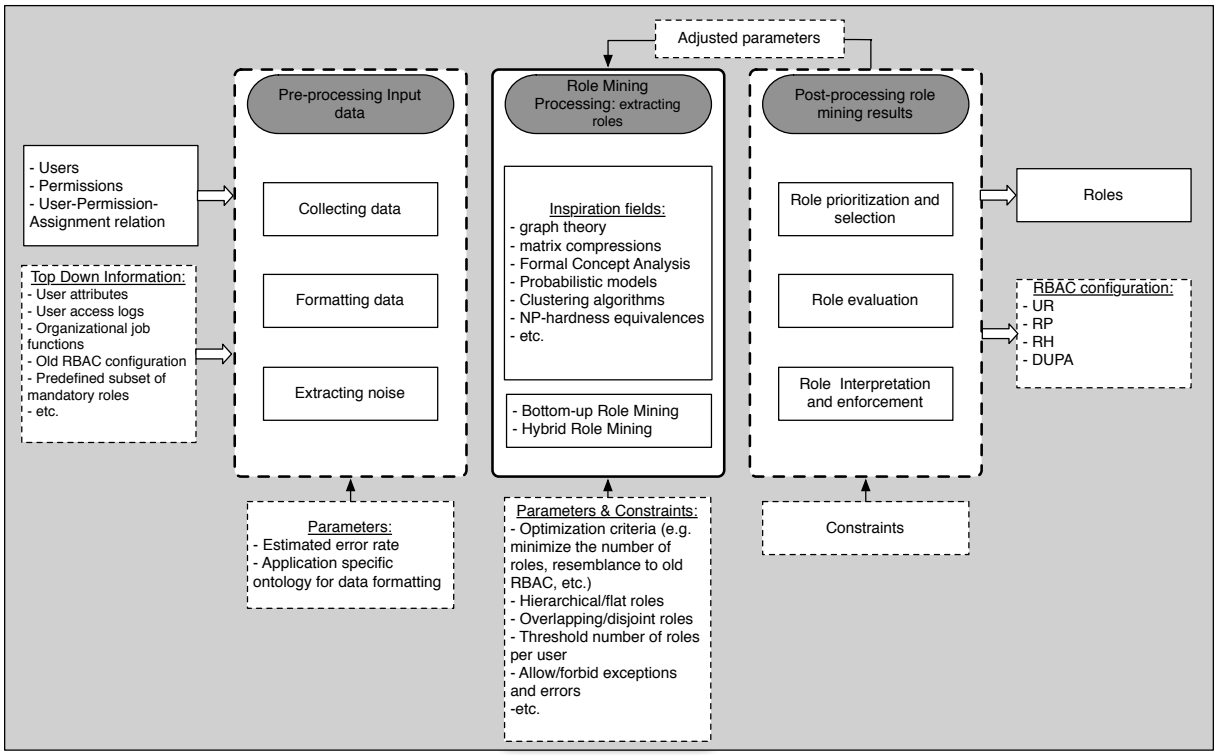


Figure 3.2: The Extended Role Mining Process

obtain more relevant input data, such as data cleansing to discard noise and exceptions because they can be distort the calculation of roles. In many cases, role mining is tunable and supports different kinds of user constraints. Likewise, leveraging the results obtained from role mining may require assistance from post-processing tools, and even iterative role mining. Finally, the output of the process may be tunable according to the input data characteristics and the user requirements to support role hierarchy RH and exceptional direct user-to-permission assignments $DUPA$ in addition to structured UR and RP. We detail all these issues sequentially with the solutions proposed in the following sections, starting from the input data pre-processing via role mining tools, ultimately examining the assessment and exploitation of the resulting roles and RBAC configurations.

3.2 Preprocessing Stage

3.2.1 Input Data Nature

Input data of the role mining process contains basic mandatory data, and some optional data (figure 3.2). By mandatory data we mean bottom-up information. It consists in a set of users, a set of permissions and the user's permission assignment relation. It is usually presented as a Boolean matrix as explained in the previous section. Though generally considered as a non-problematic part in role mining literature, we underline some issues related to basic input data. First, in some cases, collecting this datum is not a trivial task. The predominant application of RBAC targets virtual user account application. However, when the intended application field involves more abstract actors such as in network security or operating systems, the definition and the identification of the users and the permissions may be challenging. Second, additional bottom-up information should be utilized. For example, to our knowledge, several role mining papers have suggested but never dealt with the collection and exploitation of the history logs of user's activities to detect active, important, exceptional and misused authorizations.

In figure 3.2, the optional input data is denoted *TDI* for *Top Down Information*. It includes all the provided business information that may tune the role mining process to generate more adequate roles. Despite being for long time considered as sheer bottom-up approach, current tendency converges to the incorporation of top-down data within the bottom-up process, which may lead to more meaningful and maintainable roles [28]. Most of the recent contributions since 2008 have taken into account such kind of input data [15, 50, 25, 12, 49], and proposed hybrid role mining approaches. There are several types of *TDI*. First, business organizational information, such as organization into multiple geographic sites, departments, job titles, etc. This information may be represented as attributes of the users, together with other personal information such as name, salary, age, qualification, etc. The integration of this kind of information may be imposed by the organization (e.g. delimit role perimeter to department frontier) or recommended by the role mining tool designers in order to get intelligible roles. Another kind of information rests with permission weights. In fact, in some cases different permissions do not hold the same importance in the system. Such cases were mentioned in several papers, and specifically tackled in [43]. The weights assigned to the permissions can tune the role mining process. For example, if approximation errors are tolerated by the role mining algorithm, these errors should never affect permissions with high weights. Besides, organizations soliciting role mining services may have an old RBAC configuration altered by successive updates that needs to be revised. The administrators may prefer to take into account the old RBAC state in the RM

algorithm. Moreover, organizations have the possibility to impose a subset of roles, already defined by role engineering process for example, to be included in the puzzle of roles generated by role mining process. Using *TDI* in role mining raises two important issues. The first on: how to choose which information enhances the RM process or not. Indeed, using all the information provided by the organization might be detrimental to role mining process. The second issue: how to integrate the *TDI* information within the bottom-up role mining tool. We deal with these issues in the following section.

3.2.2 Preprocessing Input Data

Deciding which TDI to keep for RM

Frank et al. [25] propose an entropy-based method to select which user attribute should be kept for role mining. Attributes are prioritized by their ability to reduce the entropy when their correspondent users are clustered. Entropy-reduction is calculated for each attribute, according to a dedicated probabilistic model for the role mining system. Then, attributes are weighted by their entropy-reduction when integrated in the role mining algorithm. The authors in [52] reuse this technique, but evidence a clue deriving from the entropy-reduction measure: it gives advantage to attributes with a high cardinality of possible values. In practice, this kind of attributes is not significant for role mining. For example the attribute *name* of users has a high cardinality but no impact for the mining of roles. This is why they propose a bi-objective method to prioritize the attributes by both minimizing the cardinality and maximizing the entropy-reduction. In a parallel direction, other contributions, such as [15], integrate the *TDI* selection within the role mining process itself, and not in the pre-processing stage.

Extracting Noise from the Input Data

In early role mining contributions, *UPA* has been considered to be error free. The target of role mining has been to extract roles so that *UR* and *RP* be an exact decomposition of *UPA*. In practice, the initial user-permission relation usually suffers from errors corresponding to noise. There are two kinds of noise. Over-assignment noise means that a user is assigned to a permission erroneously. This usually happens when a user changes his job position, and security administrator omits to revoke his old permissions no longer needed. Under-assignment is the opposite: users do not have access to permissions necessary to carry out their tasks. Running role mining over noisy data leads to an over-partitioned set of roles. Different fragments of the roles are generated

instead of the actual roles. Molloy et al. [52] notice that, in most cases, actual roles are structured hence represented by strong patterns in the *UPA*, while noise is random constituting thus the marginal values in the *UPA* relation. Thus, they propose to use matrix rank reduction techniques in order to extract noise coming from over-assignment. They test different existing rank reduction methods. For example, using the Singular Value Decomposition SVD [37], they decompose the *UPA* into its eigenvector and calculate the corresponding eigenvalues. The eigenvectors are assimilated to the directing patterns in the initial matrix, and the respective eigenvalues indicate the strength of each eigenvector in the construction of the initial *UPA* matrix. Thus, they keep only the eigenvectors corresponding to the largest eigenvalues to recompose the matrix UPA^* which will be an approximation of *UPA*.

This operation discards the weaker patterns assimilated to noise, and solely keeps the stronger ones assimilated to the signal. Molloy et al. [52] have solved two issues raised by this approach. First, they use thresholds in their algorithms to decide on the amount of discarded data and set it automatically. Second, a possible problem with this kind of rank reduction methods arises the recomposed matrix UPA^* not Boolean, needs supplementary processing to become Boolean before entering the role mining main process (figure 3.2). Some other role mining techniques such as *HierarchicalMiner* [49] and *HP Role Minimization* [21], try to handle noise within the role mining process itself. They allow direct user permission *DUPA* in the resulting RBAC configuration, and expect that this will contain the initial assignments resulting from both errors and exceptions. However, [52] shows that running role mining on preprocessed *UPA* and extracted noise is more efficient than allowing divergence in the resulting decomposition. There are still other remaining issues about preprocessing the *UPA* to extract noise. Indeed, this method fails to detect structured noise, i.e. noise that forms strong patterns and can be wrongly identified as standalone roles. It does not efficiently manage under-assignment noise too. Finally, it is difficult to automatically distinguish exceptions from noisy assignments. We still need human expertise to validate the preprocessing stage.

3.3 Role Mining Processing: Extracting Roles

3.3.1 Parameters, Constraints and Optimization Criteria

Role mining process is tuned by different constraints. Some may come from clients, or designers of the role mining solution, and others from feasibility consideration imposed by technical limitations of the adapted solutions. Some constraints also involve

the input data. We assemble all these considerations under the label *Parameters & Constraints* in figure 3.2. Starting from a flat discretionary and scattered set of rules, Role Mining generates a structured RBAC state, or a set of roles that will permit to configure such a state. The first characteristic about the final state is compactness. Thus, intuitively, the first optimization criteria would be minimality. Different possibilities are considered. Minimizing the number of roles would give a compact RBAC state, but the roles would not be necessarily intelligible and evolving. Minimizing the number of assignments of users to roles and/or permissions to roles is likely to simplify the management cost. Moreover, we can consider minimizing the number of hierarchy levels. In some cases, the number or roles or hierarchical levels is explicitly specified.

Hierarchical relationships between roles may be requested, allowed or forbidden. Another important requirement would be to allow or not multiple-assignments: assigning a user to multiple roles, or assigning a permission to multiple roles, which leads to overlapping roles, or both. Also, depending on the assumptions about the input *UPA*, exceptional *DUPA* could be allowed or not. In some cases, the organization aiming at performing role mining already benefits by an old RBAC system needing to be revised. In such a case, the organization could wish to keep the new RBAC state as close as possible to the old well-known one. Similarly, the organization may want to impose a subset of roles for functional reasons, and extract the rest of roles using role mining. Other constraints may concern the algorithm performance, notably its scalability and running time.

3.3.2 Formal Role Mining Problem Definitions

The previous sections have shown that there are different use cases for role mining, and a multitude of divergent requirements. Hence, no consensus about the formal role mining problem definition prevails. The earliest contributions solve an implicit role mining problem. The first explicit formal definition for the *role mining problem* is given in [65]). Vaidya et al. suggest minimizing the number of generated roles as the optimization criterion.

Definition 4. *Role Mining Problem RMP*
Given a set of users U, a set of permissions $PRMS$, and a user-permission assignment UPA, find a set of roles, $ROLES$, a user-to-role assignment UR, and a role-to-permission assignment RP 0-consistent with UPA and minimizing the number of roles.

A given user-to-role assignment UR, role-to-permission assignment RP and user-to-permission assignment UPA are δ-consistent if and only if $||UR \otimes RP - UPA|| \leq \delta$.

0-consistent corresponds to the exact decomposition. Several variants of the *RMP* problem follow. δ-approxRMP [65] allows a difference between *UPA* and its recomposition from $UR \otimes RP$ less than a threshold δ. $MinimalNoiseRMP$ [65] assumes that the number of roles is set as input parameter and tries to minimize the error threshold. The *Edge RMP* [42] targets to minimize the number of assignments, instead of the number of roles. Another variant is the *Minimal Perturbation RMP* [66] that proposes a bi-objective optimization by minimizing the number of roles while keeping as close as possible to a known previous RBAC state. It has been established that all these problems are NP-Hard. Zhang et al. [72] present a similar problem whose objective is to minimize both the number of user and the number of user-to-role and permission-to-role assignments.

The *inference role mining problem* presented in [24] focuses on the problem from a more generic perspective, covering the issues presented in figure 3.2. In all cases, role mining assumes implicitly that an underlying RBAC configuration exists in the deployed *UPA*. Thus role mining aims to reveal this underlying configuration. This definition provides a unified view of bottom-up and hybrid role mining. Unlike other definitions, it does not give an optimization objective function, which may suggest the way to solve the problem. Thus, it frees creativity to devise an algorithm or an objective function for this problem. Moreover, there is no obvious quality measure, which signs its flexibility and its capacity in containing any other definitions.

Definition 5. *Inference Role Mining Problem*
Let a set of users U, a set of permissions PRMS, a user-permission relation UPA, 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^*.

3.3.3 Algorithms and Solutions to Extract Roles

Role mining is mainly about leveraging already existing data mining techniques to extract roles. So, it has explored and borrowed solutions from different fields. Besides, to deal with changing requirements and contexts of application, we find solutions with different objectives. The solution should be tied to the input data, the constraints and the adopted problem definition. Here, we summarily present the most important role mining solutions classified by their inspiration fields.

Matrix Compression Algorithms

With regard to the algebraic representation of role mining problem, the matrix compression algorithms seem to be a direct application to solve the problem. Indeed, several matrix compression algorithms use decomposition to identify the important patterns in a matrix. Since RBAC is a compression of the access control rules *UPA*, from some perspective, the decomposition should correspond to roles. The general schema of the problem solved by such decomposition algorithms is: given M, find A and B so that $M = A \times B^t$ under some proper constraints. If we assimilate M to *UPA*, A to *UR* and and B^t to *RP*, we map the role mining problem to this compression schema. There are several different compression algorithms. The *Singular Value Decomposition (SVD)* [37] is one of the most used and simplest one. It decomposes a matrix into orthogonal matrices. In role mining projection, this means that a user will be assigned to only one role and roles are not overlapping. But these conditions are not usually required in practice. In addition, the matrices obtained are real and not binary, and need post-processing to be transformed to binary by approximation methods. *Non Negative Matrix Factorization (NNMF)* [53], another decomposition algorithm guarantees that the two resulting matrices from the decomposition are positive, which makes the binary transformation easier, but this algorithm suffers from instability. Indeed, it is an optimization algorithm that may give different solutions for a given instance, depending on the initialization of the algorithm. Thus, it should be run several times in order to keep the best solution. The *Binary Non Orthogonal Factorization (BNOF)* [38] provides a binary decomposition, allowing overlapping roles i.e. roles sharing permissions in common, but a user is assigned to only one role at most, not always a tolerable constraint. Matrix compression algorithms may also provide co-clustering capabilities, which can be useful to integrate *TDI* in the role mining process. For instance, the *Collective Matrix Factorization (CMF)* [58] is an algorithm used to synchronize the decomposition of two matrices. Given two matrices $X \in \mathbb{R}^{n \times m}$ and $Y \in \mathbb{R}^{l \times n}$ CMF will find two correlated decompositions of X and Y: $X = A * B^t$ and $Y = C * A^t$ so that $\alpha * |Y - C * A^t| + (1 - \alpha) * |X - A * B^t|)$ is minimum. This gives the possibility to cluster the users into roles based on both their *TDI* attributes and their assigned permissions by applying the CMF decomposition to the *UPA* matrix and the *TDI* relation presented as an attribute-to-user assignment matrix. Generally, matrix compression algorithms are more appropriate for role mining with minimality objectives. However, they do not always generate intelligible roles.

Formal Concept Analysis (FCA)

Formal concept conforms with the formalization of concept in philosophy. This discipline has a solid theoretical foundation in mathematics [30]. A *formal context* can be defined as a binary relationship between a set of attributes A and a set of objects O. It is represented as a triple $K = (O, A, J)$ where J is an incidence relationship between the objects and the attributes $J \subseteq O * A$. Two main relationships denoted α and ω are associated with each formal context. For a given subset of objects X, $\alpha(X)$ is the maximal set of shared attributes between the members of X: $\forall X \in 2^O$, $\alpha(X) = \{y \in A \mid \forall x \in X, (x, y) \in J\}$. Similarly, the maximum set of objects in common for a given subset of attributes Y is provided by $\omega(Y)$: $\forall Y \in 2^A$, $\omega(Y) = \{x \in O \mid \forall y \in Y, (x, y) \in J\}$. A *formal concept* is the set of maximal sets of objects sharing the same attributes. It is a pair $c = (X, Y)$ so that $\alpha(X) = Y$ and $\omega(Y) = X$. X is then called the *extension* and Y is the *intention* of the concept. If we assimilate the notion of *concept* to the notion of *role* by mapping objects to users and attributes to permissions, and mapping the incidence relationship J to the *UPA* matrix, role mining becomes a direct application of FCA. A wide range of efficient algorithms exist to calculate all the formal concepts corresponding to a given formal context. However, the main difficulty when applying these algorithms to role mining rest with their calculating exhaustively all the possible concepts, a lot of them irrelevant so as to be considered as roles. The set of all the concepts associated with a given formal context ordered by set inclusion forms a complete lattice [9]. This lattice would represent all the possible roles structured into a hierarchical organization. The set inclusion of formal concepts can be interpreted as a hierarchy relationship between the roles. FCA remains the solution for role mining that gives a native hierarchical structure of roles.

When leveraging the FCA in role mining, all the efforts are concentrated on pruning the obtained lattice to eliminate irrelevant roles. We find several publications grappling with this issue. Thion [61] leverages the *sub-hierarchy of Galois (SHG)* to calculate a lattice reduction that deletes the concepts which do not introduce any objects or attributes compared to respectively their ancestors and descendants, and calculates the SHG using a pre-existing tool Pluton [9]. Then, the prioritization of concepts obtained and further pruning require human expertise. Molloy et al. propose the *HierarchicalMiner* [49]. This algorithm prunes the lattice according to a tunable objective function that takes as input, in addition to the *UPA* relation, a vector of weighted criteria covering minimizing the number of roles, minimizing the number of assignments, minimizing the number of hierarchical relationships, and allowing or not exceptional direct user-to-permission assignments. Molloy et al. show in [51] how HierarchicalMiner

is a very competitive and flexible role mining algorithm. Likewise, authors of [13] prove the existence of equivalent sub-lattices as a property of lattices representing concepts of roles. They show that detecting and deleting equivalent sub-lattices may enhance the performance of RM algorithms. Finally, they define *RB-Apriori* (Role Based-Apriori), which is a modified version of an existing FCA algorithm *Apriori*. The algorithm is enhanced to eliminate the equivalent sub-lattices, interpreted as redundancy, and only keeps the relevant set of roles.

Probabilistic Model:

Frank et al. [45] propose to solve the aforementioned *inference problem* using probabilities. Assuming that there is an underlying RBAC state in the *UPA*, and knowing the generation rule from RBAC to *UPA*, they try to infer an RBAC instance from the *UPA* using a likelihood model for $P(UPA|RBAC)$. The system is described by a Bayesian probabilistic model that replaces the binary values of assignment by probabilities between 0 and 1 in the RBAC components *UR* and *RP*. The algorithm tries different values of probabilities of the RBAC components to improve the probability to find *UPA*. In order to handle exceptions and noise, a soft role mining objective function is proposed: the binary matrix *UPA* is decomposed into a structured part (for roles) and an unstructured part (for exceptional and erroneous assignments): $UPA = UR \times RP + DUPA$. The unstructured part *DUPA* is modeled as a Bernoulli process and the likelihood model is applied only to the structured part. In [25], the authors extend the probabilistic model to support *TDI* information. They define a unified objective function as a linear combination of the business information costs and the log-likelihood costs. The business information cost is evaluated using the entropy reduction model presented in the second section. The probabilistic approach for role mining is an extension of the *Multi-Assignment Clustering approach* [59], so a general clustering method for Boolean data where each object can be assigned to multiple clusters. This extension fits more specifically the requirements of role mining and is flexible enough to support both exceptions and *TDI* information. This approach privileges likelihood over minimality and compactness.

Graph Theory

In graph modeling for the role mining problem, the *UPA* is represented by a bipartite graph where users and permissions are the vertices, and the user-to-permission assignments the edges. The targeted RBAC is a tripartite graph with the roles modeled also as vertices, and the edges as the user-to-role and the role-to-permission assign-

ments. Zhang et al. [72] propose a graph optimization technique to find a solution by transforming the bipartite initial graph to a tripartite graph. In the initial phase each set of users holding exactly the same permissions is gathered into a role, and a corresponding vertex created. Then, for each pair of roles, the algorithm computes their intersection and creates a new role for it. Afterwards, it observes if the new created role enhances the optimization metric to decide wether the new role be kept or cancelled. The optimization metric represents the sum of the number of vertices and edges in the tripartite graph, which corresponds to the total number of roles and assignments. The designers of this solution intend to reduce the *administration cost* by minimizing the number of assignments, and to enhance the *enterprise security management* by minimizing the number of roles. The algorithm iterates until the roles are stable. It gives a hierarchical structure of roles, and integrates a set of predefined roles by including them optionally in the initial graph. This method supports also hybrid role mining to define a subset of roles in a top down fashion. The algorithm is initiated by a partially predefined tripartite where the predefined mandatory roles are represented by non-modifiable vertices.

Similarly, Ene et al. [21] map role mining data to a graph and reduce the RMP to the *Minimum biclique cover* which is a well-known NP-Hard problem [8]. In a bipartite graph $(V1, V2, E)$, a biclique is a set of vertices $C1 \subseteq V1$ and $C2 \subseteq V2$ so that $(c1, c2) \in E$ for all $c1 \in C1$ and $c2 \in C2$. In the role mining projection where $V1$ the set of users, $V2$ the set of permissions and E is the set of user-to-permission assignments, a biclique fulfills a potential role. A collection C of bicliques is a *biclique cover* of the edges of G if for every edge (u, v) of G corresponds a biclique $B \in C$ that covers (u, v). A minimum cardinality collection of bicliques that covers the edges of a given bipartite graph is a *minimum biclique cover (MBC)*. Then the optimal solution for the role mining problem *RMP* [21], i.e. the problem of finding the minimum number of roles that maximally grants users access to the resources assigned to them, ends up precisely as the solution to the minimum biclique edge cover problem.

Other Solutions

Most of the formal definitions of role mining problems are NP-Hard. Using the complexity theory, each of these problems could be heuristically solved by reduction to another known NP-Hard problem, for which efficient solutions already exist. For example, in [65], the basic *RMP* is reduced to the *Minimum database Tiling problem*. The *Minimum Tiling Problem* is defined as follows: given a Boolean matrix, find a tiling of the matrix with area equal to the total number of $1s$ in the matrix and consisting of the least possible number of tiles. Using the algebraic representation of role mining,

the concept of *tile* is mapped to the concept of *role*. Thus, an existing greedy solution finding tiles to cover the largest uncovered permissions is adapted to solve *RMP*. Colantonio et al. [15] adopt a different approach. They define two quality measures namely *minability* and *similarity* to decide which *TDI* is more relevant to drive the decomposition process within a divide and conquer strategy for role mining. In fact, for scalability and complexity reduction, *UPA* is divided into independent business fractions running role mining separately on each fraction. Besides, authors in [16] propose a heuristic method to analyze and detect roles using visual analysis of the *UPA*. The proposed algorithm *EXTRACT* uses a sampling heuristic to gather the users and the permissions based on the Jaccard metric. Then, the algorithm *ADVISOR* permutes the users and the permissions in the *UPA* matrix representation according to the output clusters in a way to allow a visual elicitation of the roles.

Each of the mentioned solutions stems from an existing algorithm or mathematical tool to fit a specific optimization criterion in role mining such as minimality of roles, leveraging top down information and privileging the interactivity by providing visual tools. Other fields of inspiration might be explored and exploited for role mining in a tradeoff between the performance of the existing implementations and the adherence to the specificities of the role mining problem.

3.4 Assessment and Enforcement

Post-processing phase embodies the assessment of role mining solution and the enforcement and exploitation of the generated results. The first basic metric is the reconstruction accuracy $||UPA - UR \otimes RP||$ which measures the difference between the initial *UPA* and its reconstruction from the role assignment relations generated by the role mining algorithm. The Frobenius norm ($||X||_f = \sqrt{\sum x_{ij}^2}$) is usually used to measure this error. This metric assumes the initial *UPA* to be error free. Moreover, sensitive to scale it is not comparable. For complex use cases, this metric remains purely indicative. A second method to assess role mining results consists in size measures: assessment of the obtained RBAC state compactness, by measuring the number of roles, the number of assignments, the number of hierarchical relationships, etc. Molloy et al. [51] introduce the *Weighted Structural Complexity (WSC)* of an RBAC state as a tunable size metric that takes into account the number of roles, the number of user to role assignments, the number of role to permission assignments, the size of the hierarchy and the cost of allowing direct user-to-permission assignments with different respective weights. Thus, given a weight vector $W =< wr, wu, wp, wh, wd >$, where $wr, wu, wp, wh, wd \in \mathbb{Q} + \cup\{\infty\}$, the Weighted

Structural Complexity (WSC) of an RBAC state Φ, denoted as $wsc(\Phi, W)$, is computed as follows: $wsc(\Phi, W) = wr*|ROLES|+wu*|UR|+wp*|RP|+wh*|RH|+wd*|DUPA|$. Molloy et al. [51] use the WSC metric to compare the performance of different role mining algorithms and set the weight vector of each algorithm according to its own objective to keep the comparison relatively fair. But the size metric remains insufficient for the assessment of role mining algorithms since compactness neither guarantees intelligibility of the roles nor their maintainability.

A more concluding metric would be the comparison between the true roles and the roles generated by the role mining process. Indeed, to validate that the problem is solved, we should know the hidden underlying RBAC configuration RC^*. In genuine application scenarii of role mining, this information is rarely accessible. However, in experimental scenarios, role mining algorithms are tested and validated with a reverse engineering method where true roles are accessible. Real and artificially generated RBAC states are used. Several synthetic data generation tools allowing to generate random flat roles and hierarchical roles are available [67]. We find for example the *Tree-based data generator* that generates a structure of hierarchical and disjoint roles, and *ERBAC data generator* that calculates a two level layered business role hierarchy [51]. Molloy, Li, et al. [52] propose even to induce real-like noise in the *UPA* after the creation of the roles.

But beyond the availability of the original roles, comparing two sets of roles remains an open issue. Despite multiple propositions in this field, the solutions proposed provide a pairwise comparison role-to-role based on the Jaccard metric or the Hamming metric, but fail to compare the two sets of roles.

In the more recurrent case where the true roles are not known, the generalization test can be used. The generalization test consists of randomly splitting *UPA* into $UPA(1)$ and $UPA(2)$, each representing only a subset of the users and all the permissions, then running the role mining algorithm only on $UPA(1)$. Afterwards, users from $UPA(2)$ are assigned to the generated roles according only to a subset of the permissions. Finally, we check if the remaining permissions in $UPA(2)$ coincide with permissions assigned to the affected roles and we compute the prediction error.

Assessing role mining solutions also covers other considerations such as the running time of the algorithm, and the scalability. Finally, role assessment is submitted to the subjective judgment of the security administrator. She has to evaluate how meaningful and how maintainable and extensible the roles are. Indeed, role mining tools generate a list of roles or a configured RBAC state. Evaluation tests from [51] show that the list of roles is preferable. However, in a context of hundreds of roles, the validation and the interpretation of the roles is a very hard and prone to errors task. Recent

researches are interested in the challenging problem of providing efficient tools to assist the exploitation of the generated roles. For instance, Colantonio et al. [16] provides a tool named *ADVISER* which, given a set of roles, tries to calculate the best permutation of the users and the permissions to give a visual presentation of the roles in a table.

3.5 Synthesis and Discussion

Now let us synthesize the current research trends in role mining. First, the initial idea that drove role mining was to fully automate the role configuration process by leveraging only the existing bottom-up information of *UPA*. It is clear that role mining process could not be fully automatic. Analyzing and leveraging the output of role mining process still requires an intensive human intervention. Likewise, human interaction may be required to tune, supervise and validate the output of each step of the extended role mining process, such as collecting the data, preprocessing it by discarding noise, selecting the pertinent *TDI*, and parameterizing role mining itself, depending on the tool used. Similarly, pure bottom-up approaches do not give interesting results from a practical point of view: roles are not intelligible and difficult to manage. Hybrid role mining techniques are gaining importance and are going to be imposed as the best alternatives for role engineering, offering a compromise of time, cost and meaning of the roles obtained. The main remaining challenge equates with providing tools to assist the administrator during the role mining process. Further work is needed to provide tools for the assessment and the comparison of roles along with the analysis and the enforcement of the generated roles. The most recent research publications focus on how to generate business meaningful roles using hybrid role mining techniques, multi-objective role mining problems to match the diversity and complexity of firm requirements, and assistance tools for generated roles interpretation and enforcement.

There are other remaining issues. For instance, confidentiality of the input data may be enforced by cryptography. Colantonio [11] proposes to adapt a role mining technique in order to support encrypted *UPA* and *TDI* input data, in such way as to to generate roles and then decrypt the obtained roles. More generally, we have shown that the extended role mining process can be used in different contexts of application with different requirements leading to different optimization criteria and different assessment and validation methods. Consistency between the client requirements, the adopted problem definition, the selected algorithm, and the assessment criteria is a must. Yet, it is unfortunately not always guaranteed in the proposed approaches.

Regarding the commercial exploitation of role mining, several specialized editors such as *Bridgestream*, *Eurikify* and *VAAU*, have expounded on the market in the last few years. Their market yet small amounts to about 70 million dollars. Nevertheless, it points to a very spectacular economic growth reaching 70% in 2007 and 2008 for example [19]. Following the expansion of the *Identity and Access Management (IAM)* market, we can witness the integration of the solutions of identification, authentication, role management and authorization functions into a centralized IAM platform, which may simplify the access control management and offer more secure and consistent IT systems. The IAM market has become steady in the last years and is worth around one billion dollars, with a growth of 10% in 2009. Five big editors mainly dominate this market, namely *Computer Associates*, *IBM*, *Oracle* , *NetIQ* and *SUN*. Thus, in the two last years, role mining market is getting closer to the IAM market. The big IAM editors have bought over several specialized role mining editors. For instance, *Bridgestream*, *Eurikify*, and *VAAU* has been purchased respectively by *Oracle*, *Computer Associates* and *SUN* [19].

Besides, in this rising market of role mining, commercialized solutions are mainly based on empirical algorithms. However, as we have demonstrated in this chapter, multiple formal approaches with sound theoretical background and very interesting experimental results have been proposed in literature. We notice then a gap between the solutions proposed in academic research area, and the commercialized solutions. The role mining functions implemented within the IAM platforms could better leverage the theoretical advance in the proposed approaches of role mining in literature. This is likely to enhance the commercial offer and lead to more accurate role mining technologies.

Finally, we notice that the target domain of role mining application is presently limited to user-account in a business context. However, access control frameworks cover other domains such as network security, workflows and operating systems. Applying role mining to these fields is likely to bring out new issues, require new approaches and open new commercial opportunities.

3.6 Conclusion & Key Unhandled Issues

Role mining has been first introduced as a simple process that leverages existing data mining techniques to automate bottom-up configuration of roles. Since then, through research efforts and trials to move role mining from theory to practical applications, a multitude of related problems has arisen. Today, several interesting role mining tech-

niques are proposed in literature with different assumptions and optimization criteria. In this chapter, we have investigated role mining field literature. We have identified pertinent issues and classified them into a complete extended role mining process. We have sequentially detailed the main issues related to role mining and provided a global overview of the main solutions dealing with each of them. Finally we have analyzed and discussed the current trends and the future directions in this research field.

Role mining is likely to promote the development and the adoption of high level of abstraction access control frameworks such as RBAC in enterprises. It simplifies the configuration of RBAC, reduces the costs of role engineering and also increases the confidence of business managers in such procedures by ensuring the conservation of the already deployed authorizations. Nevertheless, we still need to deal with some important issues to ensure such possible exploitation. We have identified some interesting research perspectives.

Defining the users and permissions in some applications is not intuitive. There is a wide panoply of applications of access control. The deployed rules of these applications may present in very different forms, using more or less complex tools and languages. Extracting the deployed concrete entities and rules, formatting this information adequately for role mining, and automating this operation requires defining tailored processing proper to each application. This would be an important step toward some very valuable applications that could open up to commercial exploitation on the role mining market. In particular, we are interested in network security access control systems, treating a large amount of complex firewall rules.

Besides, there are a lot of unexploited input data, and many unexploited functionalities of RBAC in the role mining field. Indeed, leveraging the logs of access from users to their permissions may reveal useful in several manners. First, the chronological aspect permits to detect patterns such as [*user a is permitted to access to b only after accessing to c*]. This aspect could be important in some RBAC configurations, and allow extending role mining usage to workflow applications. Logs may permit to detect other access control properties such as separation of duties and obligations. We can also guess the importance of permissions and assign weights to them accordingly. In addition, thanks to logs, we could solve the remaining issues about noise detection: expired and unused permissions help to detect over-assignment noise, and multiple failed requests could indicate under-assignment noise. To leverage all these features, we need not only new preprocessing techniques but also role mining algorithms with enhanced features. The output would not be solely roles defined as static sets of permissions anymore, but as contextual roles too. Finally, we demand more post processing tools to assist the administrator in leveraging the results of role mining, especially if using

enhanced features of RBAC. Role mining remains interesting mainly for its simplicity and automatism features.

Most of the available solutions have focused on generic algorithms to calculate roles until now. In a next step, we should dwell on more specialized and application oriented approaches, and make means available to leverage the mining results. In this thesis, we provide an extension of role mining solutions to fit network security application. We adapt existing role mining algorithms to extract an instance of the Net-RBAC model instead of the RBAC model from the deployed rules on a firewall. Furthermore, we aim to supply administrators with assistance tools in order to analyze role mining results . We specifically address the problem of analytically comparing a set of roles with reference to another set of roles. We devise an algorithm that maps the inherent relationship between the sets based on Boolean expressions and projects roles from one set into the other set. This approach is useful to measure how comparable the two configurations of roles are, and to interpret each role. We also investigate some related issue of the detection of unhandled perturbations or source misconfiguration. In particular, we propose a solution to detect different cases of role shadowing.

CHAPTER 4

Firewall Policy Mining

The instantiation of a high abstraction model for access control such as Net-RBAC from a firewall configuration is no trivial task. There are role mining techniques which extract a configuration of roles from the already deployed direct user-to-permission assignments using data mining techniques. However, traditional role mining techniques cannot be directly applied to firewall rules. Indeed, the targeted high level model is not RBAC but Net-RBAC.

In this chapter, we propose *policy mining*: a bottom-up approach to extract instances of the Net-RBAC model from a set of firewall rules. Policy mining is an extension of role mining that calculates roles, activities and views to meet network security requirements. We intend to reuse existing role mining techniques to leverage the achievements realized in this field. Due to the fact that the three concrete entities in a rule are structured in consistency, policy mining generates also a set of *abstract rules* that constitute the high level policy. Policy mining handles the problem of factorization of a three-dimensional matrix where traditional role mining is usually assimilated to a problem of a two-dimensional matrix factorization. We still need a more adapted bottom-up approach to the network security application.

Chapter organization. Section 4.1 formalizes the policy mining problem. Section 4.2 proposes an algorithm that solves the policy mining problem. Section 4.3 illustrates the approach proposed by showing an example of the results obtained from applying policy mining on a real firewall set of rules. Section 4.4 concludes the chapter and discusses some perspectives.

4.1 Formalization of the Approach Proposed

Ene et al. [21] have applied a standard role mining solution on firewall rules. To obtain a 2-dimensional input data matrix as required by role mining algorithms, they

have merged two dimensions of the firewall rules which are the source_hosts and the destination_hosts into one dimension considering them to be the permissions in the RBAC model, while they have clustered the services into roles. This initiative did not have much impact since it is neither intuitive nor adapted to network security application. It would correspond to just only one of the steps of the policy mining approach we propose here. We aim at extracting not only roles from the deployed rules, but activities and views as well. Moreover, we aim at mining the abstract rules that define the relations between these abstract entities. Since the expected results are different from the results expected from role mining, the problem should also be formulated differently.

We formalize the problem of *policy mining* as an extension of the *inference role mining problem* [24]:

Definition 6. *Policy Mining Problem*
Let U be a set of users, OPS a set of operations, OBJ a set of objects, and UOO a user-operation-object assignment relation. Infer the unknown Net-RBAC configuration $NS\text{-}RC = (RAV, ROLES, UR, ACTIVITIES, OPA, VIEWS, OBV)$, under the following assumptions:

1. *An underlying Net-RBAC configuration exists*
2. *Exceptions (may) exist.*

By exceptions we mean that direct user-operation-object assignment may still be allowed in the Net-RBAC configuration. This may be necessary in some cases: a compromise with the solution performance, a flexibility required by an organization, or a tip to discard noise and errors in the initial set of rules.

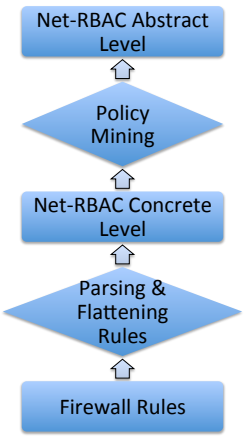


Figure 4.1: Policy Mining Framework to Extract Net-RBAC Security Policy from the Configuration of a Firewall.

4.2 Policy Mining Solution

4.2.1 Assumptions about Input Firewall Rules

Starting from the configured rules in a firewall, algorithm 1 solves the *policy mining problem* by calculating an equivalent Net-RBAC configuration. We assume that the input rules are not order sensitive and are accept only. This is a realistic assumption since tools exist in literature that transform a set of overlapping prioritized firewall rules into a set of flat rules. For instance, in [64] Tongaonkar proposes an algorithm that transforms the set of rules into an equivalent directed acyclic graph (DAG) where the nodes are the packet header fields (dest_host, src_host, dest_port, etc.), the leaf nodes are the firewall actions (allow/deny), and the edges going from a node are labeled with the different values assigned to its packet header field in the set of rules. The algorithm optimizes the DAG via a pruning and node merging process. Then it returns a set of flat and positive firewall rules. The number of flat rules is generally higher than the number of original order sensitive rules, but this does not matter since our policy mining algorithm aggregates the rules afterward. Thus, we consider that flattening the firewall rules and transforming them into a default reject policy form a preprocessing phase for policy mining (see figure 4.1).We also assume in this chapter that we handle rules from only one firewall.

4.2.2 Algebraic Representation of Input and Output Data

To unify the representation of the inputs and outputs of the algorithm and to perform calculations, the involved data are represented with Boolean matrices. Algorithm 1 takes as input a set of firewall rules represented by a 3-dimensional Boolean matrix of user-operation-object assignment *UOO*. In the preprocessing stage, the firewall rules are parsed, and the different instances of source_hosts, services and destination_hosts are saved respectively in the sets of *U*, *OPS*, and *OBJ*. Given m users, n operations and k objects, the *UOO* relation is built as an $m \times n \times k$ 3-dimensional matrix of zeroes. For each original rule where user i is allowed to perform operation j on object l, the cell $\{ijl\}$ in *UOO* is set to 1. The output of algorithm 1 is a Net-RBAC configuration consisting of a set of abstract rules represented by the 3-dimensional Boolean relation *RAV* modeled as an $M \times N \times K$ 3-dimensional Boolean matrix of M roles, N activities and K views, where a 1 in cell $\{ijl\}$ indicates that role i is allowed to perform activity j on view k. The Net-RBAC configuration includes also three relations of assignment of user-to-Role *UR*, operation-to-Activity *OPA*, and object-to-View *OBV*. *UR* is modeled as an $n \times N$ boolean matrix where 1 in cell $\{ijl\}$ indicates the assignment of role j

to user i. Similarly, *OPA* and *OBV* are respectively represented as an $m \times M$ and a $k \times K$ Boolean matrices. Thus, an abstract rule is $RAV(ijl) = 1$, meaning that users assigned to role i are allowed to perform operations assigned to activity j on objects assigned to view l. Projected to the application of firewall rule management, this means that src_hosts assigned to role i are allowed to send services of type activity j to destination_hosts assigned to view k.

4.2.3 Extending Existing Matrix Factorization Techniques

In parallel to role mining modeled as a problem of factorization of a two-dimensional matrix, policy mining is about the factorization of a 3-dimensional matrix, namely the *UOO* matrix. We have surveyed factorization tools for multi-dimensional tensors. Kemp et al. [35] have defined a learning system of concepts with an infinite relational model (IRM) that could be used to cluster multiple dimensions at the same time. However, this approach considers that a concrete entity belongs to only one cluster, which is not a well-grounded assumption when it comes to firewall policies. Moreover, the available implementation of IRM does not allow for concrete entities to be involved in multiple rules either. Obviously, these constraints do not comply well with policy mining requirements.

Besides, we do not aim to propose another role mining algorithm, but rather to leverage existing data mining techniques, including recent achievements in role mining field. We show how to adapt them to the policy mining problem regarding their application in the context of network security. Yet, role mining is mainly about leveraging already existing data mining techniques to extract roles. It has explored and borrowed solutions from different fields as described in details in the previous chapter.

Thus, algorithm 1 takes as input, in addition to the relation *UOO*, a *matrix factorization method*. We note that further arguments may be required by this factorization method. We refer by *factorization method* to any algorithm that takes as input a Boolean matrix C and gives as output two Boolean matrices A and B such as $A \otimes B \approx C$. This notation allows us to reuse most of the role mining techniques proposed in literature. Usually, several possibilities of factorization for the same initial matrix exist. The choice depends on the optimization objectives of the factorization method. We intentionally leave open the possibility of different factorization techniques to benefit from their different optimization objectives and be able to easily adapt algorithm 1 according to the requirements.

To reuse existing factorization methods for policy mining purpose, we have to address two main issues.

- First issue: these methods generally handle 2-dimensional Boolean matrices whereas the input relation of authorizations $UO\mathcal{O}$ is a 3-dimensional Boolean matrix. To solve this problem, we define a polynomial mapping that transforms a 3-dimensional Boolean matrix into a 2-dimensional Boolean matrix, and vice versa, as required by algorithm 1. Figure 4.2 provides an illustration of this transformation. Given an $m \times n \times k$ matrix, we merge the second and third dimensions with regard to the first dimension by presenting in the second dimension all the possible combinations of elements from dimensions 2 and 3. For example if we have a users×operations×objects 3D matrix, we transform it into a user to permission relation, provided that a permission is a combination of an operation and an object. This transformation is obviously reversible with a simple Euclidian division to get back to the 3-dimensional matrix from the 2-dimensional matrix (see figure 4.3). We also note that the size of data is still the same: $m \times n \times k$.

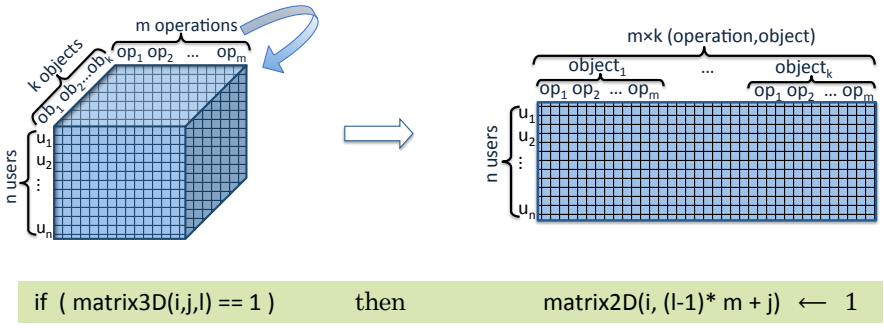


Figure 4.2: Transforming Authorization Boolean Matrix from 3-dimensions to 2-dimensions

- Second issue: the normalization of the inputs and outputs of the different factorization methods that we use, so as to integrate them automatically in the policy mining algorithm 1. We intend to use any data mining or role mining algorithm that could be leveraged to solve $M = A \otimes B$. In practice, they do not always supply directly matrices A and B as output. For example, some role mining algorithms provide only a list of candidate roles from where we have to choose the pertinent roles according to some prioritization criteria. That is to say, we may need specific processing for each factorization method before integrating it in algorithm 1. Moreover, some methods give only an approximation of the factorization of $M \approx A \otimes B$. The approximation error may be assimilated to the exceptions in definition 6. Both cases of approximate and exact factorizations are taken into consideration in our framework.

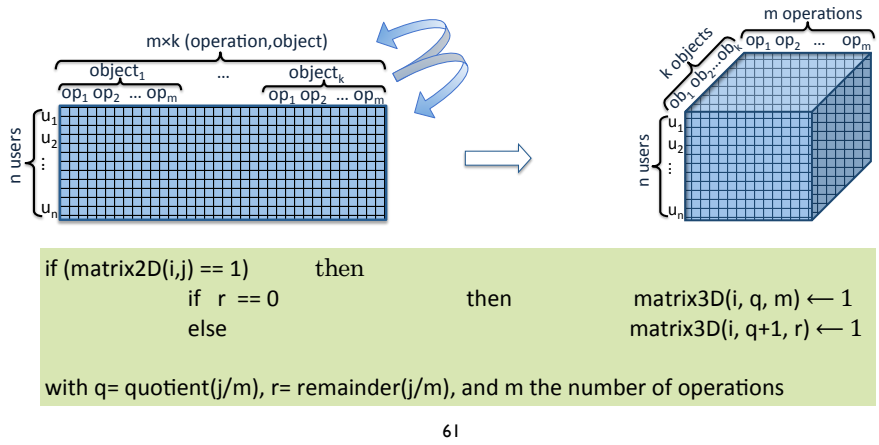


Figure 4.3: Reverse Transforming of Authorization Boolean Matrix from 2-dimensions to 3-dimensions

4.2.4 The Algorithm

Algorithm 1 Firewall Policy Mining

Input: *UOO* : the discretionary firewall rules expressed as a 3-dimensional Boolean relation between users, operations and objects

Input: FactorizationMethod

Input: Further parameters may be required by the factorization method used

Output: A Net-RBAC configuration: user-role *UR*, operation-activity *OPA*, object-view *OBV* and a Role-Activity-View *RAV* Boolean relations

1: 2D transformation: from user-operation-object to user-permission *UPA*
2: mine Roles : FactorizationMethod(*UPA*) to get : $UPA \approx UR \otimes RP$
3: 3D transformation: from role-permission matrix to role-operation-object
4: Permutation : role-operation-object to operation-object-role
5: 2D transformation: operation-object-role to operation-domain *OPD*
6: mine Activities: FactorizationMethod(*OPD*) to get : $OPD \approx OPA \otimes AD$
7: 3D transformation: activity-domain into activity-object-role
8: Permutation : activity-object-role to object-role-activity
9: 2D transformation : object-role-activity to object-capacity *OBC*
10: mind Views : FactorizationMethod(*OBC*) to get : $OBC \approx OBV \otimes VC$
11: 3D transformation: view-capacity into view-activity-role
12: Permutation : view-activity-role into role-activity-view
13: **return** *RAV*, *UR*, *OPA*, and *OBV*

The driving idea of algorithm 1 is to use the *factorization method* to extract three abstract entities: roles, activities, and views sequentially, while taking into account the abstract entities calculated in the previous steps at each new step in order to preserve consistency. By consistency we mean that the three abstract entities should be related by abstract rules so that users assigned to a given role be allowed to perform activities assigned to a given activity using objects belonging to a same View. Initially, the discretionary firewall rules are presented as a 3-dimensional Boolean matrix *UOO*. We transform it into a 2-dimensional Boolean matrix *UPA*, the same as in usual role mining applications. Then we apply the factorization method to extract the roles as sets of permissions in *RP*, and calculate also the *UR* user-to-role assignment relation of the Net-RBAC configuration. Finally, we transform the 2-dimensional matrix *RP* to obtain a new 3-dimensional matrix role-operation-object, by dissociating the operation and object dimensions from the permissions. We accomplish a simple permutation of the dimensions of the matrix to get operation-object-role instead. We transform the latter relation again into a 2-dimensional matrix, and then we apply the factorization over the operation-domain relation where a domain is the interaction between an object and a role. This second factorization extracts the activities while taking into account the previously calculated roles since the users have been substituted by the roles. We keep the *OPA* operation-to-activity assignment relation in the Net-RBAC configuration, and we repeat the same process to extract the Views while substituting the users by the roles and the operations by the activities. We obtain the relation between the roles, activities and views which constitutes the *RAV* abstract rules.

4.2.5 Properties of the Algorithm

Theorem 1. ***Correctness.*** *If the factorization method used provides exact decomposition (without errors), then the authorizations granted to the concrete entities by the Net-RBAC configuration calculated by algorithm 1 are exactly the same as the authorizations granted by the UOO input rules.*

Proof. All the steps of algorithm 1 are reversible. The 2D transformation is reversible by the 3D transformation and vice versa. The permutation of matrix dimensions is reversible by the opposite permutation. And the factorization is reversible by the boolean multiplication. So starting from the Net-RBAC configuration, we can calculate *UOO* by following the opposite way. This means that [*a user u is allowed to do an operation op over an object ob*] in the *UOO* input relation *if and only if* an R role to which u is assigned in *UR* exists and an A activity to which *op* is assigned in *OPA* too,

and a V view to which *ob* is assigned in *OBV* as well, and the relation between R, A and V equalling 1 in *RAV*. □

Computational Complexity: The time complexity of algorithm 1 is at least polynomial on the size of the input matrix *UOO* ($n \times m \times k$) and depends on the complexity of the factorization algorithm used. We run the factorization algorithm three consecutive times, with a decreasing input data size. Several role mining algorithms exist in literature with polynomial complexity on n the number of users and $m \times k$ the number of permissions such as FastMiner [67] with a $O(n^2 \times (m \times k))$ complexity. Changing the dimension of matrices and permuting matrices dimensions operations are polynomial.

4.3 Example

The motivating example in figure 4.4 shows the results of applying the bottom-up approach on an actual set of firewall rules borrowed from [63]. The left column of the figure represents the set of original firewall rules implemented in the Forward table of a single firewall that protects an informatics department. The right column of the figure represents the Net-RBAC instance resulting from policy mining. The Net-RBAC instance consists of a set of high abstraction level rules equivalent to the set of firewall rules. Concrete entities involved in the initial firewall rules have been gathered into structures, thus source-hosts are assigned to *roles*, services to *activities* and destination-hosts to *views*. The set of abstract rules obtained is obviously more compact and offers better readability. The structures of concrete entities obtained can usually be easily associated to significant entities in the network. Without having any further information beside the firewall rules, we have been able to get a significant idea about the access control policy and the network topology as shown in the right column of figure 4.4, which may be very useful for a security administrator to understand, update, and detect anomalies in a firewall configuration.

4.4 Conclusion & Perspectives

Adopting a standard high abstraction model is likely to make firewall management much simpler and safer. We have dedicated this chapter to bridge the gap between firewall rule management and known data mining techniques, including the recent achievements in role mining field. We have presented a framework for a bottom-up approach for Net-RBAC model configuration from the configuration of a firewall. We

Original Set of FW iptables Rules	Policy Mining Results
Concrete Rules: 1- -p tcp -d 192.168.1.250 --dport domain 2- -p tcp -d 192.168.1.251 --dport smtp 3- -p tcp -d 192.168.1.251 --dport smtps 4- -p tcp -d 192.168.1.251 --dport imaps 5- -p tcp -d 192.168.1.251 --dport pop3s 6- -p tcp -d 192.168.1.252 --dport http 7- -p tcp -d 192.168.1.126/25 --dport auth 8- -s 192.168.1.126/25 -p tcp -d 192.168.1.13 --dport ssh 9- -s 192.168.1.126/25 -p tcp -d 192.168.1.14 --dport ssh 10- -s 192.168.1.126/25 -p tcp -d 192.168.1.15 --dport ssh 11- -s 192.168.1.126/25 -p tcp -d 192.168.1.20 --dport ssh 12- -p tcp -d 192.168.1.252 --dport https 13- -s 192.168.1.254/28 -d 192.168.1.11 -p tcp --dport sunrpc 14- -s 192.168.1.236 -p tcp -d 192.168.1.35 --dport ipp 15- -s 192.168.1.254/28 -d 192.168.1.11 -p udp --dport nfs 16- -s 192.168.1.254/28 -d 192.168.1.11 -p udp --dport 4000:4002 17- -p udp -d 192.168.1.251 --dport smtp -j 18- -p udp -d 192.168.1.250 --dport domain -j 19- -s 192.168.1.254/28 -p udp -d 192.168.1.11 --dport sunrpc 20- -s 192.168.1.236 -p udp -d 192.168.1.35 --dport ipp 21- -d 192.168.1.126/25 -p icmp --icmp-type destination-unreachable 22- -d 192.168.1.126/25 -p icmp --icmp-type parameter-problem 23- -d 192.168.1.126/25 -p icmp --icmp-type source-quench 24- IPTABLES -A FORWARD -j REJECT	*Abstract Rules:* *1- FROM Corporate_Zone TO SSH_Server FOR Secured_Traffic* *2- FROM Any TO Web_Server FOR Secured_Web_Service* *3- FROM Any TO Corporate_Zone FOR Protocol_Control* *4- FROM Exterior TO Printer FOR Printing* *5- FROM DMZ TO SUN_NFS_Server FOR UNIX_Services* *6- FROM Any TO DNS_Server FOR DNS* *7- FROM Any TO Messaging_Server FOR E-mailing* *Source_Host to Role assignment:* *1- Corporate_Zone= {192.168.1.126/25}* *2- Exterior = {192.168.1.236}* *3 -DMZ = {192.168.1.254/28}* *4- Any = {any}* *Service to Activity assignment:* *1- Secured_Traffic = {-p tcp -dport ssh}* *2- Secured_Web_Service = {{-p tcp -dport auth}, {-p tcp -dport https}, {-p tcp -dport http}}* *3 -Ptrotocol_Control = {{tcp FOR auth}, {-p icmp -icmp-type: destination unreachable},{ -p icmp -icmp-type parameter-problem}, { -p icmp -icmp-type source-quench}}* *4- Printing = {{-p tcp -dport ipp}, {-p udp -dport ipp}}* *5- UNIX_Services = {-p tcp -dport sunrpc}, { -p udp -dport sunrpc}, {-p udp -dport nfs}, {-p udp -dport 4000:4002}}* *6- DNS = {{-p tcp -dport domain},{-p udp -dport domain}}* *7- E-mailing = {{-p tcp -dport smtp}, {-p tcp -dport smtps}, {-p tcp -dport imaps}, {-p tcp -dport pop3s}, {-p udp -dport smtp}}* *Destination_Host to View assignment:* *1- SSH_Server = {[192.168.1.13 - 15] and 192.168.1.20}* *2- Web_Server = {192.168.1.252}* *3 - Corporate_Zone = {192.168.1.126/25}* *4- Printer = {192.168.1.35}* *5- Sun_NFS_Server = {192.168.1.11}* *6- DNS_Server = {192.168.1.250}* *7- Messaging_Srever = {192.168.1.251}*

Figure 4.4: Example of Firewall Policy Mining Results

have proposed a formalization of policy mining problem as well as an algorithm to solve it. This solution allows us to extend and leverage existing mining tools to the network security application. We have exemplified our approach by a sample of the results obtained from a real data set.

This work has several interesting perspectives. Currently, it is applicable for stateless firewall rules. We consider using the concept of *context* defined in the OrBAC model to manage stateful firewalls. Moreover, in the preprocessing phase, we transfer the set of rules to a set of accept only rules in a default close policy. From access control perspective, this does not affect the deployed policy. However, negative rules are implemented on firewalls for several practical reasons such as defense in depth, redundancy, and countering spoofing attacks. As a perspective, the preprocessing phase should support also the transformation of the rules into a reject only policy in a default open policy in order to perform policy mining on the negative rules also and help the administrator to analyze its deployed reject rules as well. Beyond the network security application, other access control applications may require the ternary rules modeling as proposed by the OrBAC model. This policy mining technique would offer a bottom-up approach for all these applications. More important, network access control usually requires multiple firewalls that work in unison to enforce the same access control policy. This policy mining technique should be extended to the mining of a system of firewalls policy for a real impact in the network security management application. We address this problem in the next chapter.

CHAPTER 5

Mining a High Level Access Control Policy in a Network with Multiple Firewalls

Large and medium networks are usually protected by more than one firewall. The policy mining technique presented in the previous chapter provides a bottom-up approach that automatically extracts instances of the Net-RBAC policy from the deployed rules on a firewall. It is likely to highly promote the usage of this model. Nevertheless, it has been designed for a single firewall. In order to provide a complete automatic bottom-up framework for network policy mining, we still need a further processing of policy mining performed on each firewall into a global network policy.

In this chapter, we develop the problem of integration of Net-RBAC policies resulting from policy mining over several firewalls to mine a high level network policy. We propose a two-staged process. In the first stage, we unify the hierarchies of the abstract entities of all the firewalls. We assimilate the problem to the general mathematic problem of partially ordered set merging. In the second stage, we build the effective deployed network policy rules. We integrate the highly abstracted rules from the firewalls while checking several security properties. We detect irrelevance anomalies consisting of rules that never apply because they are enforced in a firewall that is not on the path between the source and the destination. We also detect inaccessibility anomalies due to inconsistency between the configuration of firewalls on the path of the same flow. The correctly deployed rules are aggregated into the network policy, whereas the detected anomalies are reported.

Chapter organization. Section 5.1 introduces the problem of Net-RBAC policies integration, and explains the followed methodology to solve it. Section 5.2 presents an algorithm that tackles abstract entities integration. Section 5.3 presents a methodology for network access control rules mining through abstract rules integration while verifying accessibility and relevance properties in the deployed policy. Section 5.4 illustrates the network policy mining approach by a realistic example. Section 5.5 concludes the chapter.

5.1 A Bottom-Up Framework to Mine A Model Based Network Security Policy

Most of the organizations deploy their access control network policy over multiple firewalls to enhance the overall performance, and also to define security zones in the network topology by distributed check points. The policy mining technique presented in the preceding chapter has been designed for the management of a single firewall. Obviously, gathering the rules from all the firewalls involved in a given policy into one set, and then performing policy mining as in the preceding chapter is unfeasible. It may distort the intended access control policy for many reasons. The rules are order sensitive, so merging rules from different firewalls raises the problem of ordering them. Moreover, the filtering effect of a rule depends on the location of the firewall of enforcement in the topology.

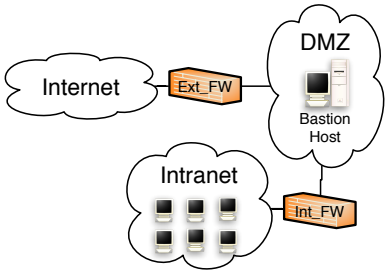


Figure 5.1: Network Architecture Using Two Firewalls and a Screened Subnet.

For instance, we analyze the architecture in figure 5.1. Let's suppose that the policy intends to allow http traffic coming from Internet to reach the secured Bastion_Host, and prevent it from reaching directly the internal network. Yet, http traffic that is relayed by the Bastion_Host is allowed to reach the internal hosts. One possibility to implement this policy is to enforce the exterior firewall Ext_FW the following rule:

r1: allow -src any -service http -dest any

and to enforce in the interior firewall Int_FW the following couple of rules:

r2: allow -src Bastion_Host -service http -dest any
r3: deny -src any -service http -dest any

When we gather the rules from the two firewalls, if we put the rules from Ext_FW before the rules from Int_FW, rule r1 will shadow both rules r2 and r3 which become useless. Thus in the policy obtained, any http traffic from the Internet can reach the internal network threatening its security. If we put the rules from Int_FW before, then r3 will shadow r1 and no traffic will reach the perimeter network at all. If the deny rule r3 is not explicit, then the internal network will be exposed to external http traffic. In all these cases, the resulting policy from the merger is different from the enforced one.

The use of *any* in a firewall instead of a specific list of destinations is very usual in practice. Administrators may make such a choice for simplicity when they know that, in a network topology, the following firewall will block the traffic undesirable to reach the next zone. From policy mining perspective, where we aim to a generic bottom-up approach to be run over the already existing configurations of the firewalls, we do not make either assumptions about the choices of the administrators or about any possible unintended error or misconfiguration problem in the firewalls. There is no way to handle exhaustively all the possible cases of interactivity problems between the firewalls with reasonable cost.

For complete network security policy mining, we run the policy mining process as described in the previous chapter on the configuration of each firewall apart first. This modular approach allows individual analysis of each firewall configuration by structuring its rules into a Net-RBAC state. This could be performed on a regular basis to check the current configuration, without necessarily running the bottom-up process for the whole network. In a second step, we extend the process of policy mining for one firewall to handle multiple firewalls as the framework in figure 5.2 shows. We analyze and integrate the resulting Net-RBAC policies from the firewalls into one global policy for the network. In this objective, and with regard to the structure of the NS_RBAC model, we define a two staged process. First we compare, merge and build the hierarchy of each of the abstract entities, namely the roles, activities and views. Second, we merge the abstract rules into one network policy.

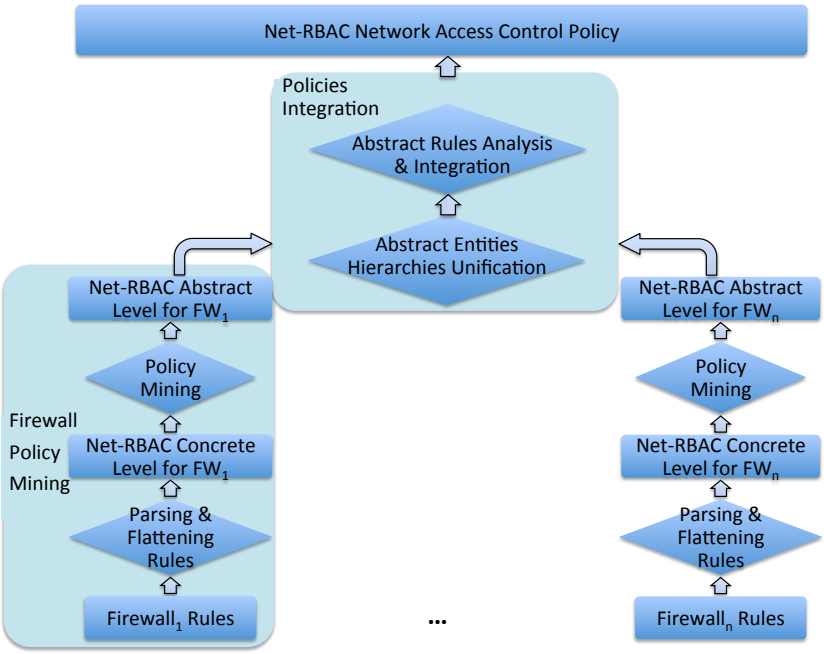


Figure 5.2: Policy Mining Framework to Extract Net-RBAC Network Security Policy from the Configurations of Multiple Firewalls.

5.2 Integrating Abstract Entities

Shafiq et al. [57] have dealt with the issue of integration of the roles of multiple *RBAC* policies. However, the algorithm presented is very sophisticated and inherently includes additional features of *RBAC* such as separation of duties, that do not tally with network security application as modeled in the policy mining process. In this chapter, we present a different approach that may be used to integrate RBAC roles as well as Net-RBAC roles, activities and views.

In Net-RBAC, abstract entities are defined both by the permissions and by the concrete entities assigned to them. From the latter perspective, each role (resp. view) is a set of source_hosts (resp. destination_hosts) presented as: IP_address, range of IP addresses, or a subnet identifier with IP/netmask. Similarly, activities are sets of services. Each service is characterized by some or all of these fields: [protocol, protocol related options, (such as the type for ICMP and the acknowledgment value for TCP) destination port(s), source port(s)]. Besides, a set of abstract entities may be hierarchical, depending on the initial set of rules and on the factorization objectives in the mining process. Net-RBAC supports abstract entities hierarchy, generalizing role hierarchies defined in RBAC.

Thus, a set of abstract entities may be considered as a *partially ordered set (poset)*, with regard to the inclusion relation. In network application, a role $r1$ is included in a role $r2$ if and only if all the source_hosts assigned to $r1$ are also assigned to $r2$. In this case, $r1$ is a sub-role of $r2$, and $r2$ is a senior-role to $r1$. An activity $a1$ is a sub-activity of $a2$ if and only if each service si from $a1$ is included in a service sj belonging to $a2$. A service $s1$ is included in a service $s2$ if and only if $s1$ is characterized by the same fields and each of the fields of $s1$ is included or equal in its corresponding field of $s2$. The characteristics of a service are viewed as an indivisible element.

To obtain a homogeneous policy for the network, we have to build unified hierarchies of roles, activities and of views from the abstract entities mined in each firewall. We compare the elements assigned to each abstract entity to detect hierarchical relationships and also exact matching entities so we avoid redundancies in the global policy. It is a posets merging problem. Chen et al. [10] have considered poset merging as a generalization of the list merging problem. They have proposed a low complexity solution to merge two chains from a poset. However they consider that the decomposition of the two posets to merge into chains is out of scope. We propose a new algorithm that does not require this decomposition.

5.2.1 Abstract Entities Merger Algorithm

We handle only the roles in this section, the process being the same for the activities and the views. Algorithm 2 integrates the roles of one firewall into the network policy roles. The inputs are the roles of a firewall on the one hand, and the current global roles on the other hand. For consistency with the outputs of policy mining, we adopt the Boolean matrices representation of data as follows. Given m users, and n roles (i.e.,$|U| = l$, $|ROLES| = n$), the user-to-role UR mapping is represented as an $l \times n$ Boolean matrix where a 1 in cell $\{ij\}$ indicates the assignment of role j to user i. The role-to-role RH mapping is represented as an $n \times n$ Boolean matrix where a 1 in cell $\{ij\}$ indicates the role i is a sub-role of the role j. If the set of roles is flat, RH will be a diagonal matrix. Thus, the inputs to algorithm 2 are the matrices FUR, FRH, NUR, NRH that represent the relations of respectively: firewall-user-to-role assignment, firewall roles' hierarchy, current network-user-to-role assignment, and current network roles' hierarchy. The set of roles in FUR and NUR must be defined over the same set of users U. We assume that we have complete and correct information about the ordering within both the initial sets of roles.

Algorithm 2 merges the two sets of roles and identifies the redundancies of roles between the firewalls along with the inheritance relationships. The intermediary matrix

K is built to bookkeep the relations between the roles from the two sets while the algorithm is running (line 1). The rows of K represent the roles of the firewall and the columns represent the roles of the network policy previously integrated from other firewalls. K_{ij} value will designate the relation between the role fR_i[1] from the set FUR and the role nR_j from the set NUR. For clarity, the relations between the roles are represented in K with symbols: nh for not handled yet, d for unrelated with inheritance relation, l in K_{ij} means fR_i is a sub-role of nR_j, and h means fR_i is a senior-role of nR_j. Finally, r stands for redundancy. When we compare two roles, we transfer their resulting order on their senior-roles and sub-roles to avoid unnecessary comparisons (lines 7, 8, 14, 18, 22). To find the indices in matrix K of the roles in the crossing of the two sets such as *the senior-roles of a given role fR_i in FUR and sub-roles of a given nR_j role in NRH* (e.g. line 18), we leverage some of the properties of initial matrices. Indeed, in FRH, the column i provides the indices of the sub-roles of fR_i and the row i provides the senior-roles of fR_i. The same holds for nR_j in NUR. Thus, a simple Boolean multiplication between row i of FRH and column j of NRH provides the indices in K of the cells that correspond to relations between the sub-roles of fR_i in the firewall and the senior-roles of nR_j in the current network policy which are required in lines 8 and 18 of the algorithm. Likewise, a direct multiplication of column j of NRH with row i of FRH provides the indices in K of the senior-roles of fR_i and the sub-roles of nR_j (lines 7 and 14), and a multiplication of column i of FRH with transpose of column j in NRH provides indices of relations between sub-roles of nR_i and sub-roles of nR_j (line 22).

The output of algorithm 2 is the updated set of network roles NUR' after the integration of the roles UR, and their NRH' updated hierarchy relation. The set of users is still the same, and the set of roles is augmented by the number of firewall roles not redundant with roles already in the network policy ($max_{n,m} \leq m' \leq m + n$).

5.2.2 Algorithm Properties

Theorem 2. ***Correctness.*** *Algorithm 2 calculates the relationships between roles from both input sets correctly.*

Proof. A role fR_i is calculated to be a sub-role of nR_j only if the users of fR_i are included in the set of users of nR_j by direct comparison (line 13) or nR_j is a super-role of a senior-role of fR_i (lines 7,14). A role fR_i is calculated to be a senior-role of nR_j only if the users of nR_j are included in the set of users of fR_i by direct comparison

[1] In algorithm 2 we abbreviate assigned_users(R) by R

Algorithm 2 Abstract Entities Merger

Input: NUR and NRH: an $l \times m$ and an $m \times m$ Boolean matrices of the user's assignment and hierarchy of current roles in the network policy

Input: FUR and FRH: an $l \times n$ and an $n \times n$ Boolean matrices of the user's assignment and hierarchy of the roles in the firewall to be integrated in the network policy

Output: NUR' and NRH': an $l \times m'$ and an $m' \times m'$ Boolean matrices of the user's assignment and hierarchy of the roles in the updated network policy

1: create $n \times m$ matrix K with all the cells initialized to nh
2: **for** each role fR_i in FRH (i from 1 to n) **do**
3: **for** each role nR_j in NRH (j from 1 to m) **do**
4: compare fR_i to nR_j
5: **if** $fR_i == nR_j$ **then**
6: The row i in K is set to r
7: **for** each fR_v sub-role of fR_i in FRH; **for** each nR_w senior-role of nR_j in NRH; $K_{vw} \leftarrow l$
8: **for** each fR_v senior-role of fR_i in FRH; **for** each nR_w sub-role of nR_j in NRH; $K_{vw} \leftarrow h$
9: **else**
10: **if** $K_{i,j} == nh$ **then**
11: **switch** comparison of fR_i to nR_j **do**
12: **case** 1: $fR_i \subsetneq nR_j$
13: $K_{ij} \leftarrow l$
14: **for** each fR_v sub-role of fR_i in FRH; **for** each nR_w senior-role of nR_j in NRH; $K_{vw} \leftarrow l$
15: **end case**
16: **case** 2: $fR_i \supsetneq nR_j$
17: $K_{ij} \leftarrow h$
18: **for** each fR_v senior-role of fR_i in FRH; **for** each nR_w sub-role of nR_j in NRH; $K_{vw} \leftarrow h$
19: **end case**
20: **case** 3: $fR_i \cap nR_j == \emptyset$
21: $K_{ij} \leftarrow d$
22: **for** each fR_v sub-role of fR_i in FRH; **for** each nR_w sub-role of nR_j in NRH; $K_{vw} \leftarrow d$
23: **end case**
24: **case**
25: $K_{ij} \leftarrow d$
26: **end default**
27: **end switch**
28: **end if**
29: **end if**
30: **end for**
31: **end for**

32: $m' \leftarrow m$
33: **for** each role fR_v corresponding to $row_v \neq r$ in K **do**
34: $m' \leftarrow m' + 1$
35: add a column m' in NUR' zeroed with assigned_users(fR_v) set to 1
36: add a row m' and a column m' to NRH' zeroed
37: **for** w :=1 to m **do**
38: **if** $K_{vw} == l$ **then** $NRH'_{m'w} \leftarrow 1$
39: **else if** $K_{vw} == h$ **then** $NRH'_{wm'} \leftarrow 1$ **end if**
40: **end for**
41: **end for**
42: **return** NUR' and NRH'

(line 17) or nR_j is a sub-role of a sub-role of fR_i (lines 8,18). Redundancy is detected only by direct comparison (line 6). Finally, only the partially overlapping roles (line 25), the disjoint roles (line 21), and the respective sub-roles of two disjoint roles (line 22) are not related by the inheritance relation according to the algorithm. □

Theorem 3. ***Completeness.*** *Algorithm 2 discovers all the redundancies and the inheritance relationships between roles from the two input sets.*

Proof. The two embedded *for* loops (lines 2 and 3) ensure that all the cells of K that are *not handled* will be processed. We need to handle the relation between two roles only once since two roles can be related by one and only one from the following relationships: $ri \subsetneq rj$ or $ri \supsetneq rj$, or $ri = rj$, or no inheritance relationship in case the roles are disjoint or partially overlapping. All the cases of redundancy are detected first by direct comparison (line 6). The *switch case* structure (line 6 to 21) considers all of the remaining possibilities. □

Computational Complexity: Algorithm 2 complexity is calculated by the number of comparisons of roles necessary to establish the inheritance relation. It depends on the inputs. The maximum number of comparisons is $n \times m$ if both the two sets are flat. The algorithm is more efficient when the inheritance tree is dense in the two sets. The best complexity is deterministic when each input set is a chain, i.e. a totally ordered set.

5.3 Integrating Abstract Rules

The target network policy from mining is a unified set of rules independent from the topology except for routing decisions that take part in the high level security purposes. The rules are checked to be relevantly and consistently deployed in the network. In this phase, we integrate the abstract rules resulting from policy mining on each firewall into one global set of rules.

5.3.1 Abstract Rules Merger Algorithm

The inputs of algorithm 3 represent the abstract rules of each firewall extracted by policy mining. We have substituted in the rules the redundant entities between the different firewalls by the unified entity names identified in the previous stage of abstract entities integration. Besides, the algorithm requires the routing paths and the position of the firewalls in the topology. For each rule in each firewall, algorithm 3 checks the relevance and accessibility properties before adding the rule to the network policy. A rule is relevant if the firewall where it is implemented is on the path between the role and the view zones. Otherwise, the rule is reported as irrelevant since it never applies, and is discarded from the final mined network policy (line 5). If the rule is relevant, algorithm 3 verifies the accessibility of the rule on the whole path. If one or several other firewalls on the path between the role and the view zones block the traffic of the rule, then the rule is not properly enforced in the network. It is also discarded from the mined network policy, and reported as an inaccessibility anomaly. There are two cases: partial and total inaccessibility. If the traffic is wholly blocked at a firewall this means total inaccessibility. If a subset of the traffic is accessible along the path, we derive that sub-rule and we add it to the final policy, and we report the main rule as inaccessible.

Verifying the accessibility of a rule from a given firewall in the other firewalls on the path is not a straightforward operation. The handled rule may be supported in the other firewalls under different forms: an exact matching rule, or divided into several rules that may be border overlapping. This springs from the multiple ways to implement the same concrete rules on the firewalls and the many ways to cluster these rules by policy mining. So, we have to compare it to all the other rules in each firewall on the path until it is covered, or otherwise it is inaccessible. Al-Shaer et al [6] have proposed a modeling for the interaction of inter-firewall rules based on rule comparison in order to automatically reveal conflicts at rule insertion, removal, and modification. They have modeled the rules as k-tuples of concrete entities, and defined five categories

of relationships between rules. Nevertheless, we have noted that some rules may match both categories *partially disjoint* and *correlated* definitions, whereas these categories are assumed to be disjoint in their anomaly detection automate design. In this chapter, our work is grounded on a new definition of the interaction categories for the comparison of rules that we use for accessibility verification. Any pair of compared rules matches one and only one of these relationships considered by algorithm 3 :

1. Independent: two rules are totally disjoint if and only if one or more of the corresponding fields are totally disjoint (e.g. $role1 \cap role2 = \emptyset$) because they can never match the same packet.

   In all the remaining categories, there is a dependency between the two rules, i.e. packets exist that may be matched by both of them. This means that each couple of respective abstract entities in the three fields of the rules (role1-to-role2, activity1-to-activity2, view1-to-view2) is overlapping.

2. Exact matching: all of the three fields are exactly matching. The two rules make up a redundancy between the firewalls.

3. Totally overlapping: each of the three entities in one rule are included in the corresponding entity in the other rule. Depending on the order of the two compared rules, this relationship can be inclusion or generalization.

4. Partially overlapping: at least one pair of entities is border-overlapping and the remaining pairs are totally or partially overlapping.

The algorithm compares each pairwise of abstract entities (role1-to-role2, activity1-to-activity2, view1-to-view2) as sets of concrete entities, and counts the number of occurrences of each symbol in the table of reference below:

Table 5.1: Characterization of the Relations Between Two Rules.

$E1 \subsetneq E2$	$E1 \supsetneq E2$	$E1 = E2$	$E1 \cap E2 = \emptyset$	$E1 \cap E2 \neq \{E1, E2, \emptyset\}$	
any	any	any	≥ 1	any	Disjoint
0	0	3	0	0	Exact Match
≥ 1	0	≥ 0	0	0	Inclusion
0	≥ 1	≥ 0	0	0	Generalization
any	any	any	0	≥ 1	Partial Overlap

Algorithm 3 uses *cache_r_j_FW_k* to cumulate the coverage of the r_j rule by rules from the FW_k firewall that is also crossed by the traffic of r_j. Moreover, to avoid comparing two rules more than once during all the execution, algorithm 3 also uses *cache_r_l_FW_i* to bookkeep the compared rules coverage with the current r_j rule. All the caches of the algorithm are initialized to an empty rule with three empty fields. To allow a good readability, algorithm 3 considers that there is one path between the role and view zones of a given rule (line 3). In practice, if the function *get_path* returns several paths, all the paths are addressed subsequently. The rule is reported to be irrelevant only if the firewall where it is enforced does not belong to any of the paths. The accessibility of the rule is checked for each path apart. Moreover, we add to the structure of the rule a *context_path* that specifies the list of the crossed firewalls before we add it to the network policy. Likewise, algorithm 3 does not show how we handle the differentiation between partial and total inaccessibility although it is supported in our implementation (sections 5.4, 8.5). In that respect, we add a further control after having compared the rule to all the other rules in a given firewall (line 35) so that if *cache_r_j_FW_k* does not cover r_j, but is not empty, r_j may be partially accessible. We create a new rule corresponding to *cache_r_j_FW_k* and we insert it in the firewall rules in order to check its accessibility later within the second main *for* loop (line 11). The outputs of algorithm 3 are the network policy rules with the pattern (role, activity, view, and optionally context_path), checked to be relevant and accessible, and the report of irrelevant and inaccessible rules.

5.3.2 Algorithm Properties

Theorem 4. ***Correctness.*** *All the rules in the final network policy are accessible and relevant.*

Proof. In algorithm 3, a rule is added to the network policy only in line 43. This clause is within the *else* flow of control of the *if* line 4, which guarantees that the relevance of the rule has been checked. Moreover, adding the rule demands one condition: that the Boolean variable *Path_Accessibility* be true. This variable is initialized to true after checking that the rule is relevant (line 7), but is set to false if the rule is not accessible on any firewall on the path (line 39) within the *for* loop that checks accessibility on the path of the rule(line 8), and cannot be set to true again for the same rule, which guarantees the accessibility of the added rule. □

Theorem 5. ***Completeness*** *All the firewalls abstract rules that are relevant and accessible are added to the final policy.*

Algorithm 3 Abstract Rules Merger

Input: The set of abstract rules (R,A,V) of each firewall in the network

Input: The routing paths

Output: The set of abstract rules of the access control policy of the network (R,A,V, context_path)

Output: The report of verification of relevance and accessibility of the firewall rules

1: **for** each firewall FW_i in the network **do**
2: **for** each rule r_j in FW_i **do**
3: Path ← get_path(Role(r_j), View(r_j))
4: **if** $FW_i \notin Path$ **then**
5: Report (Irrelevance, FW_i, r_j)
6: **else**
7: Path_Accessibility ← True
8: **for** each firewall FW_k in Path (other than FW_i) **do**
9: FW_Accessibility ← False
10: **if** FW_k is not already handled (in line 1) **then**
11: **for** each rule r_l in FW_k **do**
12: **switch** comparison of r_j and r_l **do**
13: **case** 1: disjoint
14: do nothing
15: **end case**
16: **case** 2: exact match
17: FW_Accessibility ← True
18: do not consider r_l from FW_k again
19: **end case**
20: **case** 3: r_l is a generalization of r_j
21: FW_Accessibility ← True
22: cache_r_l_FW_i ← cache_r_l_FW_i $\cup$ $r_j \cap r_l$
23: **end case**
24: **case** 4: r_j is a generalization of r_l
25: cache_r_j_FW_k← cache_r_j_FW_k $\cup$ $r_j \cap r_l$
26: do not consider r_l from FW_k again
27: **end case**
28: **case**
29: cache_r_j_FW_k← cache_r_j_FW_k $\cup$ $r_j \cap r_l$
30: cache_r_l_FW_i ← cache_r_l_FW_i $\cup$ $r_j \cap r_l$
31: **end default**
32: **end switch**
33: **end for**
34: **end if**

```
35:            if cache_r_j_FW_k == r_j then
36:              FW_Accessibility ← True
37:            end if
38:            if Not FW_Accessibility then
39:              Path_Accessibility ← False
40:            end if
41:          end for
42:          if Path_Accessibility then
43:            Add {r_j,Path_context} to Network Policy Rules
44:          else
45:            Report (Inaccessibility, r_j)
46:          end if
47:        end if
48:      end for
49: end for
```

Proof. The main *for* loop (line 1) ensures that all the firewalls are processed. For each firewall, algorithm 3 processes the rules one by one in an embedded *for* loop (line 11) to check their relevance and then their accessibility. The accessibility of a rule in another firewall on the path is confirmed if the traffic of the rule is permitted by one or multiple rules. The comparison between two rule belongs at least to one of them which can be proved by mandatory counting. The categories are exclusive: a rule fits at most one of them. There is no need to compare with the rules of firewalls that have been already handled in the main *for* loop, since all the rules of these firewalls have been previously compared with all the rules of firewalls on the same paths, and the comparison results are in the corresponding caches. To ensure that the caches of the other firewalls are complete, the algorithm finishes the comparison with all the rules even if the boolean variable *FW_Accessibility* is set to true. After processing a given rule, algorithm 3 removes its sub-rules from firewalls which are located in the same path (lines 18, 26). Indeed if the rule is totally inaccessible, all the sub-rules are also totally inaccessible, and if it is added to the network policy, they are taken into consideration as well in an aggregated form. □

Computational Complexity: Time complexity of algorithm 3 is determined by the number of rules comparison. Only rules from firewalls involved in the same path at least for one rule are compared. The worst case complexity is O($c \times n^c$), when all the firewalls are on one path, with c the number of firewalls, and n the average number of rules on each firewall. Considering that c is a bounded number and that n approaches

infinity, this complexity is polynomial. The best time complexity is deterministic when every rule crosses only one firewall.

5.4 Example

We illustrate our approach by an example from the results obtained from network policy mining. We consider an access control policy for the architecture given in figure 5.5. The traffic of the customers and of the employees are separated through two independent screened networks using four firewalls [20]. Employees are allowed to surf on the Internet, and to remotely connect to their work stations only throughout perimeter_network_2. Some of the traffic must be relayed by the proxy Bastion_Host. The servers of the organization are provided with public interfaces through the perimeter_network_1. Administration and saving backups of the servers are monitored from the private internal network.

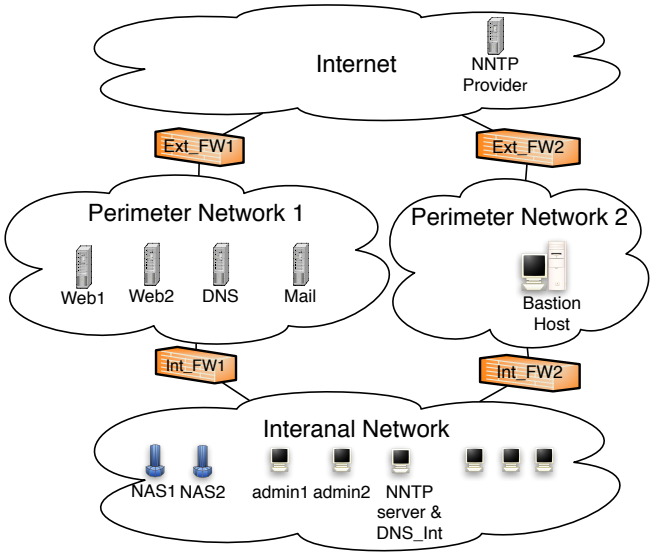


Figure 5.3: Network Topology with Two Separated Screened Networks.

Figure 5.4 shows the abstract rules resulting from policy mining on each of the four firewalls. These are the inputs of the policy integration process. We do not present the original concrete rules nor the concrete entities to abstract entities assignment of each firewall apart. We are solely interested in the subsequent phase of firewall policies integration for network policy mining. Naming the roles, activities and views derives from the appreciation of the security administrator and is not automatically generated by policy mining. Figure 5.6 shows the results of merging the abstract entities with algorithm 2. On the left side, we can see the unified sets of roles and activities, and

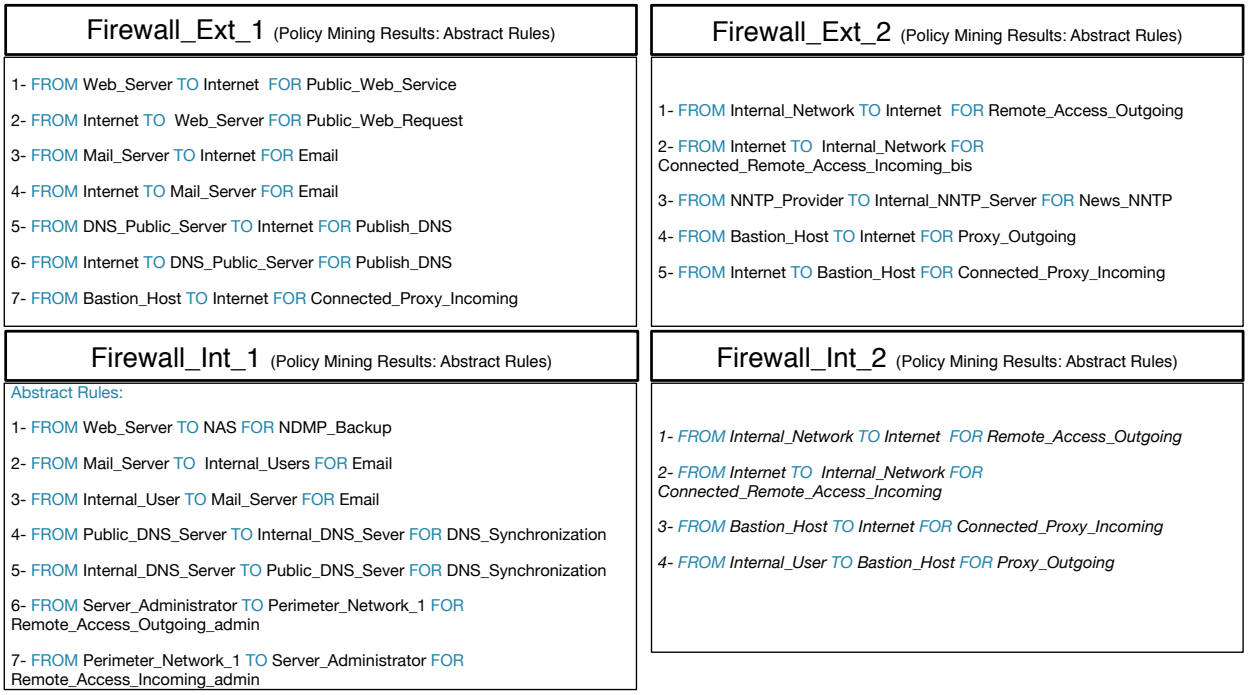


Figure 5.4: Abstract Rules Resulting from Policy Mining On Each Firewall.

the concrete entities assigned to them. In this example, the roles happen to be exactly the same as the activities, so we represent only one set. The right side shows the unified hierarchy relationships of roles/views, and activities. We can note that *any* is an exception considered as an indivisible element.The results of rule merging with algorithm 3 are in figure 5.5. In the report, we can see that *rule 7* from *FW_Ext1* is detected to be irrelevant since the firewall is not on the path from the Bastion_host to the Internet. Likewise, *rule 3* in *FW_Ext2* is totally inaccessible because *FW_Int2* is also crossed by the traffic, and blocks all of it. *Rule 2* in *FW_Int2* is partially accessible because *FW_Ext2*, which is on the same path, permits all the traffic between the source and the destination but for the Command Channel FTP passive mode. The rule is reported to be partially inaccessible, and a new rule (*rule 2* in figure 5.5) with the maximal traffic allowed by both firewalls is created and added to the network policy rules.

5.5 Conclusion & Perspectives

In this chapter, we have proposed an integration process of Net-RBAC policies of multiple firewalls involved in the same network security defense. This process aims at

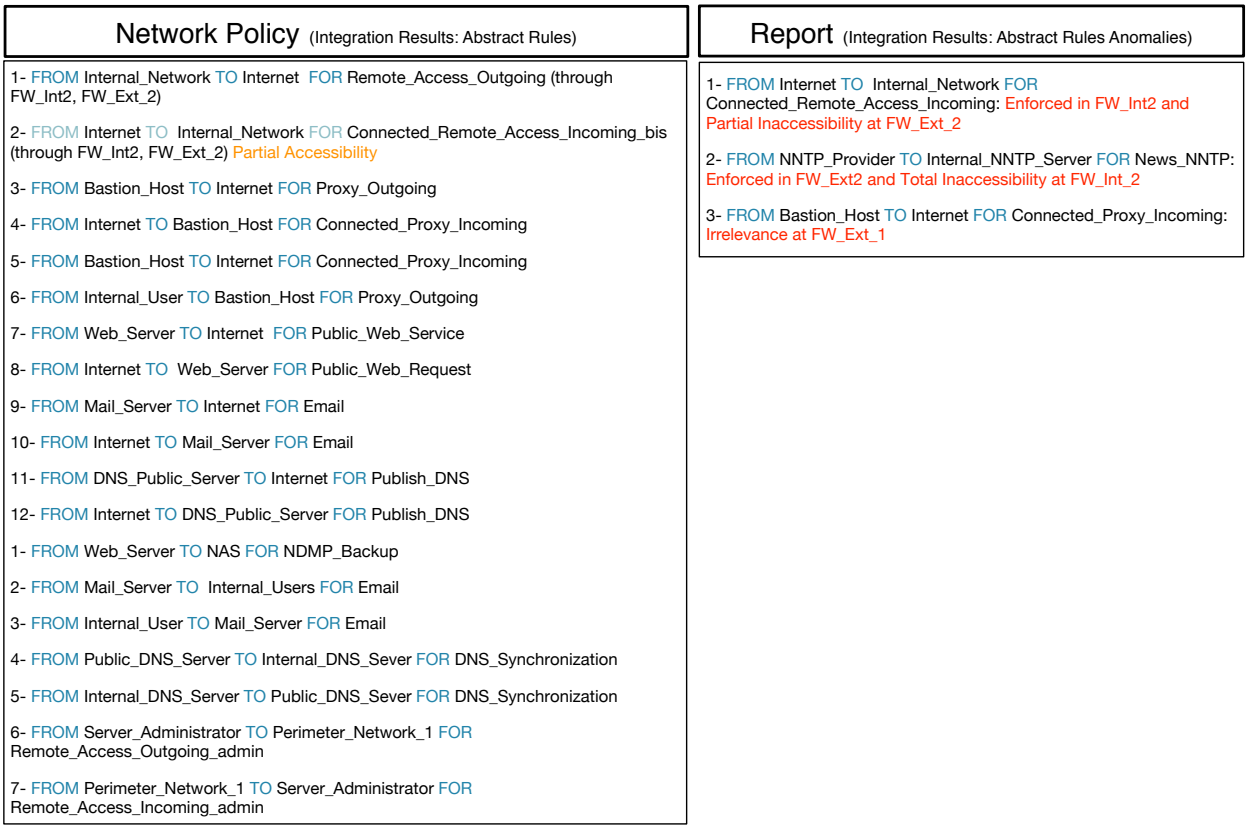


Figure 5.5: Network Policy Rules Resulting from the Integration of the Firewalls Policies.

completing the bottom-up approach framework that allows to mine a network policy modeled with the Net-RBAC model from the rules enforced in several firewalls. In this view, we have shown that it is necessary to run policy mining on each firewall apart, and integrate the policies obtained only after. We have proposed a methodology based on unifying the roles, activities and views hierarchies of the network policy through defining a poset merging algorithm. Then, we have proposed a solution to build the network high level policy rules while verifying relevance and accessibility properties. We have illustrated the whole process with a realistic example.

In addition to the high practical interest for network security management, the abstract entities integration algorithm could be used for poset merging in general. The policy integration solution could be extended to augmented versions of the Net-RBAC or OrBAC to support stateful firewalls.

Unified Abstract Entities

(Assignment of Concrete Entities To Abstract Entities)

Source_Host to Role assignment: (idem Destination_Host to View assignment:)

1- Internet = {Any}

2 -NNTP_Provider = {217.73.146.215 www.premium-news.com}

3- Internal_Network = {192.168.1.126/25}

4- Internal_User = {192.168.1.13 - 15] and 192.168.1.20}

5- Server_Administrator = {192.168.1.5}

6- Internal_NNTP_server & Internal_DNS_Server = {192.168.1.252}

7- NAS = {192.168.1.[13 -14]}

8- Perimeter_Network_2 = {192.168.1.126/25}

9- Bastion_Host = {192.168.1.5}

10- Perimeter_Network_1 = {192.168.1.254/28}

11- DNS_Public_Server = {192.168.1.245}

12- Mail_Server = {192.168.1.250}

13- Web_Server = {192.168.1.241-242}

Service to Activity assignment:

1- Remote_Access_Outgoing = {(direct SSH outgoing) : -p tcp-sport any -dport 22 -ack any, - p tcp -sport 22 -dport any -ack -yes, (FTP passive mode command channel) -p tcp -sport >1023 -dport 21 -ack any, (FTP passive mode data channel) - p tcp -sport >1023 -dport >1023 -ack any, (direct Telnet outgoing) -p tcp -sport >1023 -dport 23 -ack any}

2- Connected_Remote_Access_Incoming = { FTP (passive mode) CC {-p tcp -sport 21 -dport >1023 -ack yes}, FTP (passive mode) DC {-p tcp -sport >1023 -dport >1023 -ack yes}, Telnet incoming connected -p tcp -sport 23 -dport >1023 -ack yes, (direct SSH) { -p tcp -dport 22 -ack any, - p tcp -sport 22 -dport any -ack -yes}}

2- Connected_Remote_Access_Incoming_Bis = { FTP (passive mode) DC {-p tcp -sport >1023 -dport >1023 -ack yes}, Telnet incoming connected -p tcp -sport 23 -dport >1023 -ack yes, (direct SSH) { -p tcp -dport 22 -ack any, - p tcp -sport 22 -dport any -ack -yes}}

3 -News_NNTP = {-p tcp -sport >1023 -dport 119 -ack any, -p tcp -sport 119 -dport >1023 -ack yes}

4- Proxy_Outgoing = { (FTP passive mode CC) -p tcp -sport >1023 -dport 21 -ack any, FTP (passive mode) DC {-p tcp -sport >1023 -dport >1023 -ack yes, HTTP -p tcp -sport >1023 -sport 80 -ack any}

7- Connected_Proxy_Incoming = { FTP (passive mode) CC : {-p tcp -sport 21 -dport >1023 -ack yes}, FTP (passive mode) DC {-p tcp -sport >1023 -dport [1023-6000] & >6020 -ack yes}, HTTP -p tcp -sport 80 -sport >1023 -ack yes}

8- NDMP_Backup = {-p tcp -dport 10000}

9- Email = {SMTP -p tcp -sport >1023 -dport 25 -ack any, -p tcp -sport 25 -dport>1023 -ack yes}

10- DNS_Synchronization = { -p tcp -sport >1023 -dport 53 -ack any, -p tcp -sport 53 -dport >1023 -ack yes}

11- Remote_Access_Outgoing_admin = { SSH -p tcp-sport any -dport 22 -ack any, Telnet -p tcp -sport >1023 -dport 23 -ack any}

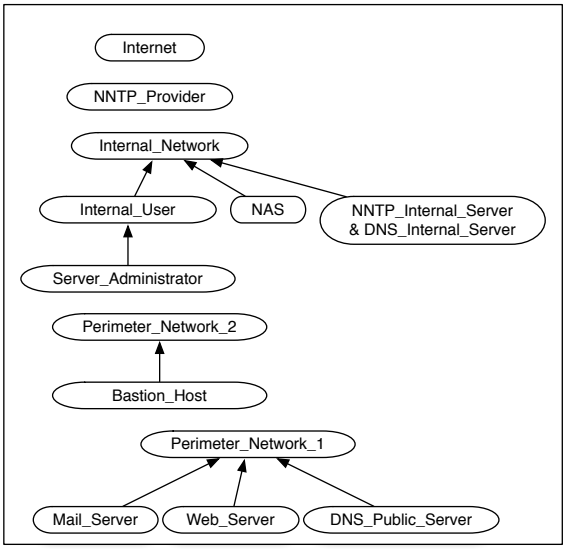
12- Remote_Access_Incoming_admin= { SSH p tcp -sport 22 -dport any -ack -yes, Telnet -p tcp -sport 23 -dport >1023 -ack yes}

13- Public_Web_Service = {HTTP -p tcp -sport 80 -dport >1023 -ack yes}

14- Public_Web_Request = {HTTP -p tcp -sport >1023 -dport 80 -ack any, HTTPS -p tcp -sport >1023 -dport 443 -ack any}

15- Publish_DNS = {-p udp -sport 53 -dport 53}

Unified Hierarchies of Roles/Views



Unified Hierarchies of Activities

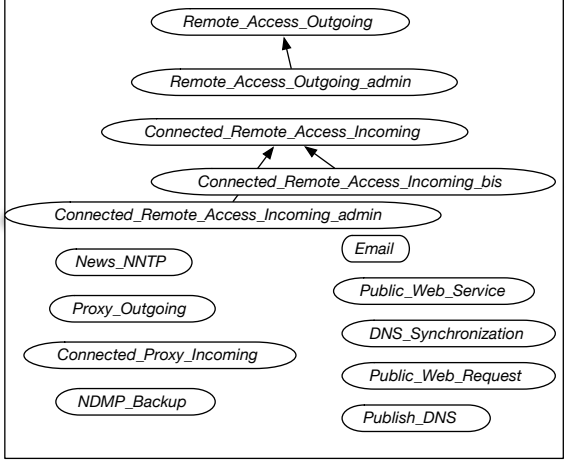


Figure 5.6: Network Unified Abstract Entities and their Hierarchies.

CHAPTER 6

Role Set Comparison and Analysis

Role mining and policy mining are likely to promote the development and the adoption of abstract access control frameworks such as RBAC and Net-RBAC in real applications. This bottom-up approach simplifies the configuration of the model, reduces the costs of role engineering, and also increases the confidence of business managers in such procedures by ensuring the conservation of the already deployed authorizations. Nevertheless, we still need to deal with some important issues before such exploitation can be possible. One of the major remaining issues is assisting the administrator in leveraging the outcome of role mining. This chapter addresses the specific problem comparing two equivalent sets of roles. We have noted that several use cases express the need for such a comparison. We have also experienced the requirement of comparing equivalent configurations of abstract entities along our research on policy mining development. We propose an automatic analytic approach that enables to understand one set of roles by reference to another.

In this chapter, we formally define the problem of comparing two sets of roles according to the underlined requirements extracted from literature (see chapter 3). We demonstrate that the problem is NP-complete. We present a greedy solution that tackles 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.

Chapter organization. Section 6.1 presents the use cases considered in this chapter and introduces a motivating example. Section 6.2 formally defines the role set comparison problem. Section 6.3 presents our solution to the problem, underlines some of the properties of our solution and elaborates some heuristics to enhance our approach. Section 6.4 concludes the chapter.

6.1 Motivation

6.1.1 Use Cases for Comparing Two Sets of Roles

The need for 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 role mining algorithm with the reverse engineering technique: starting from an RBAC configuration, then applying role mining over the resulting *UPA*, and last comparing the roles obtained with the original ones. 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 typically find 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 one 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 a subset of other 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 the privileges pertaining to him in the older 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 in forbidding access to extra-privileges to any user. 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 highly 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 necessary to find where the permissions of an old role have 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 discover how to optimally assign the permissions to the users without adding extra-permissions using the new roles.

By extension, all the above mentioned use cases for RBAC roles deriving from role mining hold also for the Net-RBAC roles, activities and views that coming from policy mining. For the sake of concision, we talk only about RBAC roles in the rest of this chapter and we provide further precisions whenever necessary.

6.1.2 Motivating Example

The following example illustrates the problem treated in the chapter. Table 6.1 shows two equivalent RBAC configurations of a hypothetical financial department of an organization. They could be an old and new representation of a subset of the the organization access control policy. We can imagine the following scenario. At point, security experts have defined the set of original roles in *OR* (see Table 6.1(a)). In this distribution of roles, they have planned that any employee who needs to make a purchase has to contact an employee from the *financial department*, who will *create a purchase-order*. Some employees in the department are in charge of *reading all the existing purchase-orders* and deciding to *create cash transfers* for the relevant orders according to their priority. An employee with higher responsibility has a *reading access to all the purchase-orders* and must *validate the cash transfers* before they become effective. We suppose that after some time, the progress in the staffing of the organization has led the user to role assignments in *UR*1(see Table 6.1(c)). In the terminology of the RBAC Definition 2.2.1, the operations are *OPS* = {*create (c), read (r), validate (v)*} and the objects *OBJ* ={*purchase-order (p-Order), cash transfer (Trans)*}. The set of permissions is then: *PRMS* = {*cTrans, rP-order, cP-order, vTrans*}. The first set of roles is *Original_ROLES* ={ *Handle Order(r1), Create Order(r2), Supervise Trans-*

Table 6.1: Motivating Example: Original and Mined Equivalent RBAC Configurations

(a) Original Roles Matrix *OR*

Original Roles	cTrans(p1)	rP-order(p2)	cP-order(p3)	vTrans(p4)
Handle Order(r1)	1	1	0	0
Create Order(r2)	0	0	1	0
Supervise Transfer(r3)	0	1	0	1

(b) Mined Roles Matrix *MR*

Mined Roles	cTrans(p1)	rP-order(p2)	cP-order(p3)	vTrans(p4)
Manage Order(R1)	1	1	1	0
Validate Transfer(R2)	0	0	0	1

(c) Original User to Role Assignment Matrix *UR*1

User-ORole	r1	r2	r3
U1	1	1	0
U2	1	1	1
U3	0	0	0
U4	1	1	0
U5	1	1	0

(d) Mined User to Role Assignment Matrix *UR*2

User-MRole	R1	R2
U1	1	0
U2	1	1
U3	0	0
U4	1	0
U5	1	0

(e) Current User to Permission Assignment Matrix *UPA*

User-Perm	p1	p2	p3	p4
U1	1	1	1	0
U2	1	1	1	1
U3	0	0	0	0
U4	1	1	1	0
U5	1	1	1	0

fer(r3)}. The combination of *OR* and *UR*1 the role-to-permission and the user-to-roles relations gives the user-to-permission access control relation *UPA* (see Table 6.1(e)): $UR1 \otimes OR = UPA$.

If we focus on the user-to-role assignment in the original RBAC configuration *UR*1(in Table 6.1(c)), we notice that $r1$ and $r2$ are always assigned to the users together, and that $r3$ is always assigned to the users with $r1$ and $r2$, which means that $p3$ permission in $r3$ does not have any impact on the *UPA* since it is also provided by $r2$ to the same user. These remarks may warn that the old configuration of roles is no more suitable to the new policy of access control and to the current distribution of responsibilities over the employees. In our scenario, we assume that security administrator has decided to apply some role mining technique on the current *UPA* matrix (see Table 6.1(e)), and has obtained a new RBAC configuration with the new set of roles *MR* (see Tab 6.1(b)). The set of mined roles is $Mined\_ROLES$ ={*Manage Order(R1), Validate Transfer(R2)*}. It contains only two non overlapping roles, $R1$ and $R2$. The new user to role assignment relation is *UR*2 (in Table 6.1(d)). Actually, most of role

mining techniques calculate only the Role to permission matrix $RP2$ and the security administrator has to assign the users to the new roles manually to get $UR2$.

The two RBAC configurations in table 6.1 are such that: $UR2 \otimes OR = UR1 \otimes OR = UPA$, which means that the two sets of roles are *equivalent* according to the following definition.

Definition 7. *Equivalent Sets Of Roles*
Two sets of roles $RP1$ *and* $RP2$ *are* equivalent *if they may lead to a same User-to-Permission-Assignment relation UPA i.e. two respective User-to-Role matrices* $UR1$ *and* $UR2$ *exist for* $RP1$ *and* $RP2$ *so that* $UR1 \otimes RP1 = UR2 \otimes RP2 = UPA$*;*

Regarding Net-RBAC, we device a similar definition for equivalent sets of Net-RBAC-abstract entities. UR matrix of user to role assignment is still the same for Net-RBAC roles. To get RP matrix, we use the 3-dimension to 2-dimension-transformation that we have presented in chapter 4 for the need of algorithm 1. For equivalent sets of activities from two Net-RBAC configurations, we replace the user to role relations in definition 7 by the operation to activity (OPA) relations as in definition 2 of Net-RBAC, and we replace the role to permission relations by the matrix obtained from the 3-dimension-to-2-dimension-transformation to get activities in term of users and objects. And so on for equivalent sets of views.

Reverting to the motivation example of table 6.1, in order to understand and analyze the new set of roles, and to assign the users to the new roles, the security administrator needs to compare the mined roles with the original roles that he used to control. Comparing the mined roles to the original roles should reveal that $R1$ is the combination of $r1$ and $r2$, i.e. $R1 = r1 \cup r2$ and that $R2$ is $r3$ minus the permission $p2$ that was shadowed in $RP1$, since has been is also assigned through role $r1$, so $R2 = r3 \cap \neg r1$. It is easy to establish manually such a relationship between the old roles and the new roles in a simple example like in table 6.1, but in real life applications, it is very hard to handle a large number of roles manually, and automatic solutions are required to perform this task. However, the existing solutions in literature are unable to find a satisfactory comparison between two sets of roles.

6.1.3 Comparing Sets of Roles in Literature

We have provided a bibliographic study of the methods proposed for the enforcement and assessment of role mining results in section 3.4 from chapter 3. In this section, we focus more on the requirements for the comparison of roles, and provide detailed description of the contributions that handle this problem to show their limitations.

In [40], Kuhlmann et al. address the problem by counting how many of the roles lying in the original reference set are exactly the same as those discovered by their proposed role mining method. Similarly, Vaidya et al. in [67] consider the average number, instead of the total one, of original roles 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, 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 as roles in the original role set. For instance, we compare the two *OR* and *MR* sets of roles presented in the previous example (Table 6.1) using the method in [40] or in [67]. Since none of the original roles has been discovered by the role mining algorithm, we will obtain Similarity(*OR*,*MR*) = 0 with both methods. This result suggests that the two sets of roles are not comparable though they are equivalent in reality.

Hassan Takabi et al. in [60], 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 criterium. In order to address similar issues, Vaidya et al. propose in [66] a series of similarity and dissimilarity metrics based on the Jaccard Coefficient and Jaccard Distance[1]. Their metrics have been recurrently used in literature to compare roles and sets of roles and to evaluate role mining algorithms [51]. Applied to our motivation example, Similarity(*MR*,*OR*)= (Jacc(R1, r1) + Jacc(R2, r3))/2 = (0.66 + 0.5)/2 = 0.58. However, these Jaccard-based measurements still experience several important limitations. First, they compute role similarity for one role from one set with respect to only one role in the compared set. In the motivation example, we get Jacc(R1,r1) = 0.66 though it would be more accurate to calculate Jacc(R1, r1 $\cup$ r2) = 1. Second, the same role may be used to be compared with several roles from the other set. Thus, as a quality performance criterium, it may artificially be affected by reference sets with a high number of closely similar roles. For instance, if we suppose that the role mining algorithm has produced another role $R3$ = {p1, p2, p4}, the similarity becomes Similarity(*MR*,*OR*) = (Jacc(R1,r1) + Jacc(R2,r3) + Jacc(R3,r1))/3 = 0.61>0.58. Thus, the role $r2$ is used twice, and the similarity metric is enhanced by the role $R3$ not useful at all.

[1]The Jaccard Coefficient and the Jaccard Distance are well-known measurements on the asymmetric information field. For non-binary data, the Jaccard Coefficient corresponds to $JC_{AB} = \frac{|A \cap B|}{|A \cup B|}$, and the Jaccard Distance to $JD_{AB} = 1 - JC_{AB}$.

Alternatively, P. Streich et al. [59] propose a one-to-one matching for the comparison of two sets, based on the average Hamming distance[2]. 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 roles discovered 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 (e.g., Hamming(r2,R1) = Hamming(r2,R2) = 2; but in reality r2 shares a permission with R1 and has nothing to do with R2).

More recently, Molloy et al. [52] 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 [14], they compare the two sets of roles generated by role mining over noisy data and then clean data. They assimilate the role mapping problem to a bipartite matching optimization problem known as *The Minimum Weighted Bipartite Matching Problem* (MWBM). Given two sets of roles SR1 and SR2, they create one vertex in $V1$ for each role in SR1 and one vertex in $V2$ for each role in SR2. For each pair of roles (R1,R2), so that $R1$ is included in $V1$ and $R2$ is included in $V2$, they add an edge (R1,R2) with Jaccard(R1,R2) weight. Then, they run an algorithm solving the maximum weighted bipartite matching problem. The solution contains the closest pairwise matching by maximizing the global similarity metric. Each role in SR1 is matched with only one role in SR2, so as to mininmize the sum of the distances. Finally, the metric equates to the sum of the distances between the matched roles. Applied to the motivation example proposed in section 6.1.2, this method will match the $R1$ role with $r1$ and $R2$ with $r3$, with a similarity metric value of 0.58. This method has addressed the problem of matching the same role in one set several times while other interesting roles are discarded. But the 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. Usually, this is not true. For instance, in the motivation example, it would be much more interesting for a security administrator to know that R1 is the union of r1 and r2, and that R2 is equal to r3 minus the permission shared with r1.

The previous work has only focused on the issue of comparing two sets of roles from the perspective of finding a similarity metric for the assessment of a set of roles with

[2]The Hamming distance between two vectors of n elements from an alphabet A: $a = (a_i)_{i\in[1,n]}$ and $b = (b_i)_{i\in[1,n]}$ is the number of elements in a that are different from the elements in b at the same positions: $Hamming\_Distance(a, b) = |\{i, a_i \neq b_i\}|$.

reference to another set of roles, and do not care about providing the administrator with assistance to analyze and leverage role mining results.

Further work is needed to provide tools that analytically compare roles. In this respect, we consider the problem of comparing two sets of roles from a new perspective in this thesis. 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 role misconfiguration. We provide a tool that can assist a security administrator in the comprehension of the mined roles and how to enforce them too.

6.2 Role Set Comparison Problem

We propose a new approach to the problem of comparing two sets of roles. Our approach targets to leverage all the relationships between roles from the two sets: merged roles, partitioned roles, hierarchical relationships, exceptions, etc. These relationships allow us to understand analytically the new set of roles while leveraging the experience with the set of roles of reference. To reach this target our method relies on expressing each role from one set as an algebraic formula of the roles from the other set. We think that the *Disjunctive Normal Form (DNF)* is a natural choice to structure this formula.

6.2.1 Problem Statement

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

Definition 8. *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, so 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 8 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. Since we present a role both as a set of permissions and as a Boolean vector in the algebraic representation, we borrow and map concepts and notations from logic and set theory to formalize the problem of comparing sets of roles and perform calculations. In particular, we assimilate the union of sets to the logic conjunction and the intersection of sets to the logic disjunction. For

instance, in the motivating example in table 6.1, $R1 = r1 \cup r2$ is equivalent to the fact that the disjunction of the vectors representing $r1$ and $r2$ in table 6.1(a) equals to the $R1$ Boolean vector in table 6.1(b). Likewise, when claiming that the $R2$ Boolean vector is included in the $r3$ Boolean vector, we mean that the set of permissions belonging to $R2$ is included in $r3$. The negation in logic is mapped to the complement set in set theory. Thus, providing that the universe of permissions in the motivating example is {p1, p2, p3, p4} and that $r1$ = p4, the negation of $r1$ noted $\neg r1$ is then the complementary of the role $r1$ corresponding to {p1, p2, p3}, and we represent it by inverting zeros and ones in the vector of $r1$. In this context, we consider that the expression $r3 \cap \neg r1$ is a DNF and that it equals $R1$.

The structure of a DNF fits naturally the requirement of projecting a R role from *OR* in *MR*. The permissions of R role 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 *MR* configuration, this means that the projection in *MR* of R or of a subset of it represents a super role. Moreover, using negation of roles in combination with conjunction aims to retrieve a role from a role so as to express exceptions. The disjunctions accumulate and cover the permissions of R when they are spread out in *MR*. The problem also states that 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 cannot always be found due to role mining errors, exceptions and shadowed roles. This issue is to be handled more in depth in the next section. The existence of the exact DNF matching, plus the DNF complexity, 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 a case, we find the DNF that involves the fewer conjunctions of roles in order to reduce the size of the hierarchical relationships obtained. In a second stage, we also minimize the number of clauses. Finally, we can note that the comparison is intended to stem from original roles to mined roles, but it can be obviously fulfilled in the opposite way.

6.2.2 Complexity

We demonstrate that the problem stated in Definition 8 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 9. *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, so that DNF equals R?

Definition 10. *Set Cover 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 6. *The* Role-to-Role Set Comparison Decision Problem *is NP-complete.*

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

1. The problem is NP.
2. Another known NP-complete problem exist, any instance of which can be reduced in a polynomial time to an instance of our problem, so that resolving our problem infers resolving the other one.

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 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. □

6.3 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.

6.3.1 Role Set Comparison Algorithm

Algorithm 4 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 expression obtained, 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 along the progress of algorithm 4. The second main loop is a *while* loop (line 9–33). It checks all the possibilities of disjunctive clauses from CandidateRoles until the role R is totally covered where the DNF obtained exactly matches R, or until all the combination of clauses are tested, where the DNF obtained is the best approximation of R. Since a DNF is a disjunction of conjunctions, it 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 5 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 to covering the UncoveredPerms more than the current DNF, or are already included in the DNF. The following running example illustrates the algorithm.

Algorithm 4 Compare Two Sets of Roles

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 5 Generate Conjunctive Clauses

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
```

6.3.2 Running Example

We examine the configurations of roles in table 6.2. We run algorithm 4 with the inputs (*MR*,*OR*). Algorithm 4 projects the roles $R1$ and $R2$ in the set of roles *OR*

Table 6.2: Running Example

(a) Original Roles Matrix *OR*.

Original Roles	p1	p2	p3	p4	p5	p6	p7
r1	1	1	0	0	0	0	0
r2	1	0	1	0	0	0	0
r3	0	0	1	0	1	1	1

(b) Mined Roles Matrix *MR*.

Mined Roles	p1	p2	p3	p4	p5	p6	p7
R1	1	1	0	0	1	1	1
R2	0	0	1	0	0	0	0

sequentially. We consider only the role R1 in this demonstration, thus, we will run the main *for* loop (line 2–35) only once. So we can focus on the rest of algorithm 4. 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 4: 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 5 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 sums up 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 4: 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 as follows: DNF $= r1 \cup (r3 \cap \neg r2)$. 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.

6.3.3 Properties of the Algorithm

Theorem 7. ***Correctness.*** *For each role R in OR, the returned DNF is included or equal to R.*

Proof. Lines 12 and 13 of algorithm 4 guarantee that the DNF is always a subset of or equal to R. □

Theorem 8. ***Completeness.*** *For each role R in OR, if a disjunctive normal form of roles from MR equal to R exists, algorithm 4 returns an exact matching DNF for R. If no exact matching DNF exists for the role R, algorithm 4 returns a maximal DNF of roles from MR included in R (with respect to set inclusion).*

Proof. Algorithm 4 tests exhaustively all the possible configurations in the worst case, enhancing the covering of R gradually. Thus the returned DNF maximally covers R. □

Theorem 9. ***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 4 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 out on 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 4 checks if any other clause previously added has become 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. □

6.3.4 Time Complexity of the Algorithm

Time complexity of algorithm 4 depends on both the size and the nature of the input data. If the inputs are two sets of roles of size n and m roles respectively, then the worst case complexity is $O(n \times 4^m)$. The worst case is met when the two sets of roles are not comparable, meaning that for each role in setR1 there is no DNF of roles from setR2 that exactly matches with it. The algorithm will test, for each of the n roles of the first set, all the possible configurations of disjunction of conjunction of the m roles in setR2, increasing gradually the number of roles in the conjunction from one to $2m$, and the output DNFs will only signify approximations of roles. Thus, we calculate the worst case complexity as $O(n \times \sum_{k=1}^{2m} C_{2m}^k) = O(n \times 4^m)$, with $C_n^k = \frac{n!}{k! \times (n-k)!}$.

The best case complexity is $O(n \times 2m)$, when the two sets of roles are identical. In such a case, for each of the n roles from the first set, algorithm 4 glances through the set of m roles and their negations only once and finds the exact match. We note that the complexity is independent from the number of permissions and users.

6.3.5 Heuristic to Reduce Complexity

Algorithm 4 solves an NP-complete problem. The worst case time complexity is high. However, more than the size of the input data, time complexity depends on the nature of this datum. Indeed, complexity is exponential on the number of roles from the second set when the compared sets of roles are incomparable, whereas the complexity is much lower when the sets of roles are comparable. Since we intend to use the algorithm in a context of role mining where roles are comparable, we expect that the exact matching for each role exists and will be found. In addition, it will be generally found at a low level of ConjunctiveClauseLevel. Otherwise, we can guess that no matching DNF can be found for the handled role, and stop exploring further combinations of roles prematurely. An intuitive heuristic to reduce the time complexity of the algorithm is to set a limit for the ConjunctiveClauseLevel. We provide the user of the algorithm with the possibility to set a threshold, in order to avoid algorithm exploration of all the combinations involving clauses of cardinality beyond this threshold. Besides, such an option may be required by the user. Indeed, regarding the DNFs obtained and their management security administrator may prefer simplicity and shortness to complexity and accuracy. The threshold can be given as input to the algorithm and taken into account with a minor modification of algorithm 4 by adding the condition (ConjunctiveLevel $\leq$ threshold) to line 9. Algorithm 6 is a new version of algorithm 4 where we show only the modified lines. To enhance the accuracy of matching between each role and its approximated DNF, the user can increase the threshold gradually.

Algorithm 6 Compare Two Sets of Roles with Heuristic

Input: nRoles1×nPerms reference role-permission relation setR1

Input: nRoles2×nPerms compared role-permission relation setR2

Input: threshold is an integer that represents the maximum supported number of roles in a conjunctive clause in the returned DNF

Output: a DNF of roles from setR2 for each role in setR1. The DNF is equal to the role, or the closest included possible, with no more than *threshold* roles in each conjunctive clause of the DNF.

...

9: **while** ¬RIsCovered and CandidateClauses $\neq \emptyset$ and ConjunctiveClauseLevel $\leq$ threshold

...

35: **return** DNF for role R

For n mined roles, m original roles, and a threshold t of the maximal number of roles in a conjunction, the worst case complexity becomes $\mathrm{O}(n \times \sum_{k=1}^{2t} C_{2m}^{k}) = \mathrm{O}(n \times 4^m \times$

$\frac{t}{m}$). The heuristic slightly decreases the execution time since t will be generally set much lower than m, complexity still being high. But this heuristic is applicable in an exhaustive case, to any sets of roles provided as input. To enhance time complexity more significantly, we have to focus on the input data characteristics. We have to formally study the notion of *comparable* sets of roles, and its impact on the time complexity of the algorithm. In particular, since our intended context of application is role mining, we need to characterize more specifically the use cases related with role mining, where the exact match DNFs could or could not be found. Then we will be able to preprocess the input data accordingly, with the objective to avoid providing roles with no exact match DNFs to Algorithm 4. This approach is detailed in the following chapter.

6.4 Conclusion & Perspectives

In this chapter we have proposed a solution that allows appraising and leveraging the learning outcomes of the role mining process by comparing the obtained mined roles with a reference set of roles which may be the set of original roles, or the set of permissions assigned to each user. This solution applies also for policy mining and the obtained mined abstract entities. 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 evaluated the time complexity of our algorithm.

We have demonstrated that the complexity of the proposed solution depends on the nature of the input data. In the next chapter, we will insist on characterizing *comparable* sets of roles. In particular, we will investigate the correlation of some source misconfiguration of roles with the complexity of the comparison between equivalent roles.

CHAPTER 7

Shadowed Roles Detection

Under RBAC as well as under Net-RBAC configurations, several problems of role or abstract entities misconfiguration may occur when they are manually configured and/or after several local modifications. In the last chapter, we have raised the following: some source role misconfiguration may have an important impact on the performance of the role set comparison tool.

In this chapter, we first investigate the characteristics of the input data and their impact on algorithm 4 of role set comparison. We characterize some properties of roles that guarantee the existence of the exact matching DNFs under realistic assumptions related to role mining context. We demonstrate the existing correlation between the misconfiguration problem of shadowed roles and the complexity of execution of the role set comparison algorithm. Second, we focus on shadowed roles detection as a standalone problem. We provide a definition of the problem of shadowing. Finally we propose an algorithm that identifies and reports shadowed roles in a given configuration of roles.

Chapter organization. Section 7.1 investigates the correlation between the presence of source shadowing misconfiguration and the results of the role set comparison. Section 7.2 provides a definition of shadowing and presents the shadowed roles detection problem. Section 7.3 proposes an algorithm for shadowing detection. Section 7.4 closes the chapter.

7.1 Correlation between Shadowed Roles and Role Set Comparison Results

Based on our analysis of time complexity of algorithm 4, the execution time is expected to be low when the exact matching DNFs exists for all the roles. We are interested in characterizing the input data properties that guarantee the existence of the exact matching DNFs. Besides, though algorithm 4 can compare two arbitrary sets of roles defined over the same set of permissions, it is intended to be used in the context of role and policy mining. Hence, we focus on the cases related to role mining, where the two sets of roles provided as input to algorithm 4 are *equivalent* (see definition 7). Back to definition 1 of RBAC, a role is a structure characterized by a set of assigned permissions and a set of assigned users. Likewise in definition 2 of Net-RBAC, each abstract entity (role, activity, view) is characterized by the set of concrete entities (user, operation, object) assigned to it, and also by the permissions in which it is involved.

Concerning the comparison of sets of roles, we have considered the roles from the perspective of their permissions only up to this point. Role engineering usually defines the roles by their permissions, because they are more stable than the users in an organization, and they may reflect a job function better for example. Nevertheless, to fully understand the configuration of roles and to characterize accurately their properties we need to consider them from their perspectives related to both their users and permissions. Similarly, the abstract entities in Net-RBAC are considered from both perspectives of assigned concrete entities and assigned permissions. To that aim, we use the 3-dimension to 2-dimension-transformation that we have presented in chapter 4 for the need of algorithm 1, to obtain the relation of each abstract entity with the combination of the two other concrete entities (i.e. role-(operation, object), activity-(user-object), and view-(user,operation)).

Theorem 10. ***Constrained Completeness Guarantees.*** *Given two* equivalent *sets of roles RP1 and RP2; For any role R from RP1, if a subset of k users exist so that* $\bigcap_{i=1}^{k} assigned\_perms(U_i) = R$*, then at least a Disjunctive Normal Form DNF of roles from RP2 that exactly matches with R exists too.*

Proof. Since the two configurations of roles are *equivalent* according to Definition 7 , the k users who have exactly the permissions of the role R in common, must have exactly the same permissions in the two equivalent RBAC (or Net-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 [31], we can transform the obtained CNF to a DNF. □

Theorem 10, in combination with the Completeness Theorem 8, states a sufficient condition that guarantees the existence of exact matching DNFs for the roles. In the case of role mining without errors, the original and the mined sets of roles are *equivalent*. Algorithm 4 finds for each role from one set, that satisfies the condition mentioned in Theorem 10 (there is a subset of k users so that $\bigcap_{i=1}^{k} assigned\_perms(U_i) = R$) a DNF of roles from the other set which exactly matches with it. We note that this condition is independent of the role mining technique used to calculate the sets of roles. 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.

The following corollary states that the aforementioned condition is satisfied by the roles belonging to a non overlapping configuration of roles:

Corollary 1. ***Completeness Guarantees for Non Overlapping Roles.*** *Given two "equivalent" sets of roles RP1 and RP2; If the roles of the set RP1 are not overlapping, then, for any role R in RP1, if R is assigned at least to a user, and there is no other role $R' \in$ RP1 so that assigned_users(R) $\subseteq$ assigned_users(R'), at least a Disjunctive Normal Form DNF of roles from RP2 that exactly matches with R exists.*

Proof. reductio ad absurdum:
Assume that $\bigcap_{U_i \in assigned\_users(R)} assigned\_perms(U_i) = R \cup P$ with P a set of permissions such that $P \cap R = \emptyset$ and $P \neq \emptyset$. Then P is the intersection of the permissions assigned to all the roles assigned to the users assigned to R, minus the permissions assigned to R.
Formally: $P = \bigcap_{U_i \in assigned\_users(R)} (\bigcup_{R_j \in assigned\_roles(U_i) \setminus R} assigned\_perms(R_j))$. However, there is no role shared by all the users assigned to R other than R. Since the roles are non overlapping, then $P = \emptyset$.
Consequently, $\bigcap_{U_i \in assigned\_users(R)} assigned\_perms(U_i) = R$, and R satisfies the condition of Theorem 10. □

Thus, if algorithm 4 does not find an exact matching DNF for a given role in a context of role mining, this means that the conditions of Theorem 10 or Corollary 1 are not fulfilled and can be interpreted as a possible misconfiguration of that role in its original set of roles. The first possibility being that the User-to-Role assignment is incomplete, and that this role has not been assigned to any user yet. For instance,

a job position has been created but the assigned employees are not hired yet. The second possibility being that the role is always assigned to the users simultaneously with another role, so the two roles can be merged in a single role. Another possibility may occur: the role overlapping with other roles in the set, and some permissions of the role are shadowed by other roles. This can happen if the role is always assigned to users who receive the shadowed permissions from their other assigned roles simultaneously. The shared permissions can thus underline a misconfiguration of the role or a non optimal hierarchy in the structure of roles. To summarize, when used to compare two sets of *equivalent* roles, Algorithm 4 can warn about shadowed roles in the configuration of roles when an exact matching DNF for a certain role is not found.

But the conditions of Theorem 10 or Corollary 1 are still restrictive and do not exhaustively specify when exact matching DNFs exists. For instance, they cannot tell about overlapping roles. Besides Algorithm 4 can not exhaustively detect all the shadowed roles. Indeed, if conditions are satisfied, then the algorithm will find exact matching DNFs for the roles, but if the Algorithm finds an exact matching DNFs this does not mean that the conditions are satisfied by the role. There may be still some undetected shadowed roles. Moreover, the detection of shadowed roles with Algorithm 4 is very costly in time. And the algorithm cannot tell if the role is entirely or partially shadowed. It would be more interesting to detect shadowed roles separately from the comparison algorithm, and then, handle the misconfiguration cause or discard the shadowed roles from their configuration, before comparing it with another configuration of roles. This would enhance the efficiency of the comparison algorithm. Besides, independently from the comparison of roles, exploring the shadowed roles in a given configuration of roles could be a useful application for a security administrator. For all these reasons, we now show how to handle the shadowing problem separately from the role set comparison problem.

7.2 Definition of Shadowed Roles

In order to address the problem of shadowed roles detection, we have first to provide our formal definition of shadowed roles in a given role based access control configuration.

Definition 11. *Shadowed Role*
A role R is shadowed *in a given RBAC configuration $RC = (ROLES, UR, RP)$ if it matches one or several of the following cases:*

1. *R is not assigned to any user in UR: i.e. $assigned\_users(R) = \emptyset$.*
   or

2. *There is (at least) another role R' in ROLES that has always similar user assignment as R: i.e. assigned_users(R) = assigned_users(R').*
   or

3. *R is overlapping with one or several roles in ROLES, and there is (at least) a permission $p \in$ assigned_perms(R) so that the users assigned to R have always p in their other assigned roles.*

The idea behind Definition 11 is as follows: a role is considered as shadowed in a given access control configuration if we cannot find it in the user-to-permission assignment relation *UPA*. Typically, a role mining algorithm is not likely to calculate such a role since it is based on the *UPA* relation to extract the roles. The first case of shadowing is when the role has not been assigned to any user. The second case of shadowing appears when the role is always assigned together with another role, which may indicate that it may be merged with another role. Finally, the third case of shadowing may occur only in RBAC (respect. NSRBAC) configuration with overlapping roles. Different roles may share a subset of permissions. Shadowing happens when one or several permissions of a given role are never assigned to a user who has not yet been assigned this permission through his other roles. In this case, removing the permission from the role will not affect the *UPA* relation. An example of this situation is provided in table 6.1(a) of the afore presented motivation example in the last chapter. Permission $p2$ is shared by the roles $r1$ and $r3$. However, role $r3$ is never assigned to a user who has not also got role $r1$. So permission $p2$ is actually shadowed in role $r3$.

7.3 Shadowed Roles Detection Algorithm

We define a new algorithm that examines an RBAC configuration and detects the shadowed roles.

Algorithm 7 takes as input the matrices of user-to-role assignment *UR* and role-to-permission assignment *RP* of an RBAC configuration. We consider that a role is characterized by the set of permissions assigned to it, since the user-to-roles assignment is more dynamic and changes more often. The set of roles is then the rows of the matrix *RP*. The algorithm checks the roles in the *RP* matrix for the three cases of shadowing presented in Definition 11 and reports for each role whether shadowed or not. For partially shadowed roles, the algorithm also reports which are the affected permissions.

First of all, algorithm 7 calls the function *Check For Partitions Of Roles* (line 1). This function checks all the roles of the configuration for the second case of shadowing

Algorithm 7 Detect Shadowed Roles

Input: *UR*: a user to role assignment matrix

Input: *RP*: a role to permission assignment matrix

Output: A specification for each role in *UR* matrix: whether it is "not shadowed", "not assigned", or "shadowed" with the specification of the "shadowed permissions".

```
1:  CandidateRoles ← Check For Partitions Of Roles(UR)
2:  NBUPA ← UR × RP
3:  RecurrentPerms ← permissions assigned to a user more than once in NBUPA
4:  if RecurrentPerms ≠ ∅ then
5:    for each role R in CandidateRoles do
6:      RIsShadowed ← False
7:      ShadowedPerms ← {}
8:      if assigned_users( R ) ≠ ∅ then
9:        Report: "The role <R> is not assigned to any user."
10:     else
11:       ProblematicPerms ← assigned_perms(R) ∩ RecurrentPerms
12:       for each permission p in ProblematicPerms do
13:         OnceUsers ← assigned_users( R ) ∩ assigned_users_only_once( p)
14:         if OnceUsers == ∅ then
15:           RIsShadowed ← True
16:           ShadowedPerms ← ShadowedPerms ∪ {p}
17:         end if
18:       end for
19:       if RIsShadowed then
20:         Report: "The role <R> is shadowed. The shadowed permissions are
    <ShadowedPerms>."
21:       else
22:         Report: "The role <R> is not shadowed."
23:       end if
24:     end if
25:   end for
26: else
27:   for each role R in CandidateRoles do
28:     if assigned_users( R ) ≠ ∅ then
29:       Report: "The role <R> is not assigned to any user."
30:     else
31:       Report: "The role <R> is not shadowed."
32:     end if
33:   end for
34: end if
```

Algorithm 8 Check For Partitions Of Roles

Input: RP: a role to permission assignment matrix

Output: NonPartitionsRoles

Output: This algorithm should check for Roles always assigned together to the same users and report them as shadowed roles.

1: Apply a Hash function on each column of RP
2: Sort the columns of RP according to the Hash values
3: NonPartitionsRoles ← Roles with a unique instance of Hash Value
4: Report: "The roles in <RP∪NonPartitionsRoles> are shadowed. They are partitions of roles."
5: **return** NonPartitionsRoles

stated in Definition 11. It detects the roles with similar sets of assigned_users. These roles are partitions of a larger role. Each column in the matrix *UR* represents the set of assigned_users of a role. Function *Check For Partitions Of Roles*, whose pseudocode is presented in Algorithm 8 applies a hash function to each column in the matrix, then it sorts the columns based on the hash values. Thus roles that share exactly the same set of assigned_users are gathered since they have the same hash value. Each subset of roles sharing the same users is reported as partitions of a single role. The subset's roles are reported as *Shadowed roles*. The function returns the remaining roles, that present unique hash value, because they do not fit the second case of shadowing considered in Definition 11. They become *CandidateRoles* to be checked for the two other cases (1 and 3) of shadowing in Definition 11.

CandidateRoles may be totally shadowed if they are not assigned to any user, and this is easy to check. By contrast, the third case of shadowing, is more complex to be verified because a role may be partially shadowed by one or several roles. Only overlapping roles are potentially affected by this case. The partially shadowed roles involve necessarily overlapping roles that have been assigned simultaneously to a set of users. Consequently, a subset of permissions is assigned multiple times to the same users. This is why we use the algebraic multiplication instead of the Boolean multiplication traditionally used in role mining issues: the algebraic multiplication figures out how many times the same permission has been assigned to a user. We calculate the non Boolean user to permission assignment matrix *NBUPA* (line 2). In this matrix, each column represents a permission, each row represents a user, and cell c_{ij} represents how many times permissions p_j has been assigned to the user u_i. Then, from this new matrix *NBUPA*, we deduce the set *RecurrentPerms* (line 3) that contains the permissions assigned to a user, through more than one role. This set represents the only potentially shadowed permissions.

If set *RecurrentPerms* is empty, then no overlapping roles assigned simultaneously to the users exist, and consequently no roles are affected by the third case of shadowing in the Definition 11. We move to lines 26 to 34 where we check if each role has been assigned at least to one user. If *RecurrentPerms* is not empty (line 8), then there are overlapping roles which are assigned together to users in the input RBAC configuration. So there is a risk of partially shadowed roles. Algorithm 7 handles the *CandidateRoles* one by one in the *for* loop in line 5. First, it checks if the role has been assigned at least to a user. The algorithm simply looks at the column corresponding to the role in the *UR* matrix to get the assigned_users(R). If the column is null, then the role has not been assigned to, and is considered shadowed and we move to the next role. If there is a non null value in this column the role is assigned to one or several users, and we need to check if some of the assigned_permissions(R) are shadowed. We have to handle the permissions assigned to the current role one by one to ensure that each permission has been assigned at least to one user only from the role R. This proves that the permission is not shadowed in the role. The role is not shadowed only if all its permissions assigned to R are not shadowed in it. Otherwise, the role is shadowed, and the current permission is one among the shadowed permissions. But hopefully, we do not need to check all the permissions assigned to a role since only the set of *RecurrentPerms* are potentially shadowed permissions. So, for each role R, we calculate the intersection between its assigned_roles(R) and the *RecurrentPerms*, and we call it the *ProblematicPerms* (line 11). Finally, for each problematic permission, we calculate the intersection between the set of users assigned to the role, and the set of users to whom the permission has been assigned from only one role, which we can find directly in the matrix *NBUPA* (see line 13). If the set is empty then the role is shadowed, and we add this permission to the set of *ShadowedPerms* in the current role, before moving to explore the remaining permissions in *ProblematicPerms*, so as to get the exhaustive set of shadowed permissions in this role. Otherwise, the current permission is not shadowed in this role, and the algorithm continues testing the other roles in *ProblematicPerms*. The role is reported to be unshadowed only if all the *ProblematicPerms* are not shadowed in it.

Theorem 11. ***Completeness.*** *Given two matrices UR and RP; For any role R from RP, if R fills one or several of the shadowing cases specified in Definition 11 then Algorithm 7 will report it as a shadowed role.*

Proof. The function *Check For Partitions Of Roles* (Algorithm8) checks all the roles for the case 2 of shadowing. All the roles not contained in the case 2 of shadowing are set to *CandidateRoles*. In Algorithm 7, all the *CandidateRoles* are checked for the case of shadowing, either in the for loop (lines 5 to 9) if the roles are overlapping or

in the for loop (lines 27 to 30) if the roles are not overlapping. Case 3 of shadowing is possible only in overlapping configurations of roles, and only overlapping roles are assigned together to the same users. Then, the *If* condition (line 8) ensures that all the *CandidateRoles* are checked for this case of shadowing only if permissions assigned more than once to a user exist. These roles not concerned with the case 1 of shadowing are tested for the case 3 of shadowing. For each of these roles, the algorithm verifies that each permission of the role has been assigned at least once to a user whose other roles do not grant this permission. Otherwise the role is reported to be shadowed. □

Computational Complexity Algorithm 7 is a light application that permits to detect and report the shadowed roles in a given RBAC configuration. Its time complexity is polynomial since it is O(n×m×k) with n the number of users, m the number of permissions and k the number of roles in the input RBAC configuration. We are very far from the exponential complexity of Algorithm 4. Moreover, we can avoid the worst case complexity of Algorithm 4 if we preprocess its input with Algorithm 7.

7.4 Conclusion

We have questioned the problem of detecting shadowed roles. Shadowing is usually due to a misconfiguration problem. We have formally defined the shadowing cases. We have provided a solution to detect shadowed roles. We have formally linked the problem of shadowed roles to the problem of comparing sets of roles. Indeed, in most cases related to role mining, preprocessing the compared sets of roles by eliminating shadowing from the set of roles significantly lowers the complexity of the RSCP solution.

CHAPTER 8

Experimental Evaluation

This chapter is dedicated to the experimental results of the algorithms presented in chapters 3-6.

8.1 Platform of Test

We have implemented a proof of concept platform for the assessment of all the contributions presented in this thesis. All the algorithms have been implemented in MATLAB_R2011a. The performance experiments have been run on a 2.26 GHz Intel Core 2 Duo processor on Mac OS X V 10.7.2.

8.1.1 RBAC and Net-RBAC State Generators

We have implemented a random RBAC configuration generator to obtain the synthetic data input for the 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. UR and RP matrices are randomly generated based on the provided densities which determine the probability to get 1 in a particular cell. The obtained configuration may contain partially overlapping roles and may assign multiple roles to a user. The outputs are the three Boolean matrices UR, RP, and UPA where $UPA = UR \otimes RP$.

Likewise, we have implemented a random Net-RBAC configuration generator. This tool takes as input the number of users, operations, objects, roles, activities and views, in addition to the density of the concrete-to-abstract entities assignments and the density of the abstract rules, and generates randomly matrices UR, OPA, OBV and

RAV of a Net-RBAC configuration. Matrix *UOO* of concrete authorizations is derived from the four other matrices.

8.1.2 Factorization Methods

We have adapted several existing techniques usually used for matrix factorization in order to get representative samples of the results of role mining techniques with different properties and objectives.

SVD

The *Singular Value Decomposition (SVD)* [37] decomposes a matrix into its eigenvectors and calculates the corresponding eigenvalues. The eigenvectors are the directing patterns in the initial matrix, and the respective eigenvalues indicate the strength of each eigenvector in the construction of the matrix. We assimilate them to the abstract entities. We keep only the eigenvectors corresponding to the largest eigenvalues to recompose the initial matrix, while we discard the weak patterns which may be assimilated to errors or exceptions. We set a threshold to decide on the amount of discarded data automatically. We iteratively add the eigenvalues one by one following a decreasing order of corresponding eigenvalues. At each iteration we measure the distance between the recomposition and the initial matrix. We stop adding new eigenvectors when the distance increases by less than the threshold. The problem with this method is that the factorization is not Boolean. We bypass it by using another binarization threshold. We set the values higher than this threshold to one, and the lower to zero. We have opted for SVD for its simplicity that allows us to control different parameters as well as for its available efficient implementation enabling good scalability, and finally as an example of factorization method with approximation errors.

BNOF and NMF

The *Non Negative Matrix Factorization (NNMF)* [41] and the *Binary Non-Orthogonal Decomposition (BNOF)* [38] embody two techniques of matrix compression often utilized in data mining. Both techniques take an *UPA* matrix from a data set and give an approximate 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

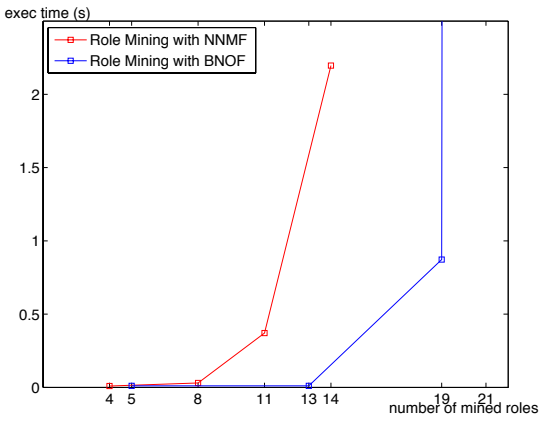
aims to minimize the root-mean-squared and results in a factorization into two matrices of non negative integers. We transform the positive matrices obtained into Boolean matrices just as we did with SVD. BNOF uses another decomposition technique which gives a Boolean decomposition. The roles obtained may be assigned to overlapping permissions, but a user can be assigned to only one role.

FCA

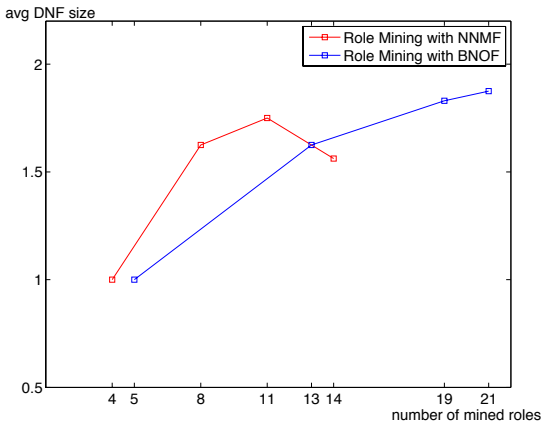
The fourth role mining algorithm that we use is a *Formal Concept Analysis (FCA)* algorithm [13, 30]. A formal context represents binary relations between objects and attributes. A formal concept is defined by a set of objects and a set of attributes which represent mutually all the objects that share the given attributes, and all the attributes shared by the given objects. It may be assimilated to an abstract entity in Net-RBAC model. For instance, if we consider the users as objects, and the permissions as their attributes, each calculated formal concept is a candidate mined role. Several efficient algorithms exist to calculate all the formal concepts corresponding to a given formal context. We have used the implementation described in [39]. 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. Sophisticated criteria to select the relevant concepts have been designed in role mining literature [50]. In this thesis, we keep all the concepts for experiments related to the role set comparison algorithm in section 8.2. For experiments of policy mining in section 8.4 we simply assign to each concrete entity the minimal set of formal concepts to cover all its attributes. We have chosen to use FCA as an example of exact factorization method and of hierarchical role mining.

8.2 Experiments with Role Set Comparison Algorithm

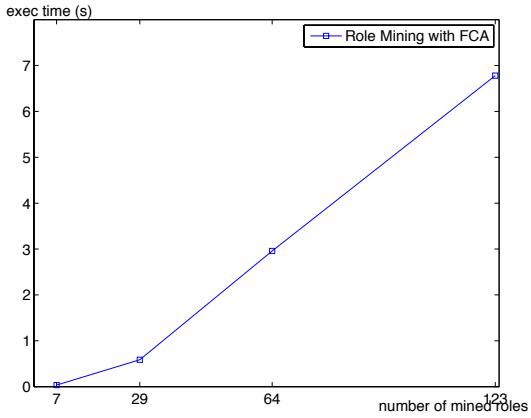
For this section, we have generated eight data sets with different sizes described in table 8.1 with the density parameter set to 0.1. First, we use the data sets 1, 2, 3 and 4 in table 8.1 for role mining tests. We provide the *UPA* matrix of each data set as input to a given role mining algorithm, and we compare the original roles in *UR* with the mined roles in UR^* using algorithm 4. We test three of the different role mining algorithms, namely those based on the *Non Negative Matrix Factorization (NNMF)*, the *Binary Non-Orthogonal Decomposition (BNOF)*, and the *Formal Concept Analysis (FCA)*. The results of the experiments are summarized in figure 8.1. The results of the first series of tests with NNMF and BNOF are shown in figures 8.1(a) and 8.1(b). The purpose is to



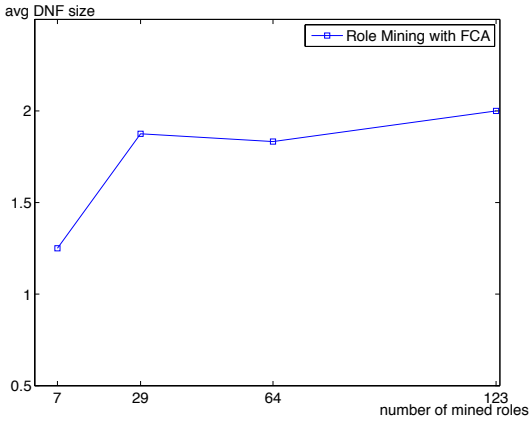
(a) Execution time when comparing roles of the RBAC data sets 1, 2, 3 and 4 with the mined roles by NNMF and BNOF



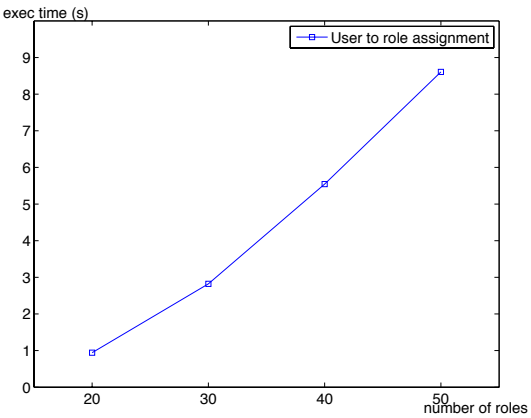
(b) Average DNF size with NNMF and BNOF



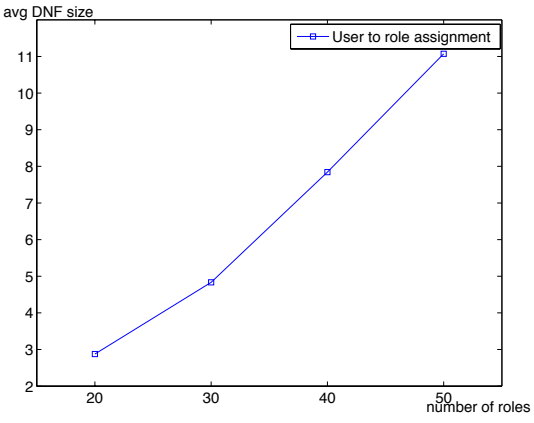
(c) Execution time when comparing roles of the RBAC data sets 1, 2, 3 and 4 with the mined roles by FCA



(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 8.1: Experimental Results with Role Set Comparison Algorithm

Table 8.1: Size of the Original RBAC Data Sets Used for the Role Set Comparison Algorithm Experiments

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

observe the behavior of algorithm 4 when comparing two sets of roles which have been configured with different objectives. All role mining processes presented in this paper have ended 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 8.1(a), we plot the execution time of algorithm 4 in terms of the number of mined roles because, as stated in section 6.3.4, the complexity of the algorithm mainly depends on the 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 fourth data set (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 4 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 on the nature of the compared roles and how similar they are. In figure 8.1(b), we plot the average size of the obtained DNFs for each data set. By size of DNF we mean here the number of roles involved, 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 increases from one to three in this series of tests.

The third role mining algorithm we use is a *Formal Concept Analysis (FCA)* algorithm. 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 section, we have decided to keep all the concepts. This explains the high number of mined roles increasing from 7 to 123 in figure 8.1(c) and 8.1(d), corresponding to

sets of original roles of only 4 to 16 roles. Our purpose is to test the ability of algorithm 4 to compare a hierarchical configuration of roles with a flat configuration of roles. We aim also at testing the scalability of algorithm 4 in such cases. The results in figures 8.1(c) and 8.1(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 DNFs obtained is still between 1 and 2, meaning that algorithm 4 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 8.1 to test the ability of algorithm 4 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 8.1(e) and 8.1(f). Figure 8.1(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, the average size of the DNFs presented in figure 8.1(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 of the four tests.

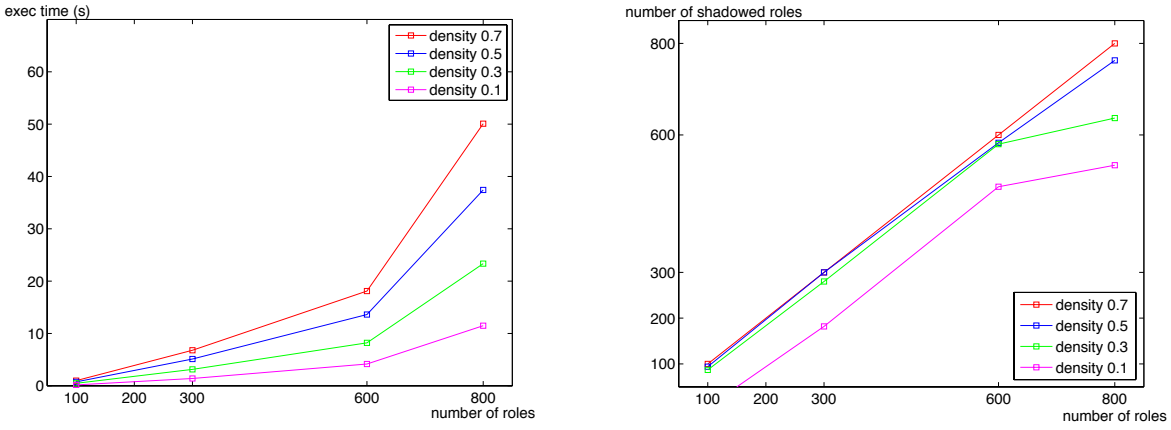
8.3 Experiments with Shadowed Roles Detection Algorithm

To test the Shadowed Role detection tool described in algorithm 7, we have generated a set of RBAC configurations. The sizes of these configurations are presented in table 8.2. For each size, we have generated a series of RBAC configurations with different values of the density parameter, namely: 0.1, 0.3, 0.5 and 0.7. Figure 8.2 summarizes the results from these experiments.

Table 8.2: Size of RBAC Data Sets Used for the Shadowing Detection Algorithm

n	nUsers	nPerms	nRoles
9	800	1000	300
10	1000	1200	600
11	1500	2000	800

In figure 8.2(a), we show the execution time of the algorithm with each of the data sets in terms of the number of roles in the RBAC configuration. We can observe that the execution time increases with the number of roles, but is still low (few seconds) even in the higher case of 800 roles. The number of roles in real world applications may be around one thousand. Moreover, since we have generated sets of roles with similar sizes but different densities, we can notice in figure 8.2(a) that the execution time is higher for sets of roles with higher density. By increasing the density, the number of shadowed roles increases in the configuration, especially the number of partially shadowed roles. The random decision of the generator in the assignment of the permissions to the roles is likely to result in more misconfiguration of roles than in real configurations. Algorithm 7 is sensitive to the existence of overlapping roles because it performs an additional processing in this case (see algorithm 7 - line 8). Figure 8.2(b) presents the number of shadowed roles detected by the algorithm 7 and confirms that the number of shadowed roles increases when the density increases.



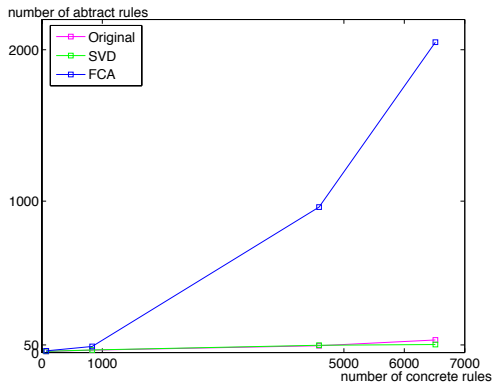
(a) Execution time as function of the number of roles

(b) Number of shadowed roles in the data sets

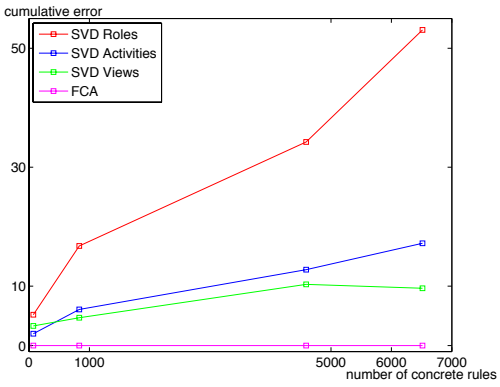
Figure 8.2: Experimental Results with Shadowed Roles Detection Algorithm

8.4 Experiments with Policy Mining Algorithm

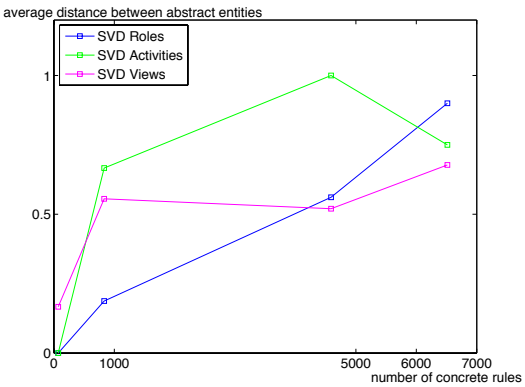
Our experiments follow this scenario: we generate a random Net-RBAC configuration, we calculate the deriving *UOO* from it, then we perform policy mining on the *UOO* relation to extract a Net-RBAC configuration in a reverse engineering process. We have generated four data sets with increasing sizes and with the density parameters set to 0.1. The results of the experiments are summarized in figure 8.3.



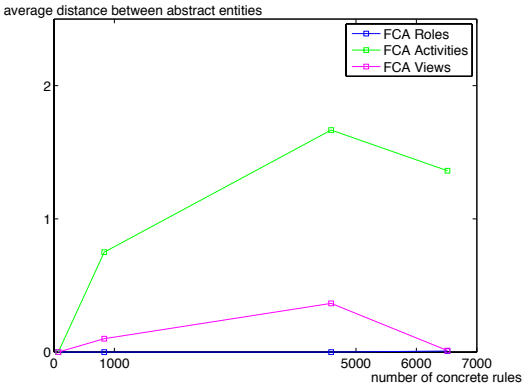
(a) Compacity of the mined abstract rules vs the original concrete rules



(b) Cumulative error with approximate factorization method



(c) Changing abstract entities mining order effect on the mined abstract entities with SVD



(d) Changing abstract entities mining order effect on the mined abstract entities with FCA

Figure 8.3: Experimental Results with Policy Mining Algorithm

Compacity of the abstract rules vs the concrete rules

Figure 8.3(a) shows the number of abstract rules obtained via policy mining in terms of the number of concrete rules in each of the four synthetic data sets. We note that the number of abstract rules is much lower than the number of concrete rules, especially with the SVD factorization method, and with all the four synthetic data sets. Policy mining obviously aggregates the firewall rules.

Cumulative error with approximate factorization method

Since performing three successive factorizations in algorithm 1, we were interested in error development when the factorization method is approximate. We have plotted the

distance between the initial concrete rules and the obtained concrete rules after each of the factorizations that calculate the roles, then the activities and finally the views (lines 2,6 and 10 in algorithm 1). We use the Frobenius norm ($||X||_f = \sqrt{\sum x_{ij}^2}$) to measure the error. The results are presented in figure 8.3(b). When using the SVD method, we note that the error increases with the data size. However, the cumulative error is not increasing. On the contrary, it tends to decrease. For example, the divergence from the initial set of rules after the factorization of the activities tends to be lower than the divergence after the factorization of the roles. When using the FCA factorization method, there is no factorization error as expected.

Mining abstract entities in a different order

In algorithm 1, we have chosen to calculate the roles first, then the activities and finally the views. This option was totally arbitrary and can be changed by a simple permutation of the dimensions of the input matrix UOO. We were interested in examining the effect of changing the order of mining on the abstract entities obtained. Figures 8.3(c) and 8.3(d) show the distance between the roles, activities and views obtained when mining roles then activities then views and when mining activities then roles then views for the same input set of rules. The entities obtained are not the same either with the approximate or the exact factorization method. It is admitted that several factorizations are usually possible for the same set of data. Optimization objectives make the difference. We could orientate the resulting abstract entities if we used factorization methods with more precise objectives in real applications.

Tests On Real Data

In addition to the experiments on synthetic data sets, we have tested policy mining on the real set of firewall rules presented in figure 4.4. First we have parsed all the instances of source_hosts to build the set of users, for example *192.168.1.126/25* is a user. We have found four different source_hosts that constitute the set of users. We note that, since some rules are defined from *any* source_host then *any* is also considered a user. We have done the same operation with services and dest_hosts. We have found twenty different operations and ten objects. We have built the UOO matrix of size $4 \times 20 \times 10$ as a 3D-matrix of zeroes, then we have set to one cells corresponding to the different rules. We have performed policy mining on that matrix using SVD then FCA. Both methods have output the same Net-RBAC configuration, without factorization error, mainly because it is relatively a small data set. In the resulting configuration we have four roles, seven activities, seven views, and seven abstract rules. We have observed

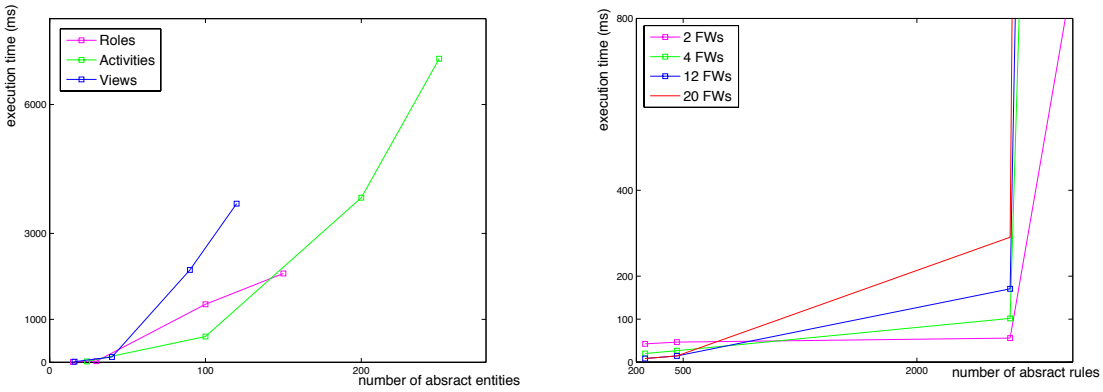
the set of concrete entities assigned to each abstract entity and named it accordingly. This could help to detect anomalies. For example if a source_host has nothing to do with a set of source_hosts assigned to the same role, this should draw the attention of the security administrator.

8.5 Multiple Firewalls Policy Mining Experiments

The experiments scenario in this section is: we generate a random Net-RBAC configuration, we divide it into multiple firewalls, and we perform policy mining on the set of concrete rules of each firewall. Then we run the algorithms 3 and 2 to obtain a network Net-RBAC configuration in a reverse engineering process. We have generated four data sets with increasing sizes and with the density parameters set to 0.1.

To simulate the enforcement of the Net-RBAC rules on multiple firewalls, we have devised an algorithm that takes the desired number of firewalls and the density of each firewall, then it picks randomly a subset of abstract rules from the Net-RBAC configuration to assign it to each firewall. For consistency, the algorithm ensures that the rules between a given role and a given view are assigned together to firewalls to simulate the firewall positioning in the topology. A relation that simulates the *routes* is generated from the repartition of network rules into firewalls. For each tuple of role and view, the relation*routes* supplies the set of firewalls between them. We have used the Singular Value Decomposition *SVD* algorithm to perform policy mining on the concrete rules of each firewall . Figure 8.4(a) shows that the execution time of Algorithm 2 tends to rises with the increasing number of abstract entities corresponding to each of the four Net-RBAC configurations divided into four firewalls. The execution time fluctuates slightly according to the hierarchy size of the abstract entities, but remains low.

For figure 8.4(b), we have subsequently divided each of the four data sets into 2, 4, 12 and then 20 firewalls. We have plotted the execution time of algorithm 3 in terms of the number of abstract rules. For the same Net-RBAC configuration, the execution time tends to rise with the number of firewalls because the average number of firewalls on a route increases and the number of comparisons for accessibility verification with it. Likewise, the execution time increases with the number of abstract entities. In the fourth data set, the number of abstract rules averages 21000, corresponding to around 180000 concrete rules, and the execution time takes few hours, whereas it takes only few seconds for data set 3 with 2598 abstract rules, corresponding to 11892 concrete rules.



(a) Performance of Abstract Entities Merger as Function of the Number of Abstract Entities in the Network

(b) Performance of Abstract Rules Merger as Function of Number of Abstract Rules in the Network

Figure 8.4: Experimental Results with Policy Merger Algorithms

8.6 Conclusion

We have implemented a proof of concept platform for the experimental study and the performance assessment of the contributions presented in this thesis. We have adapted some of the most used factorization techniques in role mining. We have developed synthetic data generators for RBAC and Net-RBAC configurations. We have also provided how to simulate network topology and routes in synthetic data. Finally, we have implemented all the proposed algorithms in the preceding chapters. The experimental results are consistent with the theoretical study. They confirm the validity of our approach of role set comparison as a beneficial solution to analyze and better understand the set of mined roles when an original set of roles exists. The experimental results also illustrate the applicability of the solution proposed for shadowed roles detection. Likewise, they illustrate the feasibility of the policy mining approach on a single and multiple firewalls.

The performance results could be enhanced with a more efficient implementation in an object oriented language, and with a larger computing capacity hardaware.

CHAPTER 9

Conclusion & Perspectives

Approaches presented in this thesis are part of a project that aims at finding practical solutions to address the network security management problems faced by security sensitive organizations such as banks. The information system is usually protected by firewalls that have been until now monitored and managed manually. This mode of functioning leads to many complications because of the growing number of configured firewalls and rules. Our approach aims at applying tailored mining techniques to the firewalls already configured, detecting misconfigurations, extracting a high abstraction level policy using an appropriate model, and managing the network access control policy from the abstract level in a repeatable cycle of bottom-up policy mining and top-down policy deployment.

In this view, we have first surveyed firewall management problems and different approaches proposed in literature to handle them in chapter 2. We have shown the limitation of the existing solutions that focus mainly on debugging and testing configured rules. We have noted the existence of efficient top-down model based approaches for the definitions of network access control policies, and the reliable automatic tools to deploy them onto firewalls. Then we have demonstrated that the Net-RBAC model is very suitable to network access control policy expression. Indeed, it considers the access control rules as a ternary relation, in the same way as firewall rules.

In chapter 3, we have surveyed the related work about role mining, the automatic bottom-up role engineering method for RBAC configuration from the deployed access control lists. We have reviewed the most important solutions, and have raised remaining issues to fully exploit this discipline in practice.

In Chapter 4 , we have defined a bottom-up approach for the Net-RBAC model configuration. We have shown the limits of existing role mining techniques to satisfy network security requirements. We have proposed a formalization of policy mining

problem, and an algorithm to solve it. We have suggested to handle a three-dimensional matrix factorization whereas the traditional role mining techniques are usually modeled as a two-dimensional matrix factorization. Our solution allows to extend and leverage existing mining tools to network security application. We have illustrated policy mining approach by applying it to a real data set.

In chapter 5 We have proposed a process of integration of Net-RBAC policies of multiple firewalls involved in the same network. This process aims at completing the bottom-up approach framework that allows to mine a network policy modeled with the Net-RBAC model from the rules enforced in several firewalls. In this perspective, we have shown that it is necessary to run policy mining on each firewall apart, and integrate the obtained policies only after. We have proposed a methodology based on unifying role, activity and view hierarchies of the network policy through defining a poset merging algorithm. Then, we have proposed a solution to build the network high level policy rules while verifying relevance and accessibility properties. We have illustrated the whole process with a realistic example.

Regarding the general applicability of role mining techniques, we have addressed two important issues related to role management. We have noted that security administrators still lack automated tools to assist them in tasks related to the evaluation and the management of RBAC roles, as well as Net-RBAC abstract entities. These issues become more important with the need to leverage outcomes of role mining and policy mining. Such tasks are very sensitive for the security of the organization system, and it is hard and risky to handle them manually.

First, we have focused on the comparison of two configurations of roles. In chapter 6, we have provided a formal approach to handle the problem of comparing two sets of roles. The solution supplied allows appraising and leveraging the learning outcomes of the role mining process by comparing the obtained mined roles with a reference set of roles which may be the set of original roles, or the set of permissions assigned to each user. 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. The algorithm proposed projects roles from one set into the other set. It maps the inherent relations among the sets based on algebraic expressions. Each role from the original roles is expressed 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 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 and negation of roles from the other set to represent accumulation of permissions, exceptions and also hierarchical relationships. This approach enables an administrator to measure how comparable the two configurations of roles are, to interpret and analyze each mined role, and to optimally and automatically assign the users to their corresponding roles. It also provides some further benefits such as the detection of unhandled perturbations or source misconfiguration. 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 evaluated our algorithm time complexity.

Second, in chapter 7 we have stressed the problem of detecting shadowed roles in a given configuration. Shadowing is usually due to a misconfiguration of roles. We have formally defined the shadowing cases. We have provided a solution to detect shadowed roles. And we have formally linked the problem of shadowed roles to the problem of comparing sets of roles. Indeed, in most cases related to role mining, preprocessing the compared sets of roles by eliminating shadowing from the set of roles significantly lowers the complexity of the RSCP solution.

All the aforementioned contributions are proved and then tested in a proof of concept platform. Chapter 8 summarizes all the experimental results. The experimental results 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 algorithm skills in assigning mined roles to users. We have also presented the experimental results that show the efficiency and the scalability of the shadowing detection solution. We have also shown the feasibility of our proposed approaches for network policy mining and multiple firewalls policies integration. We have measured the algorithms performances and studied their behavior in different practical situations such as approximate and exact factorizations. We have shown that the policy mining solution provides more compact policies and that the integration method permits to verify some important security properties so to detect inter-firewalls inconsistency problems in the deployed rules.

To conclude, this thesis has provided several fundamental results for role mining and, in particular, for policy mining in the network security application. Several examples, developed on both synthetic and real data, support our claims.

This work holds several interesting perspectives.

We plan to integrate the algorithms presented into RBAC and Net-RBAC administration tools.

Policy mining could be used for the analysis of other access control devices such as *Intrusion Detection Systems (IDS)*.

Policy mining could be used for examining firewall logs. Several role mining papers have suggested the collection and exploitation of the history logs of user activities to detect active, important, exceptional and misused authorizations, but never dealt with this possibility.

Currently, the bottom-up approach proposed for policy mining is applicable for stateless firewall rules. We consider using the concept of *context* defined in the OrBAC model as relevant to manage stateful firewalls. Policy integration solution also should be extended to augmented versions of the Net-RBAC to support stateful firewalls.

Moreover, in the preprocessing phase, we have transferred the set of rules to a set of accept only rules in a default close policy. From access control perspective, this does not affect the deployed policy. However, negative rules are implemented on firewalls for several practical reasons such as defense in depth and countering spoofing attacks. As a perspective, the preprocessing phase should also support the transformation of the rules into a reject only policy in a default open policy, in order to perform policy mining on the negative rules too and help the administrator to analyze its deployed reject rules as well.

Beyond the network security application, other access control applications may require the ternary rules modeling as proposed by the OrBAC model. This policy mining technique would offer a bottom-up approach for all these applications.

Bibliography

[1] Metasploit. *http://www.metasploit.com*. 9

[2] Cisco netsonar. *http://www.cisco.com/c/en/us/products/security/scanner/literature.html* (arrested development in 2005), 1999. 9

[3] Lumeta firewall analizer. *http://www.lumeta.com*, 2001. 13

[4] Muhammad Abedin, Syeda Nessa, Latifur Khan, Ehab Al-Shaer, and Mamoun Awad. Analysis of firewall policy rules using traffic mining techniques. *Int. J. Internet Protocol Technology*, 5(1-2), 2010. 13

[5] E.S. Al-Shaer and H.H. Hamed. Discovery of policy anomalies in distributed firewalls. In *23th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2004*, volume 4, pages 2605–2616, 2004. 11

[6] E.S. Al-Shaer and H.H. Hamed. Modeling and management of firewall policies. *Network and Service Management, IEEE Transactions on*, 1(1):2–10, 2004. 11, 61

[7] J. G. Alfaro, N. Boulahia-Cuppens, and F. Cuppens. Complete analysis of configuration rules to guarantee reliable network security policies. *Int. J. Inf. Secur., Springer-Verlag, Berlin, Heidelberg*, 7(2):1615–5262, 2008. 11

[8] Jérôme Amilhastre, Philippe Janssen, and Marie-Catherine Vilarem. Complexity of minimum biclique cover and minimum biclique decomposition for bipartite domino-free graphs. *Discrete Applied Mathematics*, 86(2-3):125 – 144, September 1998. 36

[9] Gabriela Arévalo, Anne Berry, Marianne Huchard, Guillaume Perrot, and Alain Sigayret. Performances of galois sub-hierarchy-building algorithms. In *Proceedings of the 5th international conference on Formal concept analysis ICFCA'07*, pages 166–180. Springer-Verlag, 2007. 34

[10] Peter Chen, Guoli Ding, and Steve Seiden. On poset merging. Technical report, Louisiana State University, Dpt of Computer Science & Dpt of Mathematics, 2003. 57

[11] Alessandro Colantonio. *Role Mining Techniques To Improve RBAC Administration.* PhD thesis, Università Degli Studi Roma Tre, Facoltà di Scienze Matematiche Fisiche e Naturali, XXIII Ciclo del Dottorato di Ricerca in Matematica, 2011. 39

[12] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A formal framework to elicit roles with business meaning in *RBAC* systems. In *Proceedings of the 14th ACM symposium on Access control models and technologies SACMAT '09*, pages 85–94. ACM, June 2009. 28

[13] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. Leveraging lattices to improve role mining. In *Proceedings of the 23rd International Information Security Conference IFIP SEC'08*, pages 333–347. Springer, September 2008. 35, 101

[14] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Mining stable roles in *RBAC*. In *Proceedings of the 24th International Information Security Conference IFIP SEC'09*, pages 259–269. Springer, 2009. 77

[15] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A new role mining framework to elicit business roles and to mitigate enterprise risk. *Decision Support Systems*, 50:715–731, 2011. Special Issue on "Enterprise Risk and Security Management: Data, Text and Web Mining". 28, 29, 37

[16] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Visual role mining: A picture is worth a thousand roles. *IEEE Transactions on Knowledge and Data Engineering*, 2011. 37, 39

[17] Frédéric Cuppens, Nora Cuppens-Boulahia, and Joaquin Garcia. Misconfiguration management of network security components. In *IASTED International Conference on Communication, Network,and Information Security (CNIS 2005)*, Phoenix, USA, November 2005. 11

[18] Frédéric Cuppens, Nora Cuppens-Boulahia, Thierry Sans, and Alexandre Miège. A formal approach to specify and deploy a network security policy. In Theodosis Dimitrakos and Fabio Martinelli, editors, *Formal Aspects in Security and Trust*, pages 203–218. Springer, 2004. 1, 3, 15, 21

[19] Nora Cuppens-Boulahia and Frédéric Cuppens. Gestion des identités et des autorisations. *MISC: Multisystem & Internet, Security, Cookbook*, 2012. 40

[20] D. Brent Chapman Elizabeth D. Zwicky, Simon Cooper. *Building Internet Firewalls, 2nd Edition*. O'Reilly Media, june 2000. 66

[21] Alina Ene, William Horne, Nikola Milosavljevic, Prasad Rao, Robert Schreiber, and Robert E. Tarjan. Fast exact and heuristic methods for role minimization problems. In *Proceedings of the 13th ACM symposium on Access control models and technologies SACMAT'08*, pages 1–10. ACM, June 2008. 30, 36, 43

[22] Dan Farmer and Wietse Venema. Improving the security of your site by breaking into it. 9

[23] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed *NIST* standard for role-based access control. standard, NIST, 2001. 2, 19

[24] Mario Frank, Joachim M. Buhmann, and David Basin. On the definition of role mining. In *Proceeding of the 15th ACM symposium on Access control models and technologies SACMAT '10*, pages 35–44. ACM, June 2010. 24, 32, 44

[25] Mario Frank, Andreas P. Streich, David A. Basin, and Joachim M. Buhmann. A probabilistic approach to hybrid role mining. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proceedings of the 16th ACM Conference on Computer and Communications Security CCS'09*, pages 101–111. ACM, November 2009. 28, 29, 35

[26] Martin Freiss. *Protecting Networks with SATAN*. O Reilly Media, May 1998. 9

[27] Ludwig Fuchs and Stefan Meier. The role mining process model - underlining the need for a comprehensive research perspective. In *In proceedings of the 6th International Conference on Availability Reliability and Security ARES'11*, pages 35–42. IEEE, 2011. 4

[28] Ludwig Fuchs, Günther Pernul, and Ravi S. Sandhu. Roles in information security - a survey and classification of the research area. *Computers & Security*, 30(8):748–769, August 2011. 3, 20, 28

[29] Michael P. Gallaher, Alan C. O'Connor, and Brian Kropp. The economic impact of role-based access control. Technical report, RIT for NIST (National Institute of Standards and Technology), March 2002. 2

[30] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999. 34, 101

[31] R. L. Goodstein. *"Boolean Algebra"*. Dover Publications, 2007. 91

[32] Safaà Hachana, Nora Cuppens-Boulahia, and Frédéric Cuppens. Role mining to assist authorization governance: How far have we gone? *International Journal of Secure Software Engineering (IJSSE)*, 3(4), 2012. 2, 4

[33] Duan Haixin, Wu Jianping, and Li Xing. Policy based access control framework for large networks. In *Proceedings of IEEE International Conference on Networks, ICON 2000*, pages 267–272, 2000. 14, 15

[34] Anas Abou El Kalam, Salem Benferhat, Alexandre Miege, Rania El Baida, Frédéric Cuppens, Claire Saurel, Philippe Balbiani, Yves Deswarte, and Gilles Trouessin. Organization based access control. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), 4-6 June 2003, Lake Como, Italy*, number 0-7695-1933-4, pages 120 –. IEEE Computer Society, 2003. 15

[35] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the 21st national conference on Artificial intelligence*, volume 1, pages 381–388, 2006. 46

[36] David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni. *Metasploit : The Penetration Tester's Guide*. William Pollock, San Francisco, no strach press edition, 2011. 9

[37] Virginia C . Klema and Alan J. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25(2):164–176, April 1980. 30, 33, 100

[38] Mehmet Koyuturk, Aaanth Grama, and Naren Ramakrishnan. Nonorthogonal decomposition of binary matrices for bounded-error data compression and analysis. *ACM Transactions on Mathematical Software*, 32(1):33–69, March 2006. 33, 100

[39] Petr Krajca, Jan Outrata, and Vilem Vychodil. Parallel recursive algorithm for FCA. In *Proceedings of the Sixth International Conference on Concept Lattices and Their Applications*, volume 433, pages 71–82, 2008. 101

[40] Martin Kuhlmann, Dalia Shohat, and Gerhard 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*, pages 179–186. ACM, 2003. 2, 76

[41] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Neural Information Processing Systems (NIPS)*, pages 556–562, 2000. 100

[42] Haibing Lu, Jaideep Vaidya, and Vijayalakshmi Atluri. Optimal boolean matrix decomposition - application to role engineering. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008*, pages 297–306. IEEE, April 2008. 32

[43] Xiaopu Ma, Ruixuan Li, and Zhengding Lu. Role mining based on weights. In *SACMAT '10: Proceeding of the 15th ACM symposium on Access control models and technologies*, June 2010. 28

[44] Richard Macfarlane, William J. Buchanan, Elias Ekonomou, Omair Uthmani, Lu Fan, and Owen Lo. Review of security policy implementations. *Computers & Security (COMPSEC)*, 2(31):253–270, 2011. 1

[45] Joachim M. Buhmann Mario Frank, David Basin. A class of probabilistic models for role engineering. In *Proceedings of the 15th ACM Computer and Communications Security Conference CCS'08*, pages 27–31. ACM, October 2008. 35

[46] Robert Marmorstein and Phil Kearns. A tool for automated iptables firewall analysis. In *USENIX annual technical conference*, 2005. 10

[47] Robert M. Marmorstein and Phil Kearns. Firewall analysis with policy-based host classification. In *Proceedings of the 20th conference on Large Installation System Administration LISA'06*, pages 41–51, Berkeley, CA, USA, December 2006. USENIX Association. 12, 13

[48] Alain Mayer, Avishai Wool, and Elisha Ziskind. Fang: A firewall analysis engine. IEEE, 2000. 10

[49] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. Mining roles with semantic meanings. In *Proceedings of the 13th ACM symposium on Access control models and technologies SACMAT '08*, pages 21–30. ACM, June 2008. 28, 30, 34

[50] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. Mining roles with multiple objectives. *ACM*

Transactions on Information and System Security (TISSEC), 13:36:1–36:35, December 2010. 28, 101

[51] Ian Molloy, Ninghui Li, Tiancheng Li, Ziqing Mao, Qihua Wang, and Jorge Lobo. Evaluating role mining algorithms. In *Proceedings of the 14th ACM symposium on Access control models and technologies SACMAT '09*, pages 95–104. ACM, June 2009. 4, 34, 37, 38, 76

[52] Ian Molloy, Ninghui Li, Yuan (Alan) Qi, Jorge Lobo, and Luke Dickens. Mining roles with noisy data. In *Proceedings of the 15th ACM symposium on Access control models and technologies SACMAT '10*, pages 45–54. ACM, June 2010. 29, 30, 38, 77

[53] Ralf Nikolaus. Learning the parts of objects using non-negative matrix factorization (nmf), February 2007. 33

[54] Stere Preda, Nora Cuppens-Boulahia, Frédéric Cuppens, Joaquín García-Alfaro, and Laurent Toutain. Model-driven security policy deployment: Property oriented approach. In Fabio Massacci, Dan S. Wallach, and Nicola Zannone, editors, *International Symposium on Engineering Secure Software and Systems ESSoS 2010*, volume 5965, pages 123–139. Springer, February 2010. 1, 16

[55] Stere Preda, Nora Cuppens-Boulahia, Frédéric Cuppens, and Laurent Toutain. Architecture-aware adaptive deployment of contextual security policies. In *ARES 2010 : Fifth International Conference on Availability, Reliability and Security*, Krakow, Poland, february 2010. 1, 16

[56] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control model. *IEEE Computer*, 29(2):38–47, February 1996. 19

[57] Basit Shafiq, James Joshi, Elisa Bertino, and Arif Ghafoor. Secure interoperation in a multidomain environment employing rbac policies. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1557–1577, 2005. 56

[58] Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. Technical report, School of Computer Science Carnegie Mellon University Pittsburgh, June 2008. 33

[59] 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*, pages 969–976. ACM, 2009. 35, 77

[60] Hassan Takabi and James B.D. 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*, pages 55–64. ACM, June 2010. 76

[61] Romuald Thion. Découverte automatisée de hiérarchies de rôles pour les politiques de contrôle d'accès. In *INFORSID*, pages 139–154, Mai 2007. 34

[62] A. Titov and V. Zaborovsky. Firewall configuration based on specifications of access policy and network environment. 2010. 1, 16

[63] Alok Tongaonkar, Niranjan Inamdar, and R. Sekar. Inferring higher level policies from firewall rules. In *Proceedings of the 21st Large Installation System Administration Conference LISA '07*, November 2007. 13, 50

[64] Alok S. Tongaonkar. Fast pattern-matching techniques for packet filtering. The graduate school in partial fulfillment of the requirements for the degree of master of science in computer science, Stony Brook University, May 2004. 45

[65] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi 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*, pages 175–184. ACM, June 2007. 31, 32, 36

[66] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Nabil 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*, pages 11–20. ACM, June 2008. 32, 76

[67] Jaideep Vaidya, Vijayalakshmi Atluri, and Janice Warner. Roleminer: mining roles using subset enumeration. In *Proceedings of the 13th ACM Conference on Computer and Communications Security CCS'06*, pages 144–153. ACM, November 2006. 38, 50, 76

[68] Avishai Wool. Architecting the lumeta firewall analyzer. In *Proceedings of the 10th Conference on USENIX Security Symposium*, volume 10 of *SSYM'01*, pages 7–19, Berkeley, CA, USA, 2001. USENIX Association. 13

[69] Lihua Yuan and Hao Chen. Fireman: A toolkit for firewall modeling and analysis. In *In Proceedings of IEEE Symposium on Security and Privacy*, pages 199–213, 2006. 12

[70] Vladimir S. Zaborovsky and Anton Titov. Specialized solutions for improvement of firewall performance and conformity to security policy. In *Security and Management*, pages 603–608, 2009. 1, 16

[71] Bin Zhang, Ehab Al-Shaer, Radha Jagadeesan, James Riely, and Corin Pitcher. Specifications of a high-level conflict-free firewall policy language for multi-domain networks. In *Proceedings of the 12th ACM symposium on Access control models and technologies SACMAT '07*, pages 185 – 194, New York, NY, USA, 2007. ACM. 15

[72] Dana Zhang, Kotagiri Ramamohanarao, and Tim Ebringer. Role engineering using graph optimisation. In *Proceedings of the 12th ACM symposium on Access control models and technologies SACMAT '07*, pages 139–144. ACM, June 2007. 32, 36

List of Publications Related to this Thesis

Journal Papers

- Safaà Hachana and Nora Cuppens-Boulahia and Frédéric Cuppens, *Mining a High Level Access Control Policy in a Network with Multiple Firewalls*, Journal of Information Security and Applications (JISA), 2014 (Submitted – Under Revision).
- Safaà Hachana and Frédéric Cuppens and Nora Cuppens-Boulahia and Joaquin Garcia-Alfaro, *Semantic Analysis of Role Mining Results and Shadowed Roles Detection*, Information Security Technical Report (ISTR), ARES 2012 Special Issue, 2013. Elsevier.
- Safaà Hachana and Nora Cuppens-Boulahia and Frédéric Cuppens, *Role Mining to Assist Authorization Governance: How Far Have We Gone?*. International Journal of Secure Software Engineering (IJSSE) 3 (2012). IGI Global.

International Conferences

- Safaà Hachana and Frédéric Cuppens and Nora Cuppens-Boulahia and Vijay Atluri and Stephane Morucci, *Policy Mining: a Bottom-Up Approach Toward a Model Based Firewall Management*, In proceedings of the 9th International Conference on Information Systems Security (ICISS 2013). Springer Verlag.
- Safaà Hachana and Frédéric Cuppens and Nora Cuppens-Boulahia and Joaquin Garcia-Alfaro, *Towards automated assistance for mined roles analysis in role mining applications*, In Proceedings of the 7th International Conference on Availability, Reliability and Security ARES 2012, p.123–132. IEEE. (Best Nominated Paper Award).

ANNEXE 1

FOUILLES DE RÈGLES DE POLITIQUES DE SÉCURITÉ : UNE APPROCHE ASCENDANTE POUR L'ADMINISTRATION DE LA SÉCURITÉ RÉSEAU

[1]

[1]Résumé étendu de la thèse en français.

1.1 Chapitre 1 : Introduction

Les Systèmes d'Information (SI) se sont diffusés massivement dans les organisations contemporaines pour soutenir les processus de gestion, de production et de commercialisation. Dans ce contexte, le contrôle d'accès aux réseaux informatiques, assuré essentiellement par des pare-feu, revêt une importance capitale. Cependant, la gestion manuelle des pare-feu s'avère une tâche complexe, coûteuse et sujette à l'erreur.

L'utilisation de modèles à haut niveau d'abstraction comme le modèle *RBAC (Role Based Access Control)* a prouvé son efficacité dans la définition et la gestion des politiques de contrôle d'accès. L'adoption de RBAC a bénéficié de l'intérêt porté plus récemment à la fouille de rôles ascendante pour une configuration automatique du modèle à partir des autorisations préalablement déployées. Cette discipline est communément appelée *role mining*.

1.1.1 Contribution

Cette thèse est consacrée à une approche ascendante pour l'administration de la sécurité des réseaux informatiques d'un haut niveau d'abstraction avec l'assurance d'un coût bas et d'un niveau de confiance élevé. Nous montrons que le modèle *Net-RBAC (Network Role Based Access Control)* est plus adapté à la spécification des politiques de contrôle d'accès des réseaux que le modèle RBAC.

Nous proposons une approche baptisée *policy mining* qui extrait de manière ascendante et automatique la politique de contrôle d'accès modélisée par Net-RBAC à partir des règles de sécurité déployées sur un pare-feu. Nous définissons un algorithme basé sur la factorisation matricielle, capable d'adapter la plupart des techniques de *role mining* existantes afin d'extraire des instances du modèle Net-RBAC.

En plus, comme les réseaux des entreprises sont souvent protégés par plusieurs pare-feu, dans notre quête d'une solution complète pour la fouille automatique de la politique de contrôle d'accès du réseau entier, nous traitons le problème d'intégration de politiques modélisées par Net-RBAC et résultant de l'application de la technique de *policy mining* sur plusieurs pare-feu chacun à part. Par la même occasion, nous vérifions les propriétés de sécurité reliées à la cohérence du déploiement de la politique sur plusieurs dispositifs.

En outre, nous avons noté pendant notre étude des axes de recherche autour du *role mining* un manque de base claire quant'à l'exploitation des résultats émanant du processus ascendant. Nous proposons dans cette thèse des outils destinés à assister un

administrateur de sécurité réseau dans l'analyse des résultats aussi bien du *role mining* que du *policy mining*. Nous proposons une formalisation du problème de comparaison de deux configurations de rôles et prouvons que le problème est NP-Complet. Nous définissons un algorithme qui identifie des relations pertinentes entre les rôles respectifs de deux configurations équivalentes, et fournit la projection des rôles d'une configuration dans une autre, en se basant sur des expressions Booléennes. Cette approche est appropriée pour mesurer la comparabilité de deux ensembles de rôles et aussi pour interpréter chaque rôle. Nous étudions d'autres questions connexes comme la détection des perturbations de configuration à la source.

En particulier, nous soulignons que la présence de rôles masqués dans une configuration de rôles se manifeste par une augmentation de la complexité de la comparaison avec une autre configuration. Nous présentons une solution pour détecter différentes anomalies d'ombrage dans une configuration de rôles.

Chacune des contributions sus-mentionnées se base sur un cadre théorique solide, est illustrée par des exemples réels, et est appuyée par des résultats expérimentaux.

1.1.2 Plan de la thèse

Cette thèse s'organise comme suit :

Chapitre 2 revoie et classifie les approches qui traitent les problèmes reliés à l'administration des pare-feux dans la littérature et montrent leurs limites. Ensuite, il motive pour l'adoption d'un cycle basé sur deux approches descendante et ascendante complémentaires qui font le lien entre le niveau bas de déploiement sur les pare-feux, et un haut niveau d'abstraction pour la définition et l'administration des politiques de sécurité réseau. Le chapitre introduit le modèle Net-RBAC. Il montre les limites de RBAC dans l'application aux réseaux informatiques et montre comment Net-RBAC peut remédier à ces limites.

Chapitre 3 est une analyse de l'état de l'art du domaine de role mining. Il rappelle les définitions formelles du problème de role mining, résume les solutions les plus marquantes classées par leurs domaines de référence théoriques. Il couvre les problématiques résolues dans la littérature et souligne les problématiques pas encore traitées.

Chapitre 4 présente policy mining, une extension de role mining qui s'applique sur des règles de pare-feu afin d'en extraire la politique haut niveau correspondante, formulée dans le modèle Net-RBAC. Conçue pour être flexible, policy mining pourrait étendre toute solution de role mining existante pour supporter le modèle cible Net-RBAC.

Chapitre 5 complète l'approche ascendante pour la gestion de la sécurité des réseaux en montrant comment intégrer les résultats de policy mining effectué sur plusieurs parefeux protégeant le même réseau. L'objectif de cette étape est d'extraire la politique haut niveau déployée dans le réseau tout en vérifiant des propriétés de sécurité pertinentes.

Chapitre 6 a pour objectif de fournir des outils pour assister un administrateur de sécurité dans la tâche d'exploitation de résultats de role mining ou de policy mining. Il formalise et résous le problème de comparaison de deux configurations de rôles.

Chapitre 7 porte sur une autre problématique reliée à l'analyse des résultats du role mining, notamment le problème des role masqués. Ce chapitre relie ce problème avec le problème de comparaison de configuration de rôles, et propose une solution pour le résoudre.

Chapitre 8 décrit la plateforme de test implémentée et présente les résultats expérimentaux d'évaluation des solutions proposées dans les quatre chapitres précédents. Les résultats montrent la faisabilité de l'approche ascendante de policy mining pour une administration de la sécurité réseau dirigée par le modèle Net-RBAC. Les solutions de comparaison de configurations de rôles et de détection de rôles masqués sont aussi évalués.

Chapitre 9 conclue la thèse et discute quelques perspectives.

1.2 Chapitre 2 : Vers une gestion du contrôle d'accès aux réseaux dirigée par des modèles

1.2.1 Approches existantes dans la littérature pour la gestion des pare-feux

Alors que les constructeurs de pare-feux se sont focalisés sur l'amélioration de performance en terme de vitesse, capacité de filtrage, et transparence par rapport aux services, les problématiques de gestion et de configuration des pare-feux ont demeuré une affaire académique. Cette section décrit les différentes classes d'approches qui traitent les problèmes de gestion de pare-feux dans la littérature :

- **Approaches intrusives.** La majorité des articles de recherche orientés pratique fournissent des méthodologies d'analyse des politiques de contrôle d'accès réseau par des *tests de pénétration*. Cette méthode intrusive consiste à jouer le role d'un attaquant potentiel et effectuer des tests actifs en injectant des paquets dans le réseau et scannant les ports pour observer le comportement des pare-feux et évaluer la robustesse des règles qui y sont implémentées. Il existe dans ce domaine des solutions logicielles (e.g.*SATAN* (software penetration tests tool) [10, 14]) et des solutions matérielles (e.g. *NetSonar* [1] from Cisco). Cette approche est difficile à mener. Elle nécessite souvent le recours aux services de consultants externes. En plus les scénarios d'attaques envisagés sont rarement exhaustifs, et les résultats restent statistiques. Le caractère intrusif peut de sursoit avoir un effet nocif sur le fonctionnement du réseau. Avec cette approche, la validité de la modélisation du réseau et de la politique reste contestable. De même, la formulation des requêtes présente une difficulté importante.

- **Outils de simulation pour requêtes hors ligne.** Ces outils visent à faciliter la compréhension de la configuration de règles déployées dans un pare-feux ou un système de pare-feux par simulation. Ils prennent comme input les règles des pare-feux, ainsi qu'une description de la topologie. Ils construisent un modèle de simulation capable de répondre à des question comme [*est ce que le point a peut envoyer le service s au point b ?*]. *FANG* [22] est l'un des outils de modélisation et simulation les plus importants.

- **Détection d'anomalies de configuration par analyse structurelle des règles.** Cette approche permet de découvrir les erreurs de configuration structurelles dans un pare-feu ou un système de pare-feux. Elle se base sur la comparaison et l'analyse des relations entre les règles de filtrage. Elle peut détecter les

règles conflictuelles et les règles inutiles car masquées dans une configuration de règles. Souvent, l'administrateur est notifié des erreurs potentielles détectées, et décide de manière interactive si ces erreurs sont pertinentes, ou bien si elles sont de faux positifs. Ensuite, l'administrateur réécrit les règles de manière équivalente à la configuration initiale après avoir éliminé les erreurs structurelles. Plusieurs solutions d'analyse structurelles sont proposées dans la littérature, comme Mirage (MIsconfiguRAtion manaGEment of network security components) [3, 6] et FIREMAN (FIREwall Modeling and ANalysis) [34]. L'analyse structurelle ne permet pas de comprendre le comportement des pare-feux. En plus, la définition des anomalies à détecter est un problème sensible et devrait éviter trop de faux positifs. Cette approche demande beaucoup de comparaisons de règles, ce qui soulève un problème de scalabilité. Finalement, la gestion des notifications d'erreur se fait au niveau du langage d'implémentation du pare-feu.

- **Aggrégation et refactoring des règles.** Une autre approche essaie de réécrire l'ensemble e règles déployées dans un pare-feu de manière plus compacte et optimisée afin d'améliorer leur lisibilité. Dans la plupart des cas, cette réécriture implique l'agrégation des règles. Par exemple, dans [21], Marmorstein et al. ont proposé une solution basée sur la classification des hôtes_sources et hôtes_destination dans des groupes homogènes. L'objectif est de rendre l'identification des anomalies plus facile. Dans [30], Tongaonkar et al. ont proposé l'agrégation des services aussi. Abedin et al. [2] ont appliqué des techniques de data mining sur les lots des pare-feux pour extraire les règles de filtrage effectives exécutées. L'approche de réécriture et d'agrégation offre des solutions automatiques et améliore souvent la lisibilité des configuration des pare-feux. Ceci peut représenter une aide estimable pour les administrateurs. Cependant, les solutions proposées dans la littérature ne visent pas un modèle haut niveau bien défini, et peuvent donc fournir des résultats trop peu structurées pour être exploitables.

- **Administration de la politique de sécurité à un haut niveau d'abstraction.** La plupart des problèmes liés à l'administration des pare-feux viennent de la difficulté de la manipulation des langages bas niveaux de ceux-ci. Plusieurs approches ont proposé de définir la politique de contrôle d'accès à un haut niveau d'abstraction pour la déployer ensuite via des outils de traduction automatiques sur les pare-feux. Cuppens et al. [7] ont fourni une approche formelle pour spécifier et déployer une politique de sécurité réseau d'un haut niveau d'abstraction. Ils ont utilisé un langage de control d'accès basé sur une syntaxe XML dont les sémantiques sont interprétées dans le modèle *Organization Based Access Control (OrBAC)* [16]. Preda et al. [26, 27] ont établi un cadre formel pour la validation

du déploiement automatique d'une politique de contrôle d'accès formulée avec OrBAC. Ils ont développé des algorithmes qui réalisent la traduction des règles de sécurité OrBAC vers le niveau de déploiement, en introduisant les informations sur la topologie et le routage. Chaque transition dans ce processus est validée par la méthode B.

Le recours à un niveau d'abstraction plus élevé pour la définition et la gestion des politiques de contrôle d'accès réseau permet à l'administrateur d'avoir le recul nécessaire afin de garder le contrôle. Les approches basées sur OrBAC offrent des sémantiques claires pour la définition de politique de sécurité réseau. En plus, ils permettent l'intégration directe de la politique de contrôle d'accès du réseau informatique dans la politique de contrôle d'accès de l'organisation hôte. Néanmoins, l'adoption d'une tel modèle à haut niveau d'abstraction pourrait impliquer de mettre de côté les règles déjà déployées, et de recommencer la spécifications de la politique de nouveau, dans une approche descendante pure.

1.2.2 Architecture descendante et ascendante pour une administration de la politique dirigée par un modèle

L'adoption d'un modèle à haut niveau d'abstraction semble être la meilleure alternative pour la gestion des politique de sécurité réseau. Cependant, le plus grand obstacle vers une telle démarche reste l'absence de solution d'intégration des règles de sécurité déjà déployées dans les pare-feux. Afin de surmonter ce problème, nous promouvons une approche ascendante qui analyse de manière automatique les règles configurées sur les pare-feux, et exploite des techniques de data mining pour atteindre une instance de la politique à un haut niveau d'abstraction.

Le modèle Net-RBAC

Nous proposons le modèle *Network-Role Based Access Control (Net-RBAC)*, une extension du modèle RBAC [11, 28] , pour exprimer les politiques de sécurité réseau au niveau abstrait.

Definition 1. *Network Role Based Access Control : Net-RBAC*
Une configuration Net-RBAC notée $NSRC = (ROLES, UR, ACTIVITIES, OPA, VIEWS, OBV, RAV)$ *est caractérisée par :*

- *U, ROLES, OPS, ACTIVITIES, OBJ, et VIEWS qui sony les ensembles d'utilisateurs, roles, opérations, activités, objets et vues*

- $UR \subseteq U \times ROLES$, *une correspondance plusieurs à plusieurs de la relation d'affectation utilisateurs-roles*
- $assigned\_users(R) = \{u \in U | (u, R) \in UR\}$, *la correspondance du role R à un ensemble d'utilisateurs*
- $OPA \subseteq OPS \times ACTIVITIES$, *une correspondance plusieurs à plusieurs de la relation d'affectation opération-activité*
- $assigned\_operations(A) = \{op \in OPS | (op, A) \in AD\}$, *la correspondance de l'activité A à un ensemble d'opérations*
- $OBV \subseteq OBJ \times VIEWS$, *une correspondance plusieurs à plusieurs de la relation d'affectation objet-vue*
- $assigned\_objects(V) = \{obj \in OBJ | (obj, V) \in OBV\}$, *la correspondance de la vue V à un ensemble d'objets*
- $RAV \subseteq ROLES \times ACTIVITIES \times VIEWS$, *une correspondance plusieurs à plusieurs de la relation d'affectation role-activité-vue.*

La définition 1 structure les utilisateurs en rôles, les opérations en activités et les objets en vues. Cette opération permet d'obtenir des règles abstraites de la relation *RAV*. Une règle abstraite prend la forme [*le role r set autorisé à efféctuer l'activité a sur la vue v*]. Une règle concrète impliquant un utilisateur u, un objet *ob* et une opération *op* existe si u est attribué à un rôles R, *op* à une activité A, et *ob* à une vue V, et le triplet (R, A, V) appartient à la relation *RAV*.

1.2.3 Conclusion et problématiques non traitées

Dans ce chapitre, nous avons étudié les approches existantes qui traitent les problèmes de gestion de pare-feux. Ces approches visent à analyser le comportement des pare-feux déjà configurées , mais sont souvent difficiles à mener. Nous avons recommandé une approche qui allie un processus ascendant et un processus descendant de manière cyclique, et qui est basé sur le modèle Net-RBAC. Des travaux intéressants ont été réalisés pour l'approche descendante dans la littérature. Cependant, l'approche ascendante reste encore à explorer. Dans cette thèse, nous traitons cette problématique. Nous sommes motivés par les exploits récents dans le domaine de role mining pour la configuration ascendante du modèle RBAC.

1.3 Chapitre 3 : Role Mining

1.3.1 Aperçu global du processus de role mining

Role mining [12] est la discipline qui définit les rôles de manière automatique à partir des règles de contrôle d'accès déployées dans un systèmes, dans le but de configurer une politique modélisée par RBAC. En partant d'un ensemble d'utilisateurs, de permissions, et de la relation d'affectation directe utilisateur-permission *UPA*, le processus de role mining calcule un ensemble de rôles, et l'affectation compatible des utilisateurs et des permissions à ces rôles. Cette affectation doit se faire en respect à la relation *UPA* initiale, et doit répondre à un nombre de principes fondamentaux : la sécurité, la disponibilité, la minimalité, l'interprétabilité, la généralisation.

Les matrices Booléennes sont généralement utilisées pour représenter les données manipulées par le processus de role mining. Étant donné m utilisateurs, n permissions et k rôles, la relation utilisateur-role *UR* peut être représentée par une matrice Boulénne de taille $k \times n$, avec 1 dans la case $\{ij\}$ indiquant que le rôle j est affecté à l'utilisateur i. De même, la matrice *RP* représente l'affectation des permissions aux rôles. La relation d'affectation directe utilisateur-permission est notée *UPA*. Dans une configuration RBAC, nous avons généralement : $UPA = UR \otimes RP$. Le processus de role mining a pour objectif d'extraire les matrices *UR* et *RP* à partir de *UPA*.

Pour illustrer, nous imaginons l'exemple dans la figure 1.1, inspiré des personnages de *The Simpsons*. Dans cet exemple, la matrice initiale *UPA* est représentée par une matrice Booléenne de 6 utilisateurs $\times$ 9 permissions. Nous proposons une décomposition possible de cette matrice avec 5 rôles.

Le fruit de plusieurs années de recherche dans le domaine de role mining et les efforts réalisés pour appliquer les techniques de rôles mining dans la pratique, avec une ambition de satisfaire les besoins de différents contextes, a fait surgir de nouvelles problématiques. La complexité n'est plus concentrée dans le calcul des rôles seulement, mais s'étend sur d'autres phases comme le pré-traitement des listes de contrôle d'accès fournies en input, le traitement des rôles obtenus, et l'influence du calcul des rôles pour mieux répondre aux attentes de l'application en particulier. Dans ce chapitre, nous étudions le processus de role mining étendu illustrée dans la figure 1.2.

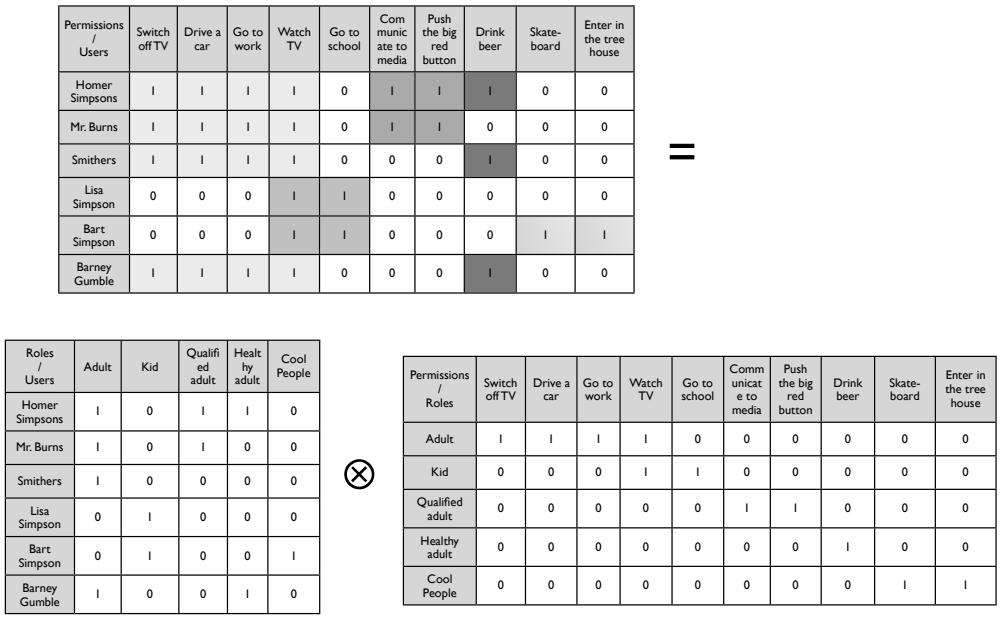


Figure 1.1 – Exemple de la représentation algébrique d'une relation de contrôle d'accès

1.3.2 Pré-traitement des données

Les données utilisées dans le processus du rôles mining sont principalement les listes de contrôle d'accès, et éventuellement des attributs relatifs aux utilisateurs ou aux permissions, et qui sont susceptibles d'améliorer la qualité des rôles obtenus. Un premier problème qui se pose est la sélection des attributs utiles dans le calcul des rôles. Frank et al. [13] proposent une méthode basée sur l'entropie pour sélectionner quels attributs seront bénéfiques pour le calcul des rôles. Le second problème relatif aux données en input est le problème des erreurs et/ou des exceptions dans la relation *UPA* initiale. Certaines techniques de role mining comme [9], essaient de contourner ce problème en autorisant une marge d'erreur dans le calcul des rôles. Molloy et al. [24] ont montré qu'il est plus intéressant d'éliminer les erreurs et les exception avant de calculer les rôles. Ce bruit se présente comme des motifs faibles dans la matrice *UPA*.

1.3.3 Extraction des roles

Le problème de role mining se formalise de plusieurs manières différentes selon les objectifs d'optimisation souhaités. Ces objectifs peuvent être la minimisation du nombre

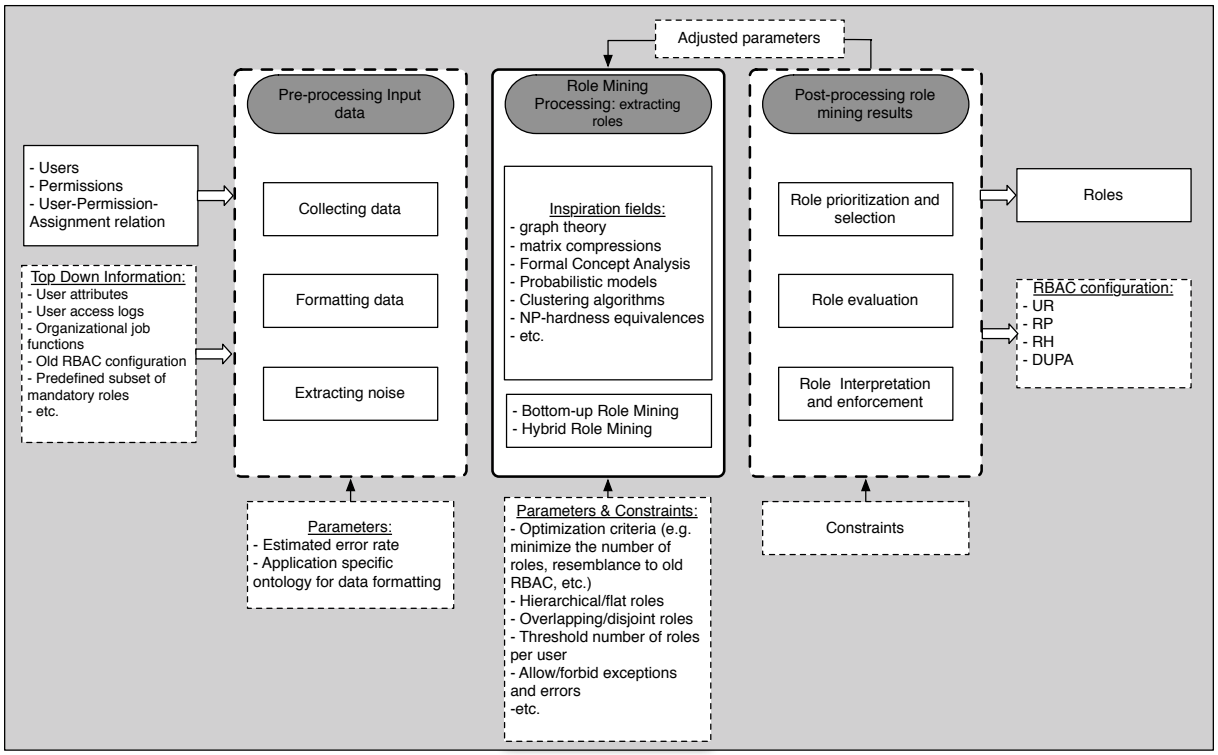


Figure 1.2 – Le processus étendu de role mining

de rôles obtenus [32], la minimisation du nombre d'affections [35], la maximisation du rapprochement avec un ensemble de rôles préexistants [33], etc.

Les techniques de role mining proposées dans la littérature sont majoritairement des adaptation de techniques déjà utilisés dans le data mining. Plusieurs domaines sont explorés comme :

- la compression des matrices : les rôles sont assimilés aux motifs récurrent dans la matrice *UPA*, et des méthodes de compression comme *Singular Value Decomposition (SVD)* [17], *Binary Non Orthogonal Factorization (BNOF)* [18], ou *Non Negative Matrix Factorization (NNMF)* [25] permettent de mettre en évidence ces motifs.

- l'analyse des concepts formels : les utilisateurs sont assimilés aux objets et les permissions aux attributs. Le calcul du treillis de concepts formels représente tous les rôles possibles. Une étape d'élagage est nécessaire par la suite pour sélectionner les rôles les plus pertinents selon les objectifs d'optimisation (ex. HierarchicalMiner [23], RM-SHG [29]).

- les modèles probabilistes : Frank et al. [20] décrivent un modèle probabiliste Bayesien qui remplace les valeurs dans UR et RP par des probabilités entre 0 et 1 afin d'améliorer la probabilité de UPA sachant la configuration de role présumée : $(P(UPA|RBAC))$.

- la théorie des graphes : Ene et al. [9] et Zhang et al. [35] utilisent des solutions connues d'optimisation de graphes pour calculer les rôles en modélisant les utilisateurs, permissions et rôles par des noeuds et les affectation par les liens.

- l'équivalence des classes de problème dans la théorie de la complexité : la plupart des définitions formelles du problème de role mining donnent des problèmes NP-Compelts. Ces problèmes peuvent être résolus par des algorithmes heuristiques définis pour résoudre des problèmes de classe de complexité équivalente. Par exemple, dans [32], le problème de role mining avec objectif de minimisation du nombre de rôles est réduit au *Minimum database Tiling problem*.

1.3.4 Évaluation et exploitation des rôles

L'évaluation des algorithmes de role mining couvre des paramètre d'ordre général comme le temps d'exécution, et la scalibilité. Le nombre d'erreurs dans la matrice des règles de contrôle d'accès obtenue par les rôles résultant par rapport à la liste de contrôle d'accès initiale ($||UPA - UR \otimes RP||$) est aussi un paramètre intéressant. Cependant, l'évaluation des rôles obtenus reste principalement un jugement subjectif qui revient à l'administrateur de sécurité. Il doit décider si les rôles obtenus sont instinctifs et naturels dans l'organisation. Cependant, dans un contexte de milliers de rôles, il est difficile de traiter et d'analyser la liste des rôles obtenus manuellement. L'administrateur a besoin d'outils automatiques pour l'aider dans cette tâche. Récemment, des outils tel que *ADVISER* [5] ont essayé de fournir une aide visuelle de présentation des rôles obtenus par mining en calculant la meilleure permutation des utilisateurs et des permissions dans la matrice UPA pour voir les rôles directement sur la matrice.

1.3.5 Synthèse et discussion

Bien que la motivation initiale de la discipline de role mining fut l'automatisation totale du processus de configuration de RBAC, il est clair aujourd'hui qu'une intervention humaine sera toujours nécessaire à toutes les étapes de processus : la sélection des données à impliquer, le paramétrage des algorithmes de role mining, et l'analyse et le tri des rôles obtenus. Cette dernière étape reste la moins explorée à ce jour. D'autres pro-

blématiques pourraient se poser comme la confidentialité des liste de contrôle d'accès. En effet, les entreprises seront souvent amenées à solliciter des experts externes pour effectuer le role mining.

Concernant l'offre commerciale, le marché du role mining a pris de l'ampleur durant les dernières années, avec l'apparition de grand éditeurs spécialisés comme *Bridgestream*, *Eurikify* et *VAAU*. Cependant, nous notons un décalage conséquent entre les techniques proposées sur le marché et les progrès réalisés dans le domaine académique.

1.3.6 Conclusion et problématiques non traitées

Le role mining est une approche prometteuse et susceptible de promouvoir l'utilisation des modèles à haut niveau d'abstraction, notamment RBAC, dans la définition et la gestion des politiques de contrôle d'accès des organisations. Cependant, les solutions proposées à ce jour sont trop simples pour satisfaire les besoins réels. Plusieurs pistes restent à explorer comme : l'extraction de rôles à partir de règles plus complexes comme les règles de contrôle d'accès dans les réseaux informatiques ou les règles faisant appel à un contexte chronologique d'actions. Des aspects comme la séparation des devoirs ou les obligations pourront être extraits aussi si les traces d'accès sont exploitées dans le role mining.

1.4 Chapitre 4 : Fouille de règles de sécurité dans un seul pare-feu

1.4.1 Formalisation de l'approche proposée

L'instanciation d'un modèle à haut niveau d'abstraction tel que Net-RBAC à partir de la configuration d'un pare-feu n'est pas tâche facile. Il existe des solutions de role mining qui extraient une configuration de rôles à partir des affectation directe utilisateur-permission. Cependant, les solutions de role mining traditionnelles ne s'appliquent pas directement sur des pare-feux. En effet, le modèle haut niveau visé pour la politique réseau est Net-RBAC et non pas RBAC.

Ene et al. [9] ont appliqué une technique de role mining standard sur des règles de pare-feu. Pour obtenir en input une matrice à deux dimensions comme requis par le role mining, ils ont confondus deux dimensions des règles de pare-feu qui sont les hôtes-source et les hôtes-destination dans une seule dimension qu'ils ont assimilé aux permissions dans RBAC. Le role mining a servi donc à regrouper les services dans des rôles. Cette initiative n'a pas eu un grand impact vu le manque d'intérêt pratique de ses résultats dans la sécurité des réseaux.

Nous proposons dans ce chapitre *policy mining* ou fouille de politique de sécurité : une approche ascendante pour extraire des instances du modèle Net-RBAC à partir des règles d'un pare-feu. Policy mining est une extension du role mining qui permet de calculer non seulement les rôles, mais aussi les activités et les vues, en réutilisant les techniques de role mining existantes. La structuration de ces trois entités abstraites donne lieu à des règles abstraites (rôle, activité, vue) qui constituent la politique à haut niveau d'abstraction du pare-feu.

Le problème de policy mining peut être formalisé comme suit :

Definition 2. *Problème de Policy Mining*
Étant donné U un ensemble d'utilisateurs, OPS un ensemble d'operations, OBJ un ensemble d'objets, et UOO la relation d'affectation utilisateur-opération-objet, trouver la configuration Net-RBAC $NS\text{-}RC = (RAV, ROLES, UR, ACTIVITIES, OPA, VIEWS, OBV)$, sous les hypothèses suivantes :

1. *Une configuration Net-RBAC sous-jacente existe*
2. *des exceptions peuvent exister.*

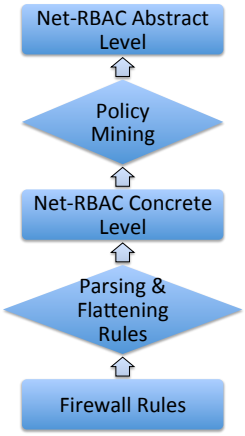


Figure 1.3 – Policy Mining pour extraire une politique de sécurité Net-RBAC à partir de la configuration d'un pare-feu.

1.4.2 Solution pour fouille de politique de sécurité

Nous considérons que les règles fournies en input sont insensibles à l'ordre et des règles positives seulement. Pour ceci, une phase de pré-traitement des règles configurées sur un pare-feu est nécessaire (voir figure 1.3). Il existe dans la littérature des solutions qui permettent d'avoir un tel résultat [31]. Après cette phase, les règles obtenues sont analysées et les différentes instances de hôtes-source, hôtes-destination, et services sont répertoriées et stockées respectivement dans les ensembles de utilisateurs, objets et opérations. L'ensemble des règles est représenté par une matrice Booléenne à 3 dimensions *UOO* (utilisateur-opération-objet).

Afin de pouvoir réutiliser les techniques existantes de role mining, qui sont assimilées généralement à une factorisation de matrice à deux dimensions, nous proposons deux algorithmes de transformation de matrice de deux dimensions à trois dimensions, et vice vers ça, sans changement de la taille des données ni des règles de contrôle d'accès. La première transformation (figure 1.4) donne toutes les combinaisons de la 2ème et de la 3ème dimensions en fonction de la 1ère. La deuxième transformation (figure 1.5) dissocie les deux dimensions par une division Euclidienne.

L'idée maîtresse de l'algorithme 1 est d'utiliser une *méthode de factorisation* qui peut être une solution de role mining classique, et ce dans la factorisation d'une matrice à 3 dimensions, notamment *UOO*. Cette méthode de factorisation permettra d'extraire les rôles, les activités, les vues, et les règles abstraites, de manière cohérente. Pour ceci, il faudra tenir compte à chaque étape de factorisation des résultats des factorisations précédentes. L'algorithme 1 enchaine une succession de transformations de nombre de dimensions de 2 à 3, factorisation, puis retour à 3 dimensions mais en remplaçant

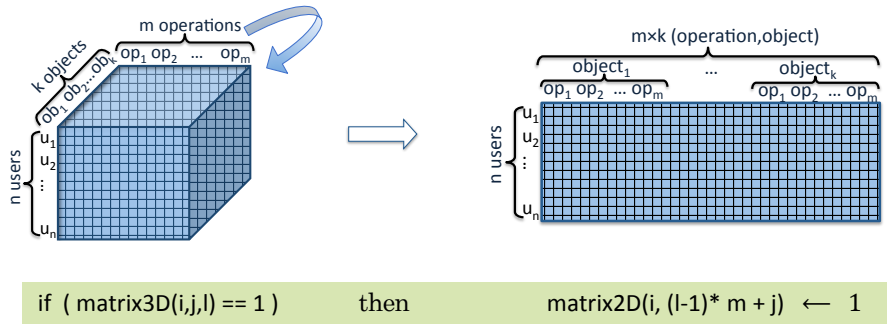


Figure 1.4 – Transformer la relation de contrôle d'accès Booléenne de 2 à 3 dimensions

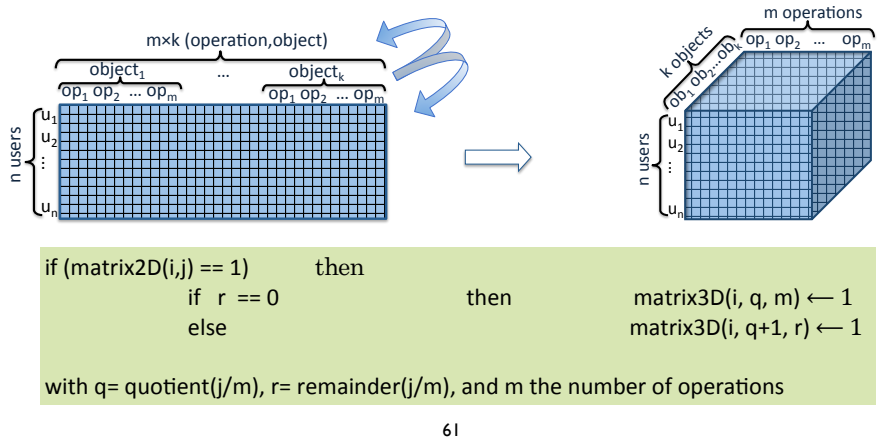


Figure 1.5 – Transformer la relation de contrôle d'accès Booléenne de 3 à 2 dimensions

une entité concrète par sa structure en entités abstraites. La matrice Booléenne à 3 dimensions résultante à la fin de l'algorithme constitue les règles abstraites de la politique Net-RBAC.

1.4.3 Exemple

L'exemple de motivation dans la figure 1.6 montre les résultats d'application de l'approche ascendante de policy mining sur un ensemble de règles tiré de la configuration réelle d'un pare-feu d'un département informatique universitaire [30]. La colonne de gauche représente l'ensemble de règles originel de la table Forward du pare-feu. la colonne de droite représente la configuration Net-RBAC qui en est extraite par policy mining. La politique Net-RBAC consiste en un ensemble de règles abstraites équivalent

Algorithm 1 Firewall Policy Mining

Input: *UOO* : the discretionary firewall rules expressed as a 3-dimensional Boolean relation between users, operations and objects

Input: FactorizationMethod

Input: Further parameters may be required by the factorization method used

Output: A Net-RBAC configuration : user-role *UR*, operation-activity *OPA*, object-view *OBV* and a Role-Activity-View *RAV* Boolean relations

1: 2D transformation : from user-operation-object to user-permission *UPA*
2: mine Roles : FactorizationMethod(*UPA*) to get : $UPA \approx UR \otimes RP$
3: 3D transformation : from role-permission matrix to role-operation-object
4: Permutation : role-operation-object to operation-object-role
5: 2D transformation : operation-object-role to operation-domain *OPD*
6: mine Activities : FactorizationMethod(*OPD*) to get : $OPD \approx OPA \otimes AD$
7: 3D transformation : activity-domain into activity-object-role
8: Permutation : activity-object-role to object-role-activity
9: 2D transformation : object-role-activity to object-capacity *OBC*
10: mind Views : FactorizationMethod(*OBC*) to get : $OBC \approx OBV \otimes VC$
11: 3D transformation : view-capacity into view-activity-role
12: Permutation : view-activity-role into role-activity-view
13: **return** *RAV*, *UR*, *OPA*, and *OBV*

aux règles initiales. En plus, les instances de hôte-source, hôte-destination, et services sont structurées dans respectivement des rôles, activités et vues.

1.4.4 Conclusion et perspectives

Nous avons dédié ce chapitre à l'établissement du lien entre le problème de gestion de pare-feu, et les techniques de role mining récemment proposées. Nous avons présenté une solution ascendante pour la fouille des règles abstraites et des entités abstraites d'une politique Net-RBAC dans les règles implémentées sur un pare-feu. Une perspective intéressante de ce travail serait de l'appliquer sur des pare-feux à état. À cet égard, nous envisageons d'utiliser le concept de *contexte* défini dans le modèle *OrBAC* pour enrichir le modèle Net-RBAC. De même, la solution présentée ici propose de traduire les règles en règles positives dans une politique fermée. Le contraire serait intéressant pour un administrateur de sécurité réseau aussi. Finalement, cette technique de policy mining serait utile au delà de l'application à la sécurité des réseaux, et ce pour toute application dans laquelle les règles de contrôle d'accès sont tertiaires.

Original Set of FW iptables Rules	Policy Mining Results
Concrete Rules: 1- -p tcp -d 192.168.1.250 --dport domain 2- -p tcp -d 192.168.1.251 --dport smtp 3- -p tcp -d 192.168.1.251 --dport smtps 4- -p tcp -d 192.168.1.251 --dport imaps 5- -p tcp -d 192.168.1.251 --dport pop3s 6- -p tcp -d 192.168.1.252 --dport http 7- -p tcp -d 192.168.1.126/25 --dport auth 8- -s 192.168.1.126/25 -p tcp -d 192.168.1.13 --dport ssh 9- -s 192.168.1.126/25 -p tcp -d 192.168.1.14 --dport ssh 10- -s 192.168.1.126/25 -p tcp -d 192.168.1.15 --dport ssh 11- -s 192.168.1.126/25 -p tcp -d 192.168.1.20 --dport ssh 12- -p tcp -d 192.168.1.252 --dport https 13- -s 192.168.1.254/28 -d 192.168.1.11 -p tcp --dport sunrpc 14- -s 192.168.1.236 -p tcp -d 192.168.1.35 --dport ipp 15- -s 192.168.1.254/28 -d 192.168.1.11 -p udp --dport nfs 16- -s 192.168.1.254/28 -d 192.168.1.11 -p udp --dport 4000:4002 17- -p udp -d 192.168.1.251 --dport smtp -j 18- -p udp -d 192.168.1.250 --dport domain -j 19- -s 192.168.1.254/28 -p udp -d 192.168.1.11 --dport sunrpc 20- -s 192.168.1.236 -p udp -d 192.168.1.35 --dport ipp 21- -d 192.168.1.126/25 -p icmp --icmp-type destination-unreachable 22- -d 192.168.1.126/25 -p icmp --icmp-type parameter-problem 23- -d 192.168.1.126/25 -p icmp --icmp-type source-quench 24- IPTABLES -A FORWARD -j REJECT	*Abstract Rules:* *1- FROM Corporate_Zone TO SSH_Server FOR Secured_Traffic* *2- FROM Any TO Web_Server FOR Secured_Web_Service* *3- FROM Any TO Corporate_Zone FOR Protocol_Control* *4- FROM Exterior TO Printer FOR Printing* *5- FROM DMZ TO SUN_NFS_Server FOR UNIX_Services* *6- FROM Any TO DNS_Server FOR DNS* *7- FROM Any TO Messaging_Server FOR E-mailing* *Source_Host to Role assignment:* *1- Corporate_Zone= {192.168.1.126/25}* *2- Exterior = {192.168.1.236}* *3 -DMZ = {192.168.1.254/28}* *4- Any = {any}* *Service to Activity assignment:* *1- Secured_Traffic = {-p tcp -dport ssh}* *2- Secured_Web_Service = {{-p tcp -dport auth}, {-p tcp -dport https}, {-p tcp -dport http}}* *3 -Ptrotocol_Control = {{tcp FOR auth}, {-p icmp -icmp-type: destination unreachable},{ -p icmp -icmp-type parameter-problem}, { -p icmp -icmp-type source-quench}}* *4- Printing = {{-p tcp -dport ipp}, {-p udp -dport ipp}}* *5- UNIX_Services = {-p tcp -dport sunrpc}, { -p udp -dport sunrpc}, {-p udp -dport nfs}, {-p udp -dport 4000:4002}}* *6- DNS = {{-p tcp -dport domain},{-p udp -dport domain}}* *7- E-mailing = {{-p tcp -dport smtp}, {-p tcp -dport smtps}, {-p tcp -dport imaps}, {-p tcp -dport pop3s}, {-p udp -dport smtp}}* *Destination_Host to View assignment:* *1- SSH_Server = {[192.168.1.13 - 15] and 192.168.1.20}* *2- Web_Server = {192.168.1.252}* *3 - Corporate_Zone = {192.168.1.126/25}* *4- Printer = {192.168.1.35}* *5- Sun_NFS_Server = {192.168.1.11}* *6- DNS_Server = {192.168.1.250}* *7- Messaging_Srever = {192.168.1.251}*

Figure 1.6 – Exemple de résultat de policy mining sur un pare-feu.

1.5 Chapitre 5 : Fouille de règles de sécurité dans un réseau protégé par plusieurs pare-feux

1.5.1 Approche ascendante pour extraire la politique de contrôle d'accès réseau

Les réseaux informatiques de grandes et moyennes entreprises nécessitent l'usage de plusieurs pare-feux pour leur protection. La technique de policy mining présentée dans le chapitre précédent fournit une approche ascendante qui extrait automatiquement une instance de Net-RBAC à partir des règles configurées un seul pare-feu.

Pour apporter une solution complète pour la fouille de règles de sécurité d'un réseau entier, nous avons encore besoin d'un traitement supplémentaire sur les politiques calculées sur chaque pare-feu à part par policy mining, afin d'extraire la politique de contrôle d'accès haut niveau de plusieurs pare-feux. Dans ce chapitre, nous développons le problème d'intégration de politiques Net-RBAC résultantes de policy mining sur plusieurs pare-feux protégeant le même réseau. Nous proposons un traitement en deux phases (voir figure 1.7). Dans la première phase, nous unifions les hiérarchies d'entités abstraites Net-RBAC de tous les pare-feux. Dans la deuxième phase, nous construisons l'ensemble des règles abstraites effectivement déployé dans le réseau. Nous vérifions des propriétés de sécurité relatives à la cohérence de déploiement des règles entre les pare-feu, ainsi qu'à la pertinence de déploiement.

1.5.2 Intégration des entités abstraites

Dans net-RBAC, les entités abstraites sont définies à la fois par les permissions et par les entités concrètes qui leur sont attribuées. Par la dernière perspective, chaque rôle (respect. vue) est un ensemble de hôtes-sources (respect hôtes-destination) représentés par des adresse IP et/ou des masques réseau. De même, les activités sont des ensembles de services caractérisés par leurs protocoles, options, et numéros de port. En plus, ces entités abstraites peuvent être hiérarchiques. Ainsi, un ensemble d'entités abstraites peut être assimilé à un *ensemble partiellement ordonné (Poset)*, en respectant la relation d'inclusion. Pour obtenir un ensemble homogène d'entités abstraite dans le réseau, nous devons unifier les hiérarchies de rôles, activités, et vues obtenus par policy mining sur les différents pare-feux protégeant le réseau. Nous comparons les élément attribués à chaque entité abstraite afin de détecter les redondances et les nouvelles relations d'hiérarchie. Il s'agit d'un problème d'intégration de posets. L'algorithme 2 que nous

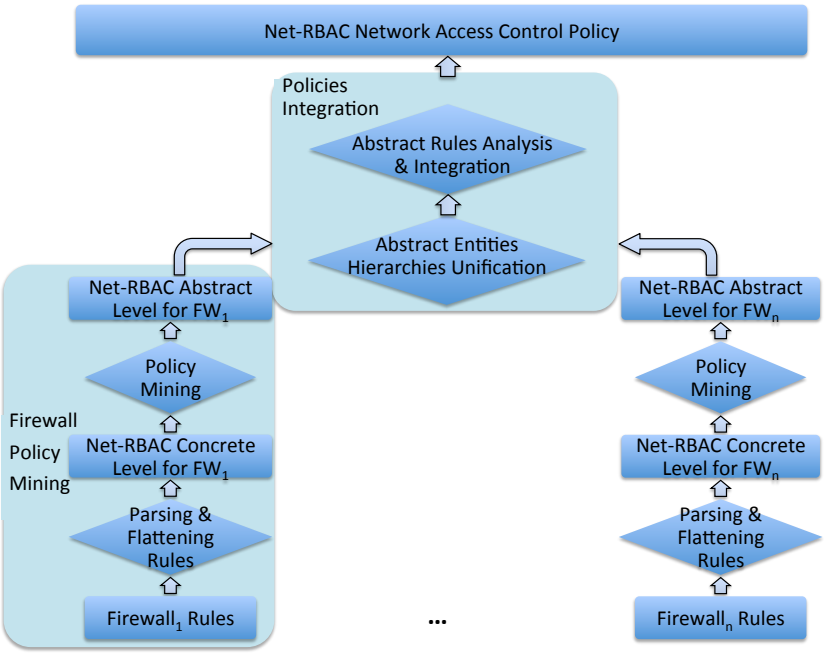


Figure 1.7 – Policy mining pour extraire une politique Net-RBAC de plusieurs pare-feux.

présentons ici traite seulement les rôles, le processus étant le même pour les activités et les vues. Cet algorithme intègre les rôles d'un pare-feu dans l'ensemble des rôles du réseau. Il prend en input les rôles du pare-feu et leur hiérarchie représentés par des matrices Booléennes, ainsi que les rôles courants du réseau et leur hiérarchie. L'algorithme détecte toute les redondances et les relations de hiérarchie entre les rôles des deux ensemble. Le output de l'algorithme 2 est l'ensemble courant des rôles du réseau mis à jour après l'intégration des rôles du pare-feu, et la relation d'hiérarchie mise à jour également.

1.5.3 Intégration des règles abstraites

La politique de contrôle d'accès du réseau qui est visée par cette approche ascendante est un ensemble de règles abstraites, indépendantes de la topologies sauf pour ce qui est relatif à des décision de routage pour des considération de sécurité à un haut niveau d'abstraction. Nous vérifions par ailleurs que les règles répondent à deux propriétés de sécurité : la pertinence d'implémentation, et l'accessibilité. Les input de l'algorithme 3 sont les règles abstraites de chacun des pare-feux du réseau. Les entités abstrates redondante entre les règles de plusieurs pare-feu sont identifiée et substituées par des noms unifiés. L'algorithme se base aussi sur les chemins de routage et la position des

Algorithm 2 Abstract Entities Merger

Input: NUR and NRH : an $l \times m$ and an $m \times m$ Boolean matrices of the user's assignment and hierarchy of current roles in the network policy

Input: FUR and FRH : an $l \times n$ and an $n \times n$ Boolean matrices of the user's assignment and hierarchy of the roles in the firewall to be integrated in the network policy

Output: NUR' and NRH' : an $l \times m'$ and an $m' \times m'$ Boolean matrices of the user's assignment and hierarchy of the roles in the updated network policy

create $n \times m$ matrix K with all the cells initialized to nh
for each role fR_i in FRH (i from 1 to n) **do**
  for each role nR_j in NRH (j from 1 to m) **do**
    compare fR_i to nR_j
    if $fR_i == nR_j$ **then**
      The row i in K is set to r
      for each fR_v sub-role of fR_i in FRH ; **for** each nR_w senior-role of nR_j in NRH ; $K_{vw} \leftarrow l$
      for each fR_v senior-role of fR_i in FRH ; **for** each nR_w sub-role of nR_j in NRH ; $K_{vw} \leftarrow h$
    else
      if $K_{i,j} == nh$ **then**
        switch comparison of fR_i to nR_j **do**
          case 1 : $fR_i \subsetneq nR_j$
            $K_{ij} \leftarrow l$
            for each fR_v sub-role of fR_i in FRH ; **for** each nR_w senior-role of nR_j in NRH ; $K_{vw} \leftarrow l$
          end case
          case 2 : $fR_i \supsetneq nR_j$
            $K_{ij} \leftarrow h$
            for each fR_v senior-role of fR_i in FRH ; **for** each nR_w sub-role of nR_j in NRH ; $K_{vw} \leftarrow h$
          end case
          case 3 : $fR_i \cap nR_j == \emptyset$
            $K_{ij} \leftarrow d$
            for each fR_v sub-role of fR_i in FRH ; **for** each nR_w sub-role of nR_j in NRH ; $K_{vw} \leftarrow d$
          end case
          case
            $K_{ij} \leftarrow d$
          end default
        end switch
      end if
    end if
  end for
end for

32: $m' \leftarrow m$
33: **for** each role fR_v corresponding to $row_v \neq r$ in K **do**
34: $m' \leftarrow m' + 1$
35: add a column m' in NUR' zeroed with assigned_users(fR_v) set to 1
36: add a row m' and a column m' to NRH' zeroed
37: **for** w :=1 to m **do**
38: **if** $K_{vw} == l$ **then** $NRH'_{m'w} \leftarrow 1$
39: **else if** $K_{vw} == h$ **then** $NRH'_{wm'} \leftarrow 1$ **end if**
40: **end for**
41: **end for**
42: **return** NUR' and NRH'

pare-feux dans la topologie. L'algorithme 3 vérifie l'accessibilité et la pertinence de chaque règle de chaque pare-feu avant de l'ajouter à la politique finale du réseau. Une règle est pertinent si le pare-feu où elle est implémentée se situe sur le chemin entre les zones du rôle et de la vue de la règle. Si la règle est pertinente, algorithm 3 vérifie sont accessibilité à travers tout le chemin. S'il y a un ou plusieurs autres pare-feux sur le chemin entre la zone rôle et la zone destination, ces pare-feux doivent aussi permettre le passage du trafic caractérisé par la règle. Sinon, la règle n'est pas proprement déployée dans le réseau, et ne devrait pas être jointe à l'ensemble des règles de la politique finale. Le problème d'inaccessibilité peut être global, ou partiel si un sous ensemble du trafic est accessible tout le long du chemin. Le output de l' algorithme 3 est l'ensemble des règles de la politique réseau suivant le modèle (rôle, activité, vue, et optionnellement chemin_contexte), ainsi qu'un rapport d'anomalies d'impertinence et d'inaccessibilité.

1.5.4 Exemple

Nous considérons une politique de contrôle d'accès pour l'architecture illustrée dans la figure 1.8. Nous supposons que la politique stipule que le trafic des clients et des employés de l'entreprise doivent être séparés et passer par deux réseau écrans indépendants. Les employés peuvent accéder à l'Internet à travers perimeter_network_2. Une partie de ce trafic doit être relayé par le Bastion_Host. Les serveurs ont des interfaces publiques et peuvent recevoir des requêtes des clients externes à travers perimeter_network_1. L'administration et les archives des serveurs sont effectués depuis le réseau privé interne de l'entreprise. Quatre pare-feux assurent le respect de cette politique [8].

Algorithm 3 Abstract Rules Merger

Input: The set of abstract rules (R,A,V) of each firewall in the network

Input: The routing paths

Output: The set of abstract rules of the access control policy of the network (R,A,V, context_path)

Output: The report of verification of relevance and accessibility of the firewall rules

1: **for** each firewall FW_i in the network **do**
2: **for** each rule r_j in FW_i **do**
3: Path ← get_path(Role(r_j), View(r_j))
4: **if** $FW_i \notin Path$ **then**
5: Report (Irrelevance, FW_i, r_j)
6: **else**
7: Path_Accessibility ← True
8: **for** each firewall FW_k in Path (other than FW_i) **do**
9: FW_Accessibility ← False
10: **if** FW_k is not already handled (in line 1) **then**
11: **for** each rule r_l in FW_k **do**
12: **switch** comparison of r_j and r_l **do**
13: **case** 1 : disjoint
14: do nothing
15: **end case**
16: **case** 2 : exact match
17: FW_Accessibility ← True
18: do not consider r_l from FW_k again
19: **end case**
20: **case** 3 : r_l is a generalization of r_j
21: FW_Accessibility ← True
22: cache_r_l_FW_i ← cache_r_l_FW_i $\cup$ $r_j \cap r_l$
23: **end case**
24: **case** 4 : r_j is a generalization of r_l
25: cache_r_j_FW_k← cache_r_j_FW_k $\cup$ $r_j \cap r_l$
26: do not consider r_l from FW_k again
27: **end case**
28: **case**
29: cache_r_j_FW_k← cache_r_j_FW_k $\cup$ $r_j \cap r_l$
30: cache_r_l_FW_i ← cache_r_l_FW_i $\cup$ $r_j \cap r_l$
31: **end default**
32: **end switch**
33: **end for**
34: **end if**

```
            if cache_r_j_FW_k == r_j then
              FW_Accessibility ← True
            end if
            if Not FW_Accessibility then
              Path_Accessibility ← False
            end if
          end for
          if Path_Accessibility then
            Add {r_j,Path_context} to Network Policy Rules
          else
            Report (Inaccessibility, r_j)
          end if
        end if
      end for
    end for
```

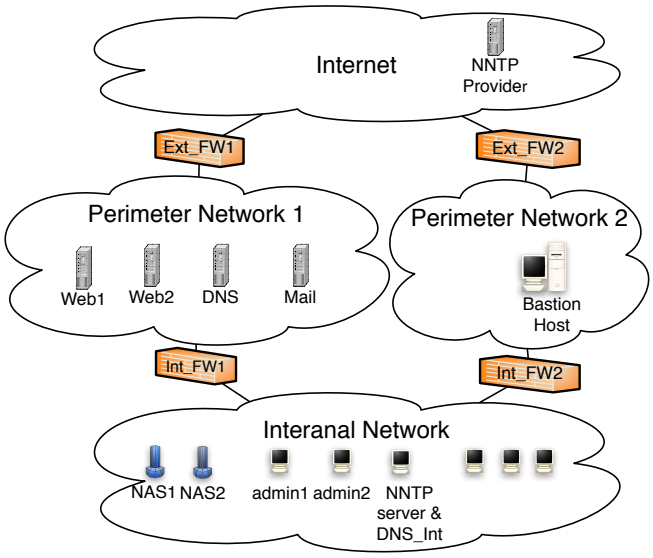


Figure 1.8 – La topologie du réseau.

La figure 1.9 montre les règles abstraites et les entités abstraites résultantes du policy mining appliqué à chacun des quatre pare-feux à part. Ce sont les input du processus d'intégration de politiques. La figure 1.11 montre les résultats de l'intégration des entités abstraites par l'algorithme 2. Les résultats de l'intégration des règles abstraites par l'algorithme 3 est dans la figure 1.10. On y trouve le rapport de vérification d'accessibilité et de pertinence aussi.

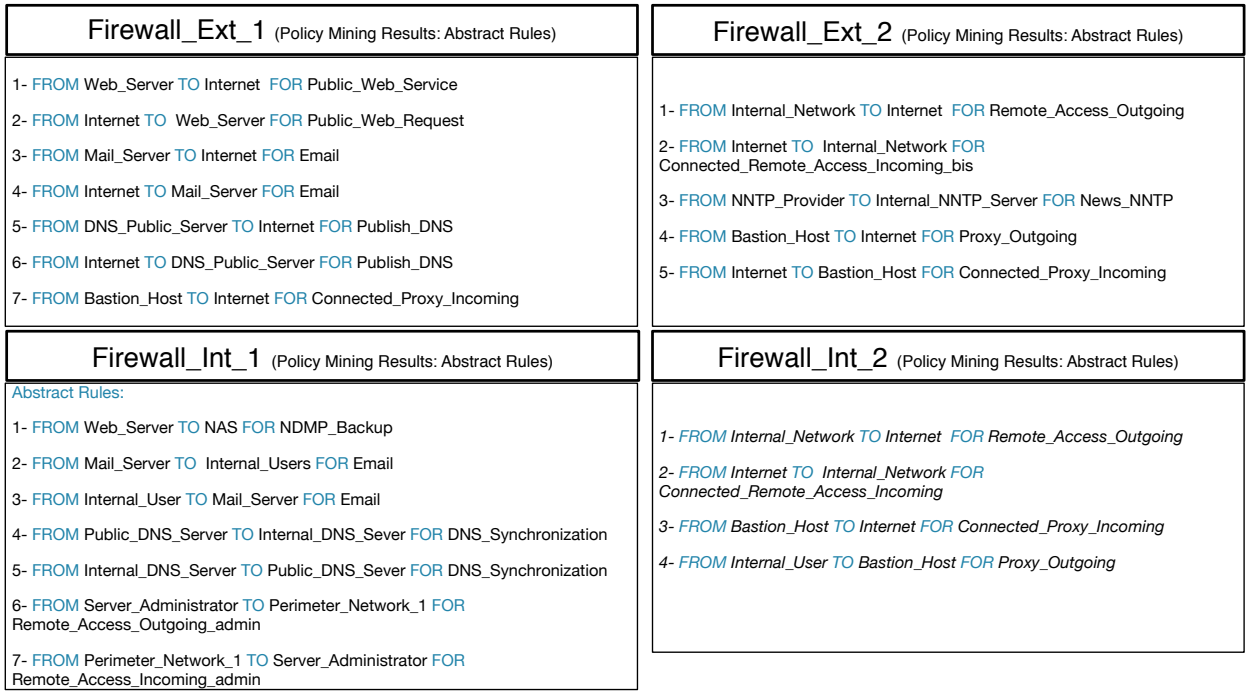


Figure 1.9 – Les règles abstraites résultant de policy mining sur chaque pare-feu.

1.5.5 Conclusion et perspectives

Dans ce chapitre, nous avons proposé une solution d'intégration de politiques Net-RBAC de plusieurs pare-feux impliqués dans la défense d'un même réseau. Ce processus est défini pour compléter l'approche ascendante de fouille de politique de sécurité modélisée par Net-RBAC à partir des règles configurées sur des pare-feux. En plus de l'intérêt pratique dans l'application de la gestions e la sécurité des réseaux informatiques, la solution d'intégration des entités abstraite pourrait être utilisée dans le cas général d'intégration de posets. La solution d'intégration de politiques Net-RBAC pourrait être étendue pour OrBAC en vue de supporter des pare-feux à état.

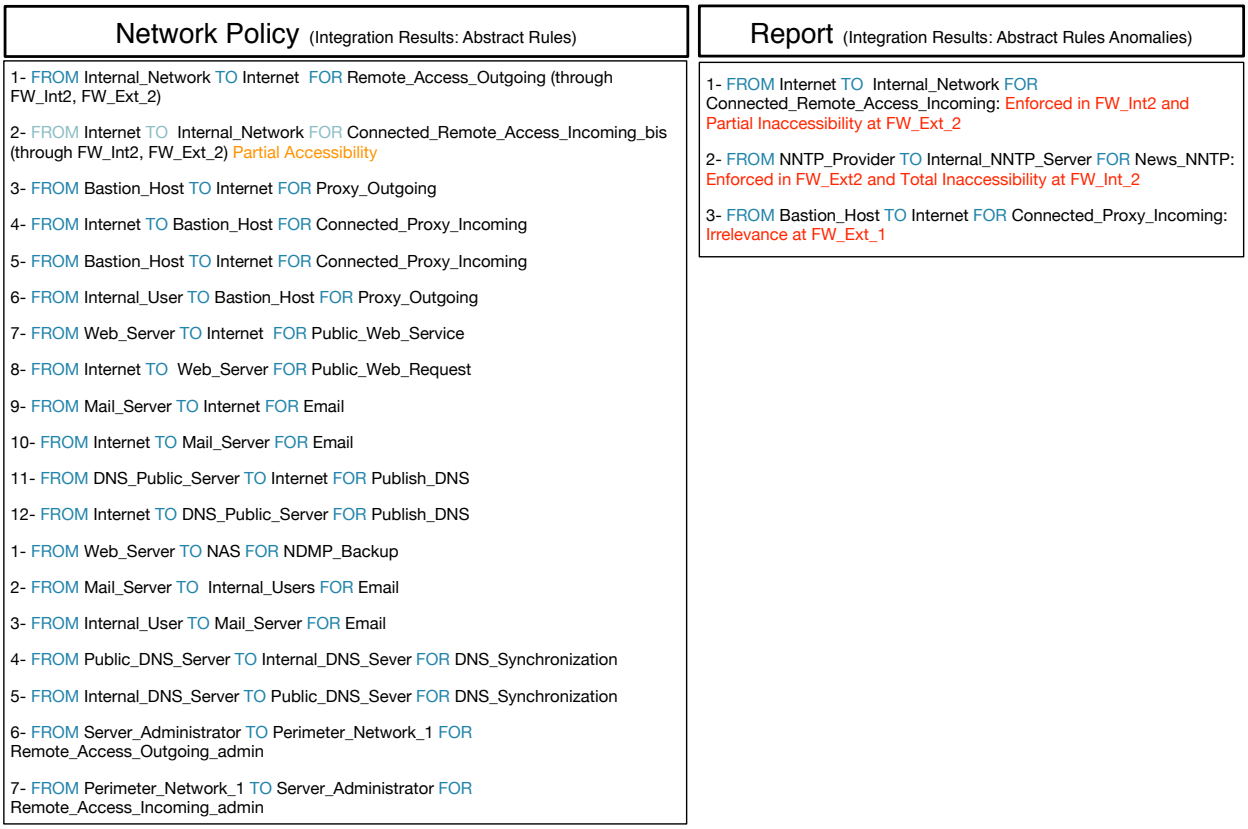


Figure 1.10 – Les règles de sécurité résultant de l'intégration des politiques des pare-feux.

1.6 Chapitre 6 : Comparaison et analyse d'ensembles de rôles

1.6.1 Motivation

Le besoin de comparer deux ensembles de rôles se présente dans plusieurs cas d'utilisation reliés au role mining. Le premier cas est l'évaluation d'un nouvel algorithme de role mining par la technique d'ingénierie inversée. La technique de role mining est appliquée sur une liste de contrôle d'accès dérivée d'un ensemble de rôles originel, ensuite les rôles calculés par mining sont comparés aux rôles originaux. Le deuxième cas d'utilisation est l'exploitation des rôles extraits par mining en les affectant aux utilisateurs de manière optimale. En effet, les techniques de rôles génèrent souvent des liste de rôles définis uniquement par leurs permissions, et parmi les quels il faudra choisir. Dans ce cas, nous considérons les groupes de permissions attribuées à chaque utilisateur comme des pseudo-rôles. Dans le cadre de l'exploitation des résultats de role mining aussi, la

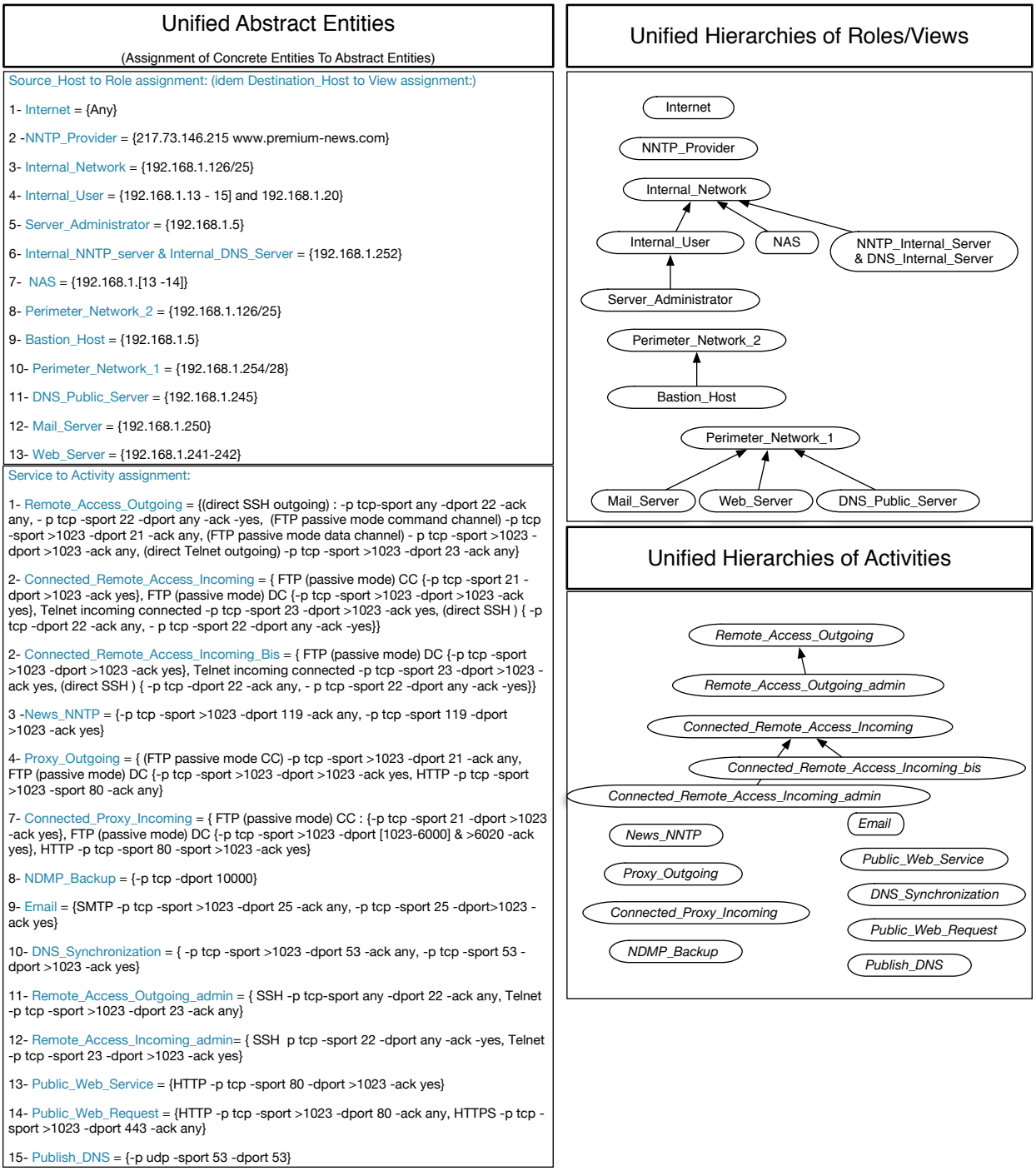


Figure 1.11 – Les entités abstraites du réseau et leur hiérarchies après intégration.

comparaison des rôles générés par role mining à un ensemble de rôles préexistants et maîtrisés peut aider un administrateur de sécurité dans l'analyse des nouveaux rôles obtenus.

Tableau 1.1 – Exemple de motivation : Configurations de roles RBAC originale et extraite par mining équivalentes.

(a) Original Roles Matrix *OR*

Original Roles	cTrans(p1)	rP-order(p2)	cP-order(p3)	vTrans(p4)
Handle Order(r1)	1	1	0	0
Create Order(r2)	0	0	1	0
Supervise Transfer(r3)	0	1	0	1

(b) Mined Roles Matrix *MR*

Mined Roles	cTrans(p1)	rP-order(p2)	cP-order(p3)	vTrans(p4)
Manage Order(R1)	1	1	1	0
Validate Transfer(R2)	0	0	0	1

(c) Original User to Role Assignment Matrix *UR*1

User-ORole	r1	r2	r3
U1	1	1	0
U2	1	1	1
U3	0	0	0
U4	1	1	0
U5	1	1	0

(d) Mined User to Role Assignment Matrix *UR*2

User-MRole	R1	R2
U1	1	0
U2	1	1
U3	0	0
U4	1	0
U5	1	0

(e) Current User to Permission Assignment Matrix *UPA*

User-Perm	p1	p2	p3	p4
U1	1	1	1	0
U2	1	1	1	1
U3	0	0	0	0
U4	1	1	1	0
U5	1	1	1	0

Dans quasiment tous ces cas d'utilisation, on a affaire à un ensemble de rôles de référence ou rôles originaux *OR*, et un ensemble de rôles résultant du mining *MR*. Les deux configurations de rôles sont définies pour les mêmes utilisateurs et les mêmes permissions.

L'exemple dans le tableau 1.1 illustre le problème traité dans ce chapitre. Il montre deux ensembles de rôles équivalents pour un département des finances hypothétique. Ils pourraient correspondre à une ancienne et un nouvelle configurations de rôles.

Les deux configurations RABC dans le tableau 1.1 sont telles que : $UR2 \otimes OR = UR1 \otimes OR = UPA$, ce qui veut dire que ces deux configurations sont *équivalentes* selon la définition suivante :

Definition 3. *Ensembles de roles équivalents*
Deux ensembles de rôles $RP1$ *et* $RP2$ *sont* équivalents *s'ils peuvent mener à la même relation utilisateur-permission UPA dans un système donnée, i.e. il existe deux distributions possibles* $UR1$ *et* $UR2$ *pour* $UR1$ *and* $UR2$ *telles que* $UR1 \otimes RP1 = UR2 \otimes RP2 = UPA$ *;*

1.6.2 Problème de comparaison de deux configurations de rôles

Nous proposons une nouvelle approche pour comparer deux configurations de rôles. Notre approche vise à exploiter toutes les relations entre rôles des deux ensembles comparés : rôles regroupés, rôles divisés, relations hiérarchiques, exceptions, etc. Ces relations nous permettrons de comprendre un nouvel ensemble de rôles en exploitant l'expérience acquise avec un ancien ensemble de rôles dans une même organisation. Dans ce but, notre méthode repose sur l'expression de chaque rôle d'un ensemble par une formule algébrique de rôles de l'ensemble comparé. Nous jugeons que la *forme normale disjonctive (DNF)* est un choix naturel pour cette expression.

Le problème traité dans ce chapitre est formalisé comme suit :

Definition 4. *Problème de comparaison d'ensembles de rôles*
Étant donné deux ensembles de rôles OR and MR, définis sur le même ensemble de permissions PRMS, trouver pour chaque role R de OR une forme normale disjonctive minimale (DNF) de rôles de MR, telle que DNF soit incluses dans R, et DNF couvre au maximum les permissions de R.

Par exemple, dans l'exemple d motivation dans le tableau 1.1, le problème de la définition 4 cherche à établir que $R1 = r1 \cup r2$

Nous pouvons montrer que le problème de la définition 4 est NP-Complet, puisque la vérification d'une solution se fait en temps polynomial, et chaque instance du problème peut être réduite en un temps polynomial à une instance du problème *Role Set Covering*, connu pour être NP-Complet.

1.6.3 Solution de comparaison de deux configurations de rôles

L'algorithme 4 permet de résoudre le problème de comparaison d'ensembles de rôles (RSCP). Le temps d'exécution de l'algorithme est exponentiel dans le pire des cas, où on ne trouve pas de DNF exacte pour un rôle, et il est polynomial dans le meilleur des cas, où tous les rôles de *MR* ont des projections de DNF d'un nombre borné de littéraux dans l'ensemble *OR*.

1.6.4 Conclusion et perspectives

Dans ce chapitre, nous avons proposé une solution pour aborder et maîtriser les résultats de role mining, en comparant les rôles obtenus à un ensemble de rôles de référence.

Algorithm 4 Compare Two Sets of Roles

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 5 Generate Conjunctive Clauses

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
```

Nous avons apporté une approche formelle basée sur la projection algébrique des rôles d'un ensemble dans un autre, en utilisant des DNFs. La complexité de l'algorithme proposé dépend de la nature des données input. Dans le chapitre suivant, nous allons caractériser formellement la notion d'ensembles de rôles *comparables*.

1.7 Chapitre 7 : Detection de rôles masqués

1.7.1 Corrélation entre rôles masquées et résultats de comparaison de configurations de rôles

D'après notre analyse du temps de complexité de l'algorithme 4, le temps d'exécution devrait être bas lorsque une DNF exacte se trouve pour chaque role dans l'ensemble de référence. Nous nous sommes intéressés à caractériser les configurations de rôles comparés qui satisfont cette condition. Dans un cadre de role mining, les deux configurations de rôles données en entrée à l'algorithme 4 sont *équivalents* (voir la définition 3). Or, un role est caractérisé à la fois par les utilisateurs qui sont affectés et par les permissions qui lui sont attribuées. Dans le chapitre précédent, nous avons considéré les rôles de la perspective de leurs permissions seulement. Pour mieux caractériser une configuration de rôles, nous devons nous intéresser aussi au utilisateurs.

Theorem 1. ***La garantie de complétude conditionnelle.*** *Étant donnés deux ensembles de roles* équivalent *RP1 et RP2; Pour tout role R de RP1, si un sous ensemble de k utilisateurs existe tel que :* $\bigcap_{i=1}^{k} assigned\_perms(U_i) = R$*, alors il existe au moins une forme normale disjonctive DNF de rôles de RP2 qui correspond exactement à R.*

Le corollaire suivant stipule que la condition sus-mentionnée est satisfaite en particulier par les rôles non chevauchants avec d'autres rôles dans une configuration :

Corollary 1. ***La garantie de complétude pour les rôles indépendants.*** *Étant donnés deux ensembles de rôles* équivalents *RP1 et RP2; Si les rôles de RP1 ne se chevauchent pas, alors, pour tout role R de RP1, si R est affecté à au moins un utilisateur, et si il n'existe aucun autre role* $R' \in RP1$ *tel que assigned_users(R)* $\subseteq$ *assigned_users(*R'*), alors il existe au moins une forme normale disjonctive DNF de rôles de RP2 qui corresponde exactement à R.*

Donc si l'algorithme 4 ne trouve pas de DNF exacte pour un certain rôle ceci veut dire que les conditions de théorème 1 et du corollaire 1 ne sont pas satisfaites.

1.7.2 Définition des rôles masqués

Pour traiter le problème de rôles masqués, nous fournissons d'abord une définition des différents cas de masquage.

Definition 5. *Rôles masqués*
Un rôle R est masqué dans une configuration RBAC : RC = (ROLES, UR, RP) s'il correspond à un ou plusieurs des cas suivants :

1. *R n'est affecté à aucun utilisateur i.e. assigned_users(R) = ∅.*
   ou
2. *Il existe au moins un autre role R′ dans ROLES qui est affecté au mêmes utilisateurs que R : i.e. assigned_users(R) = assigned_users(R').*
   ou
3. *Rpartage des permissions avec un ou plusieurs rôles dans ROLES, et il existe au moins une permission p ∈ assigned_perms(R) telle que les utilisateurs affectées à R obtiennent toujours p de leurs autres rôles*

L'idée derrière la définition 5 est que un rôle est considéré comme masqué dans une configuration donnée si un algorithme de rôle mining ne peut pas l'extraire en entier de la relation *UPA*.

Nous proposons algorithme6 pour examine une configuration de rôles et détectes les rôles masqués.

1.7.3 Conclusion

Nous avons examiné le problème de rôles masqués, souvent relié à une anomalie de configuration. Nous avons défini formellement les cas de masquage et nous avons relié ce problème au problème de comparaison de configurations de rôles équivalentes. Nous avons défini un algorithme pour détecter les rôles masqués.

Algorithm 6 Detect Shadowed Roles

Input: UR : a user to role assignment matrix

Input: RP : a role to permission assignment matrix

Output: A specification for each role in UR matrix : whether it is “not shadowed", “not assigned", or “shadowed" with the specification of the “shadowed permissions".

```
CandidateRoles ← Check For Partitions Of Roles(UR)
NBUPA ← UR × RP
RecurrentPerms ← permissions assigned to a user more than once in NBUPA
if RecurrentPerms ≠ ∅ then
  for each role R in CandidateRoles do
    RIsShadowed ← False
    ShadowedPerms ← {}
    if assigned_users( R ) ≠ ∅ then
      Report : “The role <R> is not assigned to any user."
    else
      ProblematicPerms ← assigned_perms(R) ∩ RecurrentPerms
      for each permission p in ProblematicPerms do
        OnceUsers ← assigned_users( R ) ∩ assigned_users_only_once( p)
        if OnceUsers == ∅ then
          RIsShadowed ← True
          ShadowedPerms ← ShadowedPerms ∪ {p}
        end if
      end for
      if RIsShadowed then
        Report : “The role <R> is shadowed. The shadowed permissions are <ShadowedPerms>.”
      else
        Report : “The role <R> is not shadowed.”
      end if
    end if
  end for
else
  for each role R in CandidateRoles do
    if assigned_users( R ) ≠ ∅ then
      Report : “The role <R> is not assigned to any user.”
    else
      Report : “The role <R> is not shadowed.”
    end if
  end for
end if
```

Algorithm 7 Check For Partitions Of Roles

Input: RP : a role to permission assignment matrix

Output: NonPartitionsRoles

Output: This algorithm should check for Roles always assigned together to the same users and report them as shadowed roles.

1: Apply a Hash function on each column of RP
2: Sort the columns of RP according to the Hash values
3: NonPartitionsRoles ← Roles with a unique instance of Hash Value
4: Report : "The roles in <RP∪NonPartitionsRoles> are shadowed. They are partitions of roles."
5: **return** NonPartitionsRoles

1.8 Chapitre 8 : Évaluation expérimentale

1.8.1 Platforme de test

Nous avons implémenté une plateforme de test pour prouver la faisabilité et évaluer la performance de toutes les solutions présentées dans cette thèse. Nous avons défini deux générateurs de configurations RBAC et de configurations Net-RBAC pour obtenir les données synthétiques nécessaires aux tests. Ces générateurs prennent en entrée les caractéristiques souhaitées des configurations (nombre d'utilisateurs, de permissions, de rôles, de règles, densité d'affectation, etc.) et génèrent les matrices Booléennes aléatoires correspondantes.

Nous avons aussi adapté un nombre de méthodes de factorisations, souvent utilisées dans le role mining, pour les besoins du policy mining. Ces méthodes ont été choisies de telle sorte à avoir un échantillon représentatif des objectifs d'optimisation du role mining. Parmi ces méthodes de factorisation, nous notons *Singular Value Decomposition (SVD)* [17], choisi pour sa scalabilité et sa simplicité, *Non Negative Matrix Factorization (NNMF)* [19] and the *Binary Non-Orthogonal Decomposition (BNOF)* [18] qui constituent des exemples simples de méthodes de factorisation sans erreur d'approximation, et finalement *Formal Concept Analysis (FCA)* algorithm [4, 15] comme exemple de technique de role mining qui génère des rôles hiérarchiques.

1.8.2 Comparaison de configurations de roôes

Pour cette section, nous avons généré quatre séries de données synthétiques, qui sont décrites dans le tableau 1.2.

Tableau 1.2 – Taille des configurations RBAC synthétiques

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

Nous avons utilisé les séries de données 1,2,3 et4 pour effectuer des tests de role mining. Nous donnons en entrée les matrices *UPA* de chaque série de données à un algorithme de role mining. Nous calculons les rôles, puis nous les comparons avec les rôles initiaux générés dans la même série de donnée. Nous utilisons l'algotihme 4 pour effectuer la comparaisons. Les résultats sont dans la figure 1.12.

Dans une seconde étape, nous avons utilisé les séries de données 5, 6, 7 et 8 du tableau 1.2 pour tester la capacité de l'algorithme 4 à affecter automatiquement les utilisateurs d'un système aux rôles calculés par mining. Les résultats sont visibles dans les figures 1.12(e) et 1.12(f).

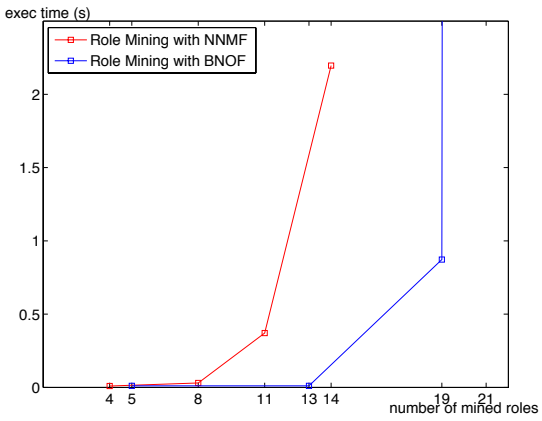
1.8.3 Détection de rôles masqués

Pour tester la solution de détection de rôles masqués décrit par l'algorithme 6, nous avons généré les séries de données contenant des configurations RBAC et décrites dans le tableau 1.3. Pour chaque taille de donnée, nous avons généré quatre séries avec une densité croissante de 0.1, 0.3, 0.5 and 0.7. La figure 1.13 récapitule les résultats de ces expériences.

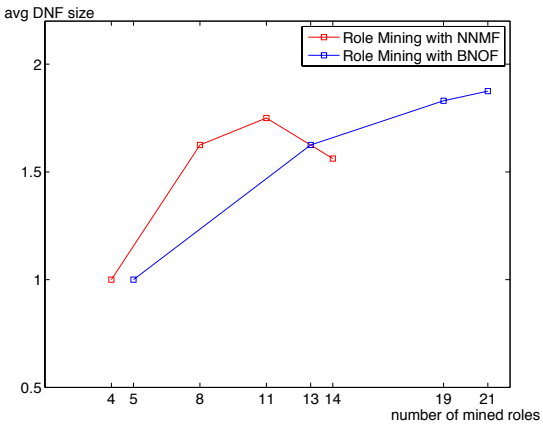
Tableau 1.3 – Taille des configurations RBAC synthétiques

n	nUsers	nPerms	nRoles
9	800	1000	300
10	1000	1200	600
11	1500	2000	800

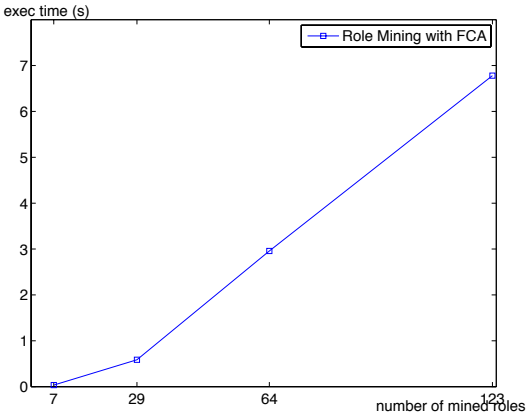
La figure 1.13(a), montre le temps d'exécution de l'algorithme en terme de nombre de rôles dans la configurationRBAC.



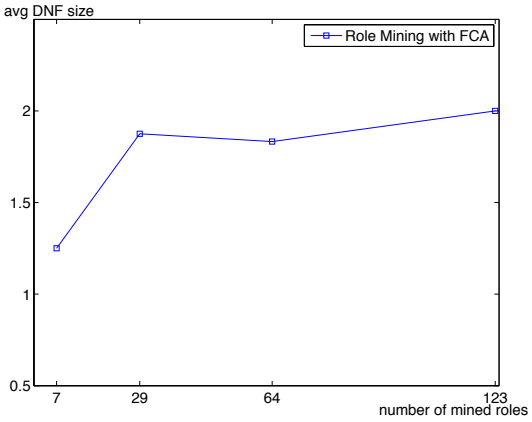
(a) Temps d'exécution pour la comparaison de rôles des séries 1, 2, 3 and 4 avec les rôles obtenus par mining avec NNMF et BNOF



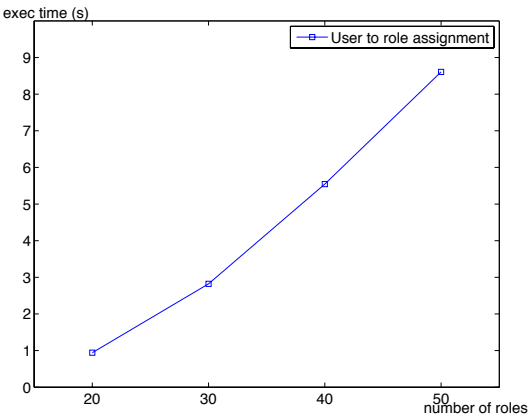
(b) Taille moyenne des DNFs avec NNMF et BNOF



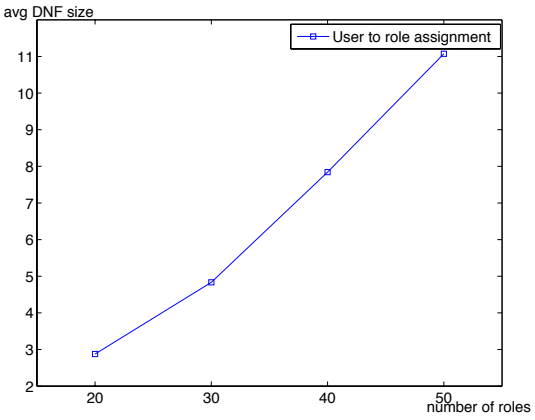
(c) Temps d'exécution pour la comparaison de rôles des séries 1, 2, 3 and 4 avec les rôles obtenus par mining avec FCA



(d) Taille moyenne des DNFs avec FCA



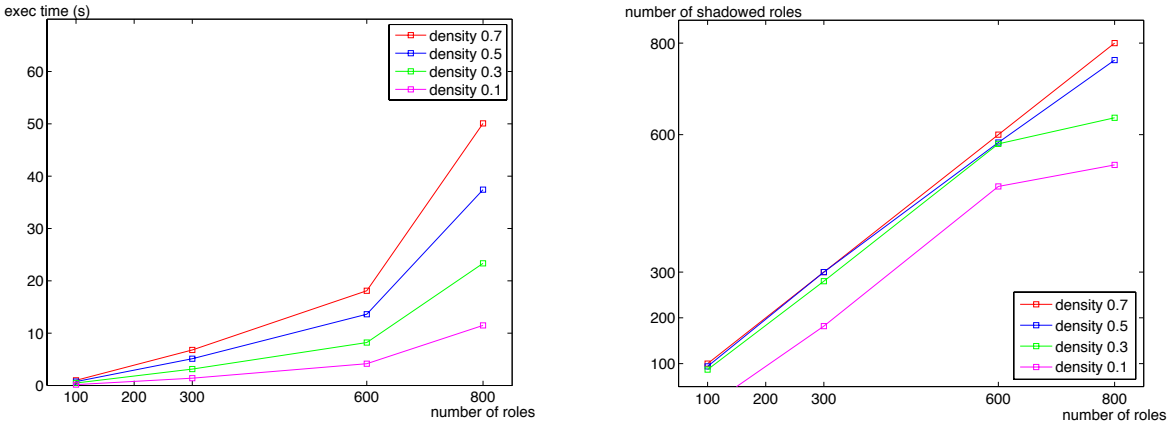
(e) Temps d'exécution de la comparaison de *UPA*avec *RP* dans RBAC des séries de données 5, 6, 7 et 8.



(f) Nombre moyen de rôles affectés à chaque utilisateur

Figure 1.12 – Résultats expérimentaux de la comparaison d'ensembles de roles.

La figure 1.13(b) présente le nombre de rôles masqués détectés par l'algorithme 6 et confirme que le nombre de rôles mal-configurés augmente quand la densité de génération aléatoire augmente, et que le temps d'exécution de l'algorithme va de paire avec lui.



(a) Le temps d'exécution en fonction du nombre de rôles

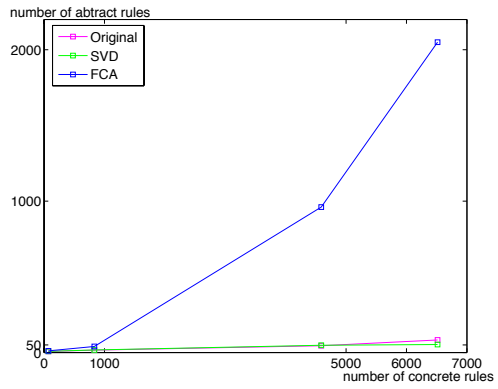
(b) Nombre de rôles masqués dans les séries de données

Figure 1.13 – Résultats expérimentaux avec l'algorithme de détection de rôles masqués

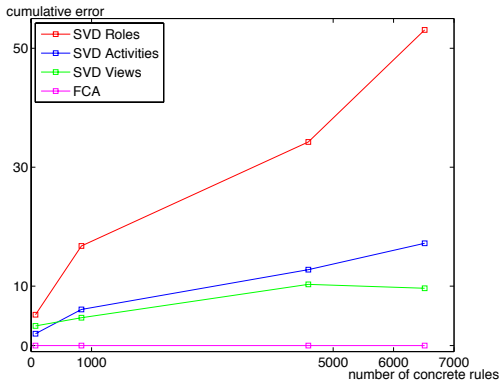
1.8.4 Fouille de règles de sécurité dans un seul pare-feu

Nos tests pour cette partie suivent le scénario suivant : nous générons une configuration Net-RBAC aléatoire, nous calculons la matrice $UO\mathcal{O}$ de cette configuration, nous effectuons le policy mining en utilisant des méthodes de factorisation différentes, et nous comparons la politique Net-RBAC obtenu avec la configuration initiale. Les résultats des tests sont décrits dans la figure 1.14.

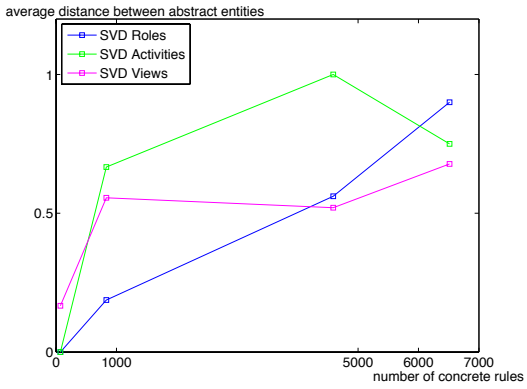
Les résultats montrent que le processus de policy mining donne une politique plus compacte que les règles concrètes initiales (figure 1.14(a)). Les résultats montrent aussi que dans le cas de méthodes de factorisation avec erreur d'approximation, l'erreur n'est pas cumulative avec les trois factorisations succéssives du policy mining (figure 1.14(b)). L'ordre de fouille des trois entités abstraites (rôles, activités, et vues) étant arbitraire, nous avons changé cet ordre et effectué le policy mining. Les résultats visibles dans les figures 1.14(c) and 1.14(d) montrent la distance entre les entités abstraites calculées par policy mining avec un ordre différent.



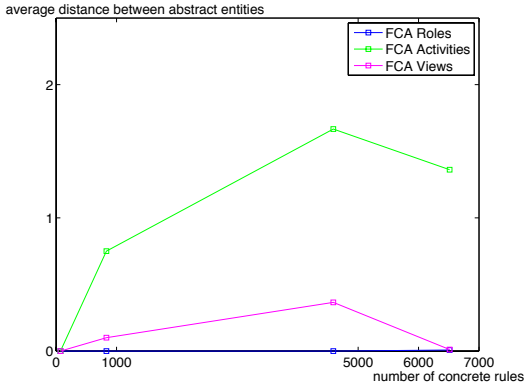
(a) Compacité des règles abstraites obtenues par policy mining vs les règles concrètes originales



(b) Erreur cumulative avec les méthodes de factorisation avec erreur d'approximation



(c) L'effet du changement de l'ordre de mining avec SV des entités abstraites



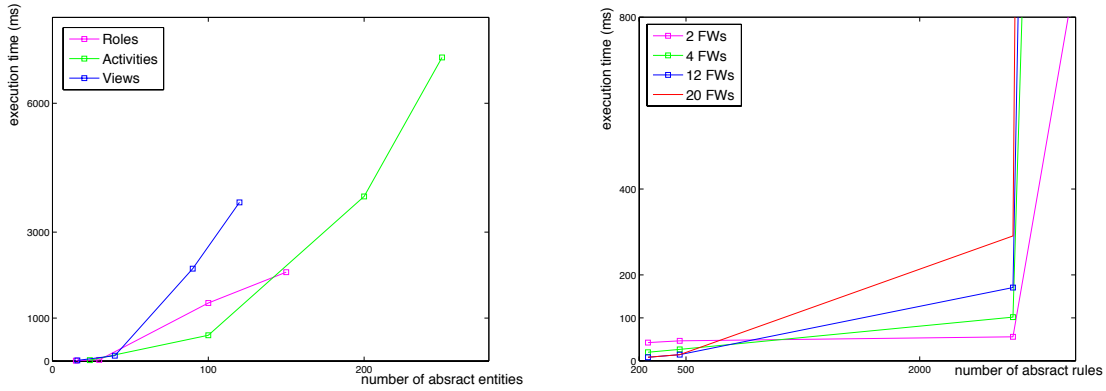
(d) L'effet du changement de l'ordre de mining avec SV des entités abstraites

Figure 1.14 – Résultats expérimentaux avec l'algorithme de Policy Mining.

1.8.5 Fouille de règles de sécurité dans plusieurs pare-feux

Le scénario des expérimentations de cette section est : nous générons une configuration Net-RBAC aléatoire, nous la répartissons sur plusieurs pare-feux, et nous effectuons le policy mining sur les règles concrètes de chaque pare-feu. Ensuite nous exécutons les algorithmes 3 and 2 pour obtenir la politique Net-RBAC correspondante en ingénierie inverse. Nous avons généré quatre séries de donnée avec des tailles croissantes. Pour simuler le déploiement de la politique Net-RBAC sur plusieurs pare-feux, nous avons défini un algorithme qui prend en entrée le nombre souhaité de pare-feux, et la densité souhaitée de chaque pare-feu. L'algorithme choisit aléatoirement les règles à affecter à chaque pare-feu, tout en assurant que les règles soient implémentées sur tous les pare-feux sur le chemin entre le rôle et la vue. Une relation appelée *routes* est générée pour

répartir les pare-feux entre les routes hypothétiques. Nous avons utilisé la méthode de factorisation *SVD* pour le policy mining sur chaque pare-feu. La figure 1.15(a) montre que le temps d'éxécution de l'algorithme 2 croit avec le nombre d'entités abstraites mais reste assez bas. La figure 1.15(b) montre le temps d'exécution de l'algorithme 3 en fonction du nombre de règles abstraites lorsqu'on divise la même politique Net-RBAC sur respectivement 2, 4, 12 et 20 pare-feux. Le temps d'exécution augmente avec ces deux paramètres.



(a) Performance de Abstract Entities Merger en fonction du nombre d'entités abstraites dans le réseau.

(b) Performance de Abstract Rules Merger en fonction du nombre de règles abstraites dans le réseau.

Figure 1.15 – Résultats expérimentaux avec les algorithmes d'intégration de politiques Net-RBAC.

1.8.6 Conclusion

Nous avons implémenté une plateforme de test pour pouvoir étudier de manière expérimentale les différentes solutions proposées dans cette thèse. Nous avons adapté des méthode de factorisations parmi les plus utilisées dans la littérature du role mining. Nous avons développé des générateurs aléatoires de configurations RBAC et Net-RBAC. Nous avons montré aussi comment simuler des topologies réseaux contenant des pare-feux. Finalement, nous avons implémenté tous les algorithmes proposés dans les chapitres précédents. Les résultas expérimentaux sont cohérents avec les études théoriques, et confirment la validité et l'apport bénéfique de notre approche. Les résultats de performance pourraient être améliorés avec une implémentation plus optimisée, et avec des ressources de calculs plus conséquentes.

1.9 Chapitre 9 : Conclusion et perspectives

L'approche présentée dans cette thèse fait partie d'un projet qui vise à trouver des solutions pratiques pour les problèmes d'administration de sécurité des réseaux dans des organisations sensibles tel que les banques. Les systèmes d'information protégés par des pare-feux sont jusqu'à ce jour gérés manuellement. Ce mode de fonctionnement n'est plus viable avec l'augmentation du nombre de règles de sécurité. Notre approche applique des technique de fouille adaptées sur des pare-feux déjà configurés, permettant de détecter les anomalies, et d'extraire une politique à haut niveau d'abstraction modélisé par Net-RBAC. Cette approche permet aussi d'adopter une démarche de définition et de gestion de politiques de contrôle d'accès à un haut niveau d'abstraction, en appliquant un cycle répétitif de processus ascendant et descendant.

Dans cette optique, nous avons d'abord étudié les solutions existantes de gestion de pare-feux dans le chapitre 2. Nous avons montré les bénéfices de l'adoption du modèle Net-RBAC. Dans le chapitre 3, nous avons surveillé la littérature du domaine de role mining. Nous avons montré que cette approche ne peut pas être appliqué directement sur des règles de pare-feu.

Dans le chapitre 4, nous avons défini policy mining, une solution ascendante de fouille de politique modélisée avec Net-RBAC à partir de la configuration d'un pare-feu. Cette solution permet de réutiliser les solutions existantes de role mining. Dans le chapitre 5, nous avons complété le processus de policy mining pour supporter une politique déployée sur plusieurs pare-feux. Nous avons développé une solution pour intégrer plusieurs politiques Net-RBAC, tout en vérifiant les propriétés d'accessibilité et de pertinence de déploiement des règles.

Concernant l'application des techniques de role mining dans le cadre général, nous avons traité deux problématiques importantes qui manquaient dans la littérature. Nous nous sommes d'abord focalisés dans le chapitre 6 sur le problème de comparaison de deux configurations de rôles équivalentes. Nous avons montré que ce problème est NP-Complet, et nous avons proposé un outil d'assistance à l'administrateur de sécurité dans l'exploitation des résultats de role mining. Ensuite, dans le chapitre 7, nous avons traité le problème des rôles masqués. Nous avons établi un lien entre la présence de rôles masqués et la complexité de comparaisons de rôles. Nous avons proposé une solution pour détecter cette anomalie.

Toutes les contributions susmentionnées sont testées et évaluées par étude expérimentale dans le chapitre 8.

Ce travail a plusieurs perspectives intéressantes. Nous projetons d'intégrer les algorithmes présentés dans les outils d'administration de RBAC et de Net-RBAC.

Policy mining pourrait être utilisé pour l'analyse d'autres dispositifs de contrôle d'accès comme les systèmes de détection d'intrusion (IDS).

Policy mining pourrait être utilisé pour examiner les traces de pare-feu.

L'approche de policy mining proposé dans cette thèse est applicable sur les pare-feux sans état. Nous envisageons d'explorer le concept de *contexte* défini dans OrBAC pour administrer des pare-feux à état. La solution d'intégration de politiques Net-RBAC pourrait être étendue pour cette version de Net-RBAC avec contexte.

En plus, dans la phase de pré-traitement, nous avons transformé les règles déployées sur le pare-feu en règles positives (ou permissions) seulement. De point de vue de contrôle d'accès, ceci n'affecte pas la politique déployée. Cependant, les règles négatives (ou interdiction) sont déployées sur les pare-feux pour plusieurs raisons pratiques comme la défense en profondeur et la prévention des attaques d'usurpation d'identité (spoofing). Comme perspective de ce travail, nous prévoyons d'effectuer la même approche ascendante sur les règles négatives seulement afin de donner à un administrateur de sécurité réseau la possibilité d'analyser ces règles.

Au delà de l'application à la sécurité des réseaux, d'autres applications du contrôle d'accès requièrent des règles de sécurité ternaires, et pourraientt donc bénéficier de l'approche ascendante de fouille de règles abstraites présenté dans cette thèse.

Bibliographie

[1] Cisco netsonar. *http ://www.cisco.com/c/en/us/products/security/scanner/literature.html* (arrested development in 2005), 1999. 5

[2] Muhammad Abedin, Syeda Nessa, Latifur Khan, Ehab Al-Shaer, and Mamoun Awad. Analysis of firewall policy rules using traffic mining techniques. *Int. J. Internet Protocol Technology*, 5(1-2), 2010. 6

[3] J. G. Alfaro, N. Boulahia-Cuppens, and F. Cuppens. Complete analysis of configuration rules to guarantee reliable network security policies. *Int. J. Inf. Secur., Springer-Verlag, Berlin, Heidelberg*, 7(2) :1615–5262, 2008. 6

[4] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. Leveraging lattices to improve role mining. In *Proceedings of the 23rd International Information Security Conference IFIP SEC'08*, pages 333–347. Springer, September 2008. 35

[5] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Visual role mining : A picture is worth a thousand roles. *IEEE Transactions on Knowledge and Data Engineering*, 2011. 12

[6] Frédéric Cuppens, Nora Cuppens-Boulahia, and Joaquin Garcia. Misconfiguration management of network security components. In *IASTED International Conference on Communication, Network,and Information Security (CNIS 2005)*, Phoenix, USA, November 2005. 6

[7] Frédéric Cuppens, Nora Cuppens-Boulahia, Thierry Sans, and Alexandre Miège. A formal approach to specify and deploy a network security policy. In Theodosis Dimitrakos and Fabio Martinelli, editors, *Formal Aspects in Security and Trust*, pages 203–218. Springer, 2004. 6

[8] D. Brent Chapman Elizabeth D. Zwicky, Simon Cooper. *Building Internet Firewalls, 2nd Edition*. O'Reilly Media, june 2000. 22

[9] Alina Ene, William Horne, Nikola Milosavljevic, Prasad Rao, Robert Schreiber, and Robert E. Tarjan. Fast exact and heuristic methods for role minimization problems. In *Proceedings of the 13th ACM symposium on Access control models and technologies SACMAT'08*, pages 1–10. ACM, June 2008. 10, 12, 14

[10] Dan Farmer and Wietse Venema. Improving the security of your site by breaking into it. 5

[11] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed *NIST* standard for role-based access control. standard, NIST, 2001. 7

[12] Mario Frank, Joachim M. Buhmann, and David Basin. On the definition of role mining. In *Proceeding of the 15th ACM symposium on Access control models and technologies SACMAT '10*, pages 35–44. ACM, June 2010. 9

[13] Mario Frank, Andreas P. Streich, David A. Basin, and Joachim M. Buhmann. A probabilistic approach to hybrid role mining. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proceedings of the 16th ACM Conference on Computer and Communications Security CCS'09*, pages 101–111. ACM, November 2009. 10

[14] Martin Freiss. *Protecting Networks with SATAN*. O Reilly Media, May 1998. 5

[15] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999. 35

[16] Anas Abou El Kalam, Salem Benferhat, Alexandre Miege, Rania El Baida, Frédéric Cuppens, Claire Saurel, Philippe Balbiani, Yves Deswarte, and Gilles Trouessin. Organization based access control. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), 4-6 June 2003, Lake Como, Italy*, number 0-7695-1933-4, pages 120 –. IEEE Computer Society, 2003. 6

[17] Virginia C . Klema and Alan J. Laub. The singular value decomposition : Its computation and some applications. *IEEE Transactions on Automatic Control*, 25(2) :164–176, April 1980. 11, 35

[18] Mehmet Koyuturk, Aaanth Grama, and Naren Ramakrishnan. Nonorthogonal decomposition of binary matrices for bounded-error data compression and analysis. *ACM Transactions on Mathematical Software*, 32(1) :33–69, March 2006. 11, 35

[19] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Neural Information Processing Systems (NIPS)*, pages 556–562, 2000. 35

[20] Joachim M. Buhmann Mario Frank, David Basin. A class of probabilistic models for role engineering. In *Proceedings of the 15th ACM Computer and Communications Security Conference CCS'08*, pages 27–31. ACM, October 2008. 12

[21] Robert M. Marmorstein and Phil Kearns. Firewall analysis with policy-based host classification. In *Proceedings of the 20th conference on Large Installation System Administration LISA'06*, pages 41–51, Berkeley, CA, USA, December 2006. USENIX Association. 6

[22] Alain Mayer, Avishai Wool, and Elisha Ziskind. Fang : A firewall analysis engine. IEEE, 2000. 5

[23] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. Mining roles with semantic meanings. In *Proceedings of the 13th ACM symposium on Access control models and technologies SACMAT '08*, pages 21–30. ACM, June 2008. 11

[24] Ian Molloy, Ninghui Li, Yuan (Alan) Qi, Jorge Lobo, and Luke Dickens. Mining roles with noisy data. In *Proceedings of the 15th ACM symposium on Access control models and technologies SACMAT '10*, pages 45–54. ACM, June 2010. 10

[25] Ralf Nikolaus. Learning the parts of objects using non-negative matrix factorization (nmf), February 2007. 11

[26] Stere Preda, Nora Cuppens-Boulahia, Frédéric Cuppens, Joaquín García-Alfaro, and Laurent Toutain. Model-driven security policy deployment : Property oriented approach. In Fabio Massacci, Dan S. Wallach, and Nicola Zannone, editors, *International Symposium on Engineering Secure Software and Systems ESSoS 2010*, volume 5965, pages 123–139. Springer, February 2010. 6

[27] Stere Preda, Nora Cuppens-Boulahia, Frédéric Cuppens, and Laurent Toutain. Architecture-aware adaptive deployment of contextual security policies. In *ARES 2010 : Fifth International Conference on Availability, Reliability and Security*, Krakow, Poland, february 2010. 6

[28] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control model. *IEEE Computer*, 29(2) :38–47, February 1996. 7

[29] Romuald Thion. Découverte automatisée de hiérarchies de rôles pour les politiques de contrôle d'accès. In *INFORSID*, pages 139–154, Mai 2007. 11

[30] Alok Tongaonkar, Niranjan Inamdar, and R. Sekar. Inferring higher level policies from firewall rules. In *Proceedings of the 21st Large Installation System Administration Conference LISA '07*, November 2007. 6, 16

[31] Alok S. Tongaonkar. Fast pattern-matching techniques for packet filtering. The graduate school in partial fulfillment of the requirements for the degree of master of science in computer science, Stony Brook University, May 2004. 15

[32] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi 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*, pages 175–184. ACM, June 2007. 11, 12

[33] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Nabil 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*, pages 11–20. ACM, June 2008. 11

[34] Lihua Yuan and Hao Chen. Fireman : A toolkit for firewall modeling and analysis. In *In Proceedings of IEEE Symposium on Security and Privacy*, pages 199–213, 2006. 6

[35] Dana Zhang, Kotagiri Ramamohanarao, and Tim Ebringer. Role engineering using graph optimisation. In *Proceedings of the 12th ACM symposium on Access control models and technologies SACMAT '07*, pages 139–144. ACM, June 2007. 11, 12

Résumé

Cette thèse est consacrée à une approche ascendante pour l'administration de la sécurité des réseaux informatiques d'un haut niveau d'abstraction. Nous montrons que le modèle Net-RBAC (Network Role Based Access Control) s'adapte à la spécification des politiques de contrôle d'accès des réseaux. Nous proposons une approche ascendante et automatique baptisée policy mining qui extrait la politique modélisée par Net-RBAC à partir des règles de sécurité déployées sur un pare-feu. Notre algorithme basé sur la factorisation matricielle est capable d'adapter la plupart des techniques de role mining existantes. Comme les réseaux des entreprises sont souvent protégés par plusieurs pare-feu, nous traitons le problème d'intégration de politiques résultant de l'application de policy mining sur chaque pare-feu. Nous vérifions les propriétés de sécurité reliées à la cohérence du déploiement de la politique sur plusieurs dispositifs. En outre, nous proposons des outils qui assistent un administrateur dans l'analyse des résultats du role mining et du policy mining. Nous formalisons le problème de comparaison de deux configurations de rôles et prouvons qu'il est NP-Complet. Nous définissons un algorithme qui projette les rôles d'une configuration dans une autre en se basant sur des expressions Booléennes. Nous soulignons que la présence de rôles ombragés dans une configuration se manifeste par une augmentation de la complexité de la comparaison avec une autre configuration et présentons une solution pour détecter différentes anomalies d'ombrage. Chaque contribution se base sur un cadre théorique solide, est illustrée par des exemples réels, et appuyée par des résultats expérimentaux.

Mots clés : Sécurité des réseaux, Contrôle d'accès, Modèle Net-RBAC, Fouille de données, Logique Booléenne

Abstract

This thesis is devoted to a bottom-up approach for the management of network security policies from high abstraction level with low cost and high confidence. We show that the Network Role Based Access Control (Net-RBAC) model is adapted to the specification of network access control policies. We propose policy mining, a bottom-up approach that extracts from the deployed rules on a firewall the corresponding policy modeled with Net-RBAC. We devise a generic algorithm based on matrix factorization, that could adapt most of the existing role mining techniques to extract instances of Net-RBAC. Furthermore, knowing that the large and medium networks are usually protected by multiple firewalls, we handle the problem of integration of Net-RBAC policies resulting from policy mining over several firewalls. We demonstrate how to verify security properties related to the deployment consistency over the firewalls. Besides, we provide assistance tools for administrators to analyze role mining and policy mining results as well. We formally define the problem of comparing sets of roles and evidence that it is NP-complete. We devise an algorithm that projects roles from one set into the other set based on Boolean expressions. This approach is useful to measure how comparable the two configurations of roles are, and to interpret each role. Emphasis on the presence of shadowed roles in the role configuration will be put as it increases the time complexity of sets of roles comparison. We provide a solution to detect different cases of role shadowing. Each of the above contributions is rooted on a sound theoretical framework, illustrated by real data examples, and supported by experiments.

Key words: Network security, Access Control, Net-RBAC, Data Mining, Boolean Logic