Application of Risk Metrics for Role Mining

Sharmin Ahmed, *The University of Western Ontario*

Supervisor: Dr. Sylvia Osborn, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Computer Science

APPLICATION OF RISK METRICS FOR ROLE MINING

(Thesis format: Monograph)

by

Sharmin Ahmed

Graduate Program in Computer Science

A thesis submitted in partial fulfillment

of the requirements for the degree of

Doctorate of Philosophy

The School of Graduate and Postdoctoral Studies

The University of Western Ontario

London, Ontario, Canada

Abstract

Incorporating risk consideration in access control systems has recently become a popular research topic. Related to this is risk awareness which is needed to enable access control in an agile and dynamic way. While risk awareness is probably known for an established access control system, being aware of risk even before the access control system is defined can mean identification of users and permissions that are most likely to lead to dangerous or error-prone situations from an administration point of view. Having this information available during the role engineering phase allows data analysts and role engineers to highlight potentially risky users and permissions likely to be misused. While there has been much recent work on role mining, there has been little consideration of risk during the process.

In this thesis, we propose to add risk awareness to role mining. We aggregate the various possible risk factors and categorize them into four general types, which we refer to as risk metrics, in the context of role mining. These risk metrics are Similarity, Compliance to Policies, Trust and Sensitivity and Permission Misuse and Abuse. Next, we propose a framework that incorporates some specific examples of each of these risk metrics before and after role mining. We have implemented a proof-of-concept prototype, a Risk Awareness system for Role Mining (aRARM) based on this framework and applied it to two case studies: a small organizational project and a university database setting. The aRARM prototype is automatically able to detect different types of risk factors when we add different types of noise to this data. The results from the two case studies draw some correlation between the behavior of the different risk factors due to different types and amounts of noise. We also discuss the effect of the different types and amounts of noise on the different role mining algorithms implemented for this study. While the detection rating value for calculating the risk priority number has previously been calculated after role mining, we attempt to find an initial estimate of the detection rating before role mining.

Keywords: access control, security metrics, threats, vulnerabilities, and risk management

Acknowledgements

I would like to sincerely thank my supervisor Prof. Sylvia L. Osborn. It has been an honour to work under her guidance and supervision for the past four years and I am truly indebted for her support, hard work, dedication and encouragement during this time. I would like to thank my advisory committee Prof. Michael A. Bauer and Prof. Robert Mercer for their helpful feedback.

To my parents, mother and father in law, my brothers, my sister in law and other family members, I appreciate the support and encouragement you have given to me. I would like to thank our friends who have always been there for encouragement.

Last but not the least, I would like to thank my husband, Iftekher, and my daughter, Raaha, for all their support and sacrifices.

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

Complex organizations need to establish access control policies in order to manage access to restricted resources. Three of the most popular models of access control include Discretionary Access Control (DAC), Mandatory Access Control(MAC) and Role Based Access Control (RBAC). DAC as defined in [1] is ”a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject”. MAC as defined in [1] is ”a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity”. In MAC individual users have no choice regarding which user is assigned to which permission and is too rigid. However, DAC lets users decide the access control policies on their data but it is a bit too liberal. A more suitable mode of access control that falls between the polar shortcomings of DAC and MAC is RBAC. In RBAC, a set of permissions is collected as roles and then these roles are assigned according to the responsibilities and qualifications of each employee. RBAC is a popular option for medium to large-size organizations [11]. RBAC introduces many different benefits, such as simplified system administration, enhanced system security and integrity, simplified regulatory compliance, and enhanced security policy enforcement [11].

Role-Based Access Control (RBAC) is a nondiscretionary access control mechanism which allows and promotes the central administration of an organization-specific security policy [17]. The NIST standard defines the RBAC model in terms of four model components: Core RBAC, Hierarchical RBAC, Static Separation of Duty Relations, and Dynamic Separation of Duty Relations [18].

Core RBAC includes users (U), roles (ROLES), objects (OBS), operations (OPS), permissions (P) and a set of sessions (SESSIONS). In each session a user may access their assigned roles. A role is primarily a job function within an organization. A permission is an approval of a mode of access to an object [18].

The Hierarchical RBAC component adds relations between roles to create role hierarchies. In a hierarchy, senior roles acquire the permissions of their juniors and junior roles acquire the users of their seniors [18].

Static Separation of Duty (SSD) Relations are used to enforce conflict of interest policies such as enforcing constraints on the assignment of users to roles. For a defined set of roles and a user-role assignment, SSD is defined as a pair $(roleset, n)$, indicating that a user can be assigned to at most $n - 1$ roles from the *roleset*, where n is a natural number ≥ 2 [18].

Like SSD, Dynamic Separation of Duty (DSD) relations also limit the permissions that are available to a user. However DSD relations differ from SSD relations in that constraints, again of the form $(roleset, n)$, indicate that at most $n - 1$ roles from *roleset* can be activated simultaneously in a session, even though all the roles in *roleset* may be assigned to the user. This deals with potential conflict at run time, whereas SSD deals with potential conflict at all times [18].

Formally (from [18]):

- U, $ROLES$, OPS, and OBS refer to users, roles, operations, and objects, respectively.

- $UA \subseteq U \times ROLES$, a many-to-many mapping user-to-role assignment relation

- $P = 2^{(OPS \times OBS)}$, the set of permissions
- $PA \subseteq P \times ROLES$, a many-to-many mapping permission-to-role assignment relation
- $SESSIONS$, the set of sessions
- $assigned\_users(r) = \{u \in U \mid (u, r) \in UA\}$
- $RH \subseteq ROLES \times ROLES$ is a partial order on $ROLES$ called the inheritance relation, written as $\geq$, where $r1 \geq r2$ only if all permissions of $r2$ are also permissions of $r1$, and all users of $r1$ are also users of $r2$.
- $SSD \subseteq (2^{ROLES} \times N)$ is a collection of pairs (rs, n) in Static Separation of Duty, where each rs is a role set, t a subset of roles in rs, and n is a natural number ≥ 2, with the property that no user is assigned to n or more roles from the set rs in each $(rs, n) \in SSD$. Formally: $\forall (rs, n) \in SSD, \forall t \subseteq rs : | t | \geq n \Rightarrow \bigcap_{r \in t} assigned\_users(r) = \emptyset$

Deploying RBAC requires first defining a complete and correct set of roles. In 1995, Edward Coyne proposed the concept of Role Engineering, the goal of which is to determine a set of complete and correct roles for RBAC [15]. Here, complete and correct refers to assigning a user a set of roles that encompasses all the permissions necessary for carrying out tasks and making sure that excess unnecessary permissions are absent. This process has been identified as one of the costliest and most time consuming components in realizing RBAC [56]. To date, various role engineering approaches have been proposed. These methods can be classified as either top-down or bottom-up. In a top-down approach, requirements are gathered from business processes for a particular job function and then permissions are added based on the necessities of the job function. While this method does produce a very accurate set of roles, it is human intensive, expensive and slow. The bottom-up approach requires the presence of an existing set of (user, permission) pairs, and uses these to come up with sets of permissions and defines them as roles.

In 2003, Kuhlmann et al. [28] first linked the bottom-up role-engineering approach with data mining. Thus the concept of role mining was born. Since then, devising a complete and correct set of roles has been recognized as one of the most important and challenging tasks in implementing role-based access control. The problem of role mining, minimizing the number of roles, has been shown to be NP Complete [54]. There are a number of papers on role mining in the literature [38]. These role mining algorithms implicitly assume clean data and use roles to achieve a complete covering of the user-permission relation. In real life, data is not always clean and access control configurations in any large organization might contain undesirable over- and under-assignments as well as exceptions [39]. Work such as [39] and [56] are aimed at cleaning the access control data to return optimal roles. We, instead, look at it from the point of view of associated risk. With the exception of [12], [13] and [6], there is little research that considers risk during the process of role mining. However, the determination of risk during the process of role mining is necessary to highlight potential users and permissions that may be most prone to error or misuse once roles are in operation [12].

Incorporating consideration of risk in access control systems has recently gained the attention of researchers. Also gaining attention is risk awareness. Risk awareness in access control systems is a new but prominent issue as the need for enabling access in an agile and dynamic way has emerged. While there may be risk awareness for an established access control system, being aware of risk even before the access control system is defined could mean identification of users and permissions that represent the most likely dangerous and error prone situations from an administration point of view. Once a role is created, its lifecycle will follow the evolution of the company: new users or new permissions can be added and old ones can be removed or replaced, etc. This continuous modification of the access control system typically introduces noise within the data namely, permissions exceptionally or accidentally granted or denied, thus increasing the risk of making mistakes when managing access to resources.

A role is considered risky if a user assigned to that role, and by exercising all of the permissions available through the role, can cause some harm to the resources of their enterprise, or perpetrate some fraud. It may be that risky roles are left in the system because they are necessary, or because some users are very highly trusted.

In a large organization, assignment to and revocation from roles can be overlooked. Let us take an example from a university setting. Graduate students Tom and Lucas are both teaching assistants for the course CS1032, which always requires two teaching assistants. As teaching assistants, they get to view and assign grades to students of the CS1032 course. Now, in the next semester, Lucas is again a teaching assistant but Tom is not. However, Tom's permission to view and assign grades to CS1032 students have not been revoked and he can still view and assign grades.

- As Tom is no longer a teaching assistant, he is getting access to something beyond his security level. This is an example of a breach in security.
- If Tom uses his privilege rights to assign grade to a student he knows well in CS 1032, then he is already abusing his permission.

Now let us assume that Sara is actually the new teaching assistant for the course CS1032 along with Lucas. However, Sara does not get access to the course CS 1032 because Tom still has access to it. Also, as a teaching assistant, Sara alone somehow gets access to viewing and assigning grades to CS1033 in which no instructor has yet been specified. So now this time,

- Sara does not get the target privileges necessary for her task.
- Sara gets access to something unlike anyone else and she is an outlier.

Having risk-based information available during the role engineering phase allows data analysts and role engineers to highlight users and permissions that are likely to be risky when designed roles are operating.

In this thesis, we propose to add risk awareness to role mining. Our goal is to help role engineers to produce a role hierarchy with a minimum of risk. To accomplish this, we aggregate the various possible risk factors and categorize them into four general types (similarity, compliance to policies, trust and sensitivity, and permission misuse and abuse), which we refer to as risk metrics, in the context of role mining. We propose a framework that incorporates some specific examples of each of these risk metrics before and after role mining. This framework explains exactly how the different risk metrics can be applied before and after role mining. Our philosophy is to identify risks, and allow the role engineer to remove them, or leave them if it is something that they are prepared to tolerate. We have implemented a proof-of-concept prototype, a Risk Awareness system for Role Mining (aRARM) based on this framework. The aRARM prototype is automatically able to detect different types of risk factors (number of outliers, number of security breaches, number of target privileges not met and number of violated constraints) when we add different types of noise to this data, thus semi-automatically being risk-aware even before the roles have been deployed. The prototype is then applied to two case studies: a small organizational project and a university database setting. The results from the two case studies show some correlation between the behavior of the different risk factors and different types and amounts of noise. We draw some hypotheses based on these behaviors. As the tool provides the flexibility to use any role mining algorithms, we also discuss the effect of the different types and amounts of noise on the four role mining algorithms (CompleteMiner [57], EdgeMinimization [16], Graph Optimization [63] and OsbornReidWesson described in [45]) implemented for this study. To determine possible permission misuse and abuse, we use the risk priority number formula as discussed in [5]. While the detection rating value for calculating the risk priority number has previously been calculated after role mining, we attempt to find an initial assumption of the detection rating before role mining.

1.1 Originality of Research

To the best of our knowledge, there are currently no studies that propose to add risk awareness to role mining by aggregating the various possible risk factors and categorizing them into four general risk metrics. There has been much work on risk awareness in access control models and in RBAC. None of this work is specific to risk awareness during the role engineering phase. Next if we consider the role mining literature, in particular in the role engineering literature, the work of Colantonio et al. in [12] and [13] and the work in [6] are the only one that comes close to ours. In [12], the authors propose a framework to evaluate risk by highlighting users and permissions that deviate from others and are outliers. In [13], Colantonio et al. also propose two risk metrics during role mining. In the first metric, referred to as similarity, values are proportional to the number of permissions a given set of users share. The second, minability, measures the complexity of selecting candidate roles given a set of user-permission assignments. The work in [6] visualizes an aggregation of the user access risk level of all the different assignments. In our work we consider the several risk factors that are applicable before and after role mining, based on literature survey, and then apply it to a framework to show how different risk factors can be detected before and after role mining during the role engineering phase and then provide an example through a proof of concept prototype. We propose risk metrics applicable before and after role mining. We propose a framework that detects the risk factors before and after role mining, implement a proof of concept prototype tool as an example implementation of our framework and apply it to two case studies. This thesis aims to show that dealing with risk awareness before and after role mining is possible and that such a framework will be useful in the minimization of risk.

1.2 Thesis Organization

The rest of the thesis is organized as follows. In the next Chapter we discuss the background on role mining, all the different researched areas related to role mining, some of the key works on risk awareness in access control and finally the research gap that motivates our work. In Chapter 3 we discuss the different risk metrics in the context of role mining. These include Similarity, Compliance to Policies, Trust and Sensitivity and Permission Misuse and Abuse. In

Chapter 4 we discuss the framework for the risk metrics that we discussed in Chapter 3. In this Chapter we discuss the structure of the framework and the several phases in the framework that include pre and post role mining stages. Chapter 5 presents our proof of concept prototype tool and its scope and design. In Chapter 6, we discuss the two case studies and the data analysis methods and then in the 7th Chapter we discuss the results and their interpretations. In the results and interpretation section we discuss the results of the two case studies separately. Then we combine and discuss the performance by looking at both the case studies together. We then discuss the performance of the case studies with respect to different role mining algorithms. This is followed by a discussion of how we determine the detection rating value before the role mining process and explain it using an algorithm. Finally we conclude the thesis by discussing our contributions, the practical significance of our research and future work.

Chapter 2

Background

In 1995, Edward Coyne proposed the concept of Role Engineering, the goal of which is to determine a set of complete and correct roles for role-based access control [15]. The various role engineering approaches can be classified as either top-down or bottom-up. In a top-down approach, requirements are gathered from businesses processes for a particular job function and then permissions are added based on the necessities of the job function. The bottom-up approach requires the presence of an existing set of user-permission pairs and utilizes these to come up with sets of permissions and define them as roles.

In 2003, Kuhlmann et al. [28] first linked the bottom-up role-engineering approach with data mining. Thus the concept of role mining was born. Since then, the concept of role-mining has gained considerable interest and has become popular as the preferred method of role engineering.

Since the emergence of the concept, there has been a lot of work in role mining. It can be roughly categorized into the following types (each of these will be discussed in subsequent subsections):

1. Definitions for Role Mining
2. Algorithms for Bottom-Up Role Mining

3. Algorithms with Optimizing Heuristics as an Improvement to a previously proposed method
4. Metrics for adding Business Attributes during the process of Role Mining to get Meaningful Roles
5. Algorithms to generate Updated Roles that are as close as possible to Previously Deployed Roles
6. Visual Elicitation of Roles
7. Addition of special Attributes to Role Mining Algorithms
    (a) Role Mining Algorithms with Cardinality Constraints
    (b) Role Mining Algorithms to deal with Noisy Data
    (c) Role Mining Algorithms with Separation of Duty Constraints
8. Miscellaneous Topics
    (a) Usage of Access Logs to generate meaningful roles during Role Mining
    (b) Methodology to reduce the Computational Workload of Large Databases
    (c) Application of Probability for Role Mining
    (d) Evaluation of Different Role Mining Algorithms based on a Quality Metric
    (e) Comparison of Public and Client Data to be used for Role Mining

2.1 Definitions of Role Mining

In 2007, Vaidya et al. [54] proposed the Role Mining Problem (RMP). The formal definition of the RMP is:

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 UA, and a role-to-permission assignment PA δ-consistent with UPA and minimizing the number of roles, k.* [54]

Here, δ-consistency bounds the degree of difference between the user-to-role assignment UA, role-to-permission assignment PA, and user-to-permission assignment UPA. For UA, PA, and UPA to be δ-consistent, the user-permission matrix generated from UA and PA should be within δ of UPA. [54]

Two other variants of the role mining problem were also defined. These are the δ-approx RMP and the MinNoise RMP.

The δ-approx RMP was introduced to bound the degree of approximation of the roles. The δ-approx RMP is formally defined as:

δ-approx RMP: *Given a set of users U, a set of permissions PRMS, a user-permission assignment UPA, and a threshold δ, find a set of roles, $ROLES$, a user-to-role assignment UA, and a role-to-permission assignment PA, δ-consistent with UPA and minimizing the number of roles, k.* [54]

The minimal noise RMP was introduced to return a set of roles such that the approximation is minimum. The minimal noise RMP is formally defined as:

Minimal Noise RMP: *Given a set of users U, a set of permissions $PRMS$, a user-permission assignment UPA, and the number of roles k, find a set of k roles, $ROLES$, a user-to-role assignment UA, and a role to-permission assignment PA, minimizing* $||M(UA) \otimes M(PA) - M(UPA)||$, [54] where $M(UA)$, $M(PA)$, and $M(UPA)$ denote the matrix representation of UA, PA and UPA respectively.

The work in this paper also proved that the RMP and all its variants are NP-complete decision problems.

However, in [19], Frank, Buhmann and Basin conducted an analysis of the existing definitions, algorithms, and assessment methods for role mining and inferred that there is a lack of consensus on goals leading to very different approaches to the role mining problem. The authors state that existing definitions fail to account for some of role mining's practical requirements. Thus they provided a definition for the role mining problem as follows:

Inference RMP: *Let a set of users USERS, a set of permissions PRMS, a user-permission relation UPA, and, optionally, part of the top-down information TDI be given. Under the assumption that exceptions and an underlying RBAC configuration exists and that the TDI infuences the RBAC configuration, infer the unknown RBAC configuration RC=(ROLES, UA, PA).* [19]

2.2 Algorithms for Bottom Up Role Mining

The work in [45] by Osborn, Reid and Wesson is primarily based on the interfacing between RBAC and relational databases but a basic role mining algorithm is given as well. In this paper, a user and all their privileges are depicted as a role. Identical priviledge sets are collapsed into a single role and then a role graph is generated by comparing the priviledge sets for subsetting relationships.

In [47], Schlegelmilch and Steffens discuss the ORCA role mining tool and its algorithm. In their algorithm, roles are generated and formed into a hierarchy. Initially, separate clusters are formed out of each permission. Clusters of permissions are merged and the cluster with the highest number of users in common is returned. More permissions are added to this cluster and the process is repeated until there are no more new clusters found. The problem with this algorithm is that there can be no overlapping of permissions among roles that are not hierarchically related. This significant drawback was criticized by many other works in later years such as in [33], [36] and [57].

In a 2006 paper, Vaidya et al. [57] propose the Roleminer Algorithm which uses subset enumeration. The algorithm, unlike ORCA, allows overlapping of permissions in roles. However, the output is a set of roles instead of a role hierarchy. The authors proposed two versions of the algorithm, the CompleteMiner and the FastMiner. CompleteMiner starts with every user's permission set, and computes the intersection of all these initial roles. Even though FastMiner is very similar to CompleteMiner, it is introduced because it significantly reduces the computational complexity of CompleteMiner by computing the set of candidate roles as all possible intersections of at most two initial roles.

Molloy et al. [36], propose an algorithm for role mining and also propose the Weighted Structural Complexity (WSC) as a metric for the measurement of how good the roles are. In the algorithm HierarchicalMiner, the Formal Concept Analysis method is used to find an initial set of roles and their hierarchy. Next, a pruning method is used to prune the initial role hierarchy by repeatedly using the Weighted Structural Complexity. Thus an optimized role hierarchy is returned. The authors also propose a WSC metric that takes attributes such as job positions, work departments, and job responsibilities into consideration. Using this metric, they propose an optimization algorithm called Attribute Miner. The weighted structural complexity (WSC) is formally defined as follows:

Weighted Structural Complexity (WSC): *Given $W =< w, w_u, w_p, w_h, w_d >$ where w_r, w_u, w_p, w_h, w_d $\in Q+ \cup \infty$ where $Q+$ is the set of all non-negative rational numbers, the Weighted Structural Complexity (WSC) of an RBAC state γ, which is denoted as $wsc(\gamma, W)$ is: $wsc(\gamma, W) = w_r * |R| + w_u * |UA| + w_p * |PA| + w_h * |t_{reduce}(RH)| + w_d * |DUPA|$ where $|\bullet|$ denotes the size of the set or relation, and $t_{reduce}(RH)$ denotes the transitive reduction of role hierarchy.*

Here, the w values are the different weights assigned to the different variables in the equation and can be assigned based on necessity. For example, if w_r is set to 1, w_u and w_p are set to 0 and w_h and w_d are set to infinity, this forces a flat RBAC such that role inheritance relations cost infinity, direct user permission assignment are forbidden and the minimum number of roles can be found [36].

Weights are also assigned to permissions in [33] by Ma et al. In this paper, the authors discuss the assignment of weights to permissions to emphasize the importance of the permissions. The similarity between both users and permissions is used to compute the weight of permissions. The authors also propose a weighted role mining algorithm to generate roles based on weights.

Colantonio et al. propose a cost-based metric for the determination of quality roles in [10]. This metric would pick roles that are cost effective to maintain. They also propose an algorithm to find roles based on this metric. The cost-based function is represented as:

$f = \alpha|UA| + \beta|PA| + \gamma|ROLES| + \delta\Sigma_{r\epsilon ROLES}c(r)$, where $\alpha,\beta,\gamma,\delta \geq 0$ and the function $c : ROLES \rightarrow R$ expresses an additional cost related to other business information different from $| ROLES |, | UA | and | PA |$.

There are some role mining algorithms that use graph theory. For example, the work in [63] by Zhang, Ramamohanarao, and Ebringer use a graph optimization method for role mining. The algorithm starts with an initial set of roles with the users and their corresponding set of permissions. The algorithm then identifies pairs of roles to analyse and performs merge or split operations if they improve the optimisation metric of the graph. The optimisation metric is $|UA| + |PA| + [ROLES]$, where UA represents the number of user-role assignments and PA represents the number of role-permission assignments and $[ROLES]$ represents the number of roles. The main goal of the algorithm is to iteratively minimize the sum of the number of roles and the number of edges.

Graph theory is used in [16], by Ene et al. In this paper, the authors use the Minimum Biclique Cover problem in graph theory and deduce that the role mining problem can be reduced to the Minimum Biclique Cover problem. They propose two algorithms. In the first algorithm, the user-permission assignment relation is used to return a minimal set of roles. In the second algorithm, the set of roles from the first algorithm is greedily improved using lattice-based post processing until no further improvement is possible.

2.3 Algorithm with Optimizing Heuristics as an Improvement to a Previously Proposed Method

There are works in the role mining literature that are improvements to previously proposed role mining algorithms. Some of these works are discussed below.

Colantonio, Pietro and Ocello [14] propose a formal framework based on a rigorous analysis of identifiable patterns in access permission data. They mention how a lattice of candidate roles can be derived from the Permission Powerset. Data redundancies associated with co-occurrences of permissions among users can be easily identified and eliminated, allowing for increased output quality and reduced processing time during mining. This algorithm is proposed as an improvement to their previously proposed algorithm RBAM (Role-Based Association-rule Mining) in [11].

Lu, Vaidya and Atluri [31] propose a framework for modeling the optimal binary matrix decomposition and its variants using binary integer programming. A greedy algorithm is proposed that is an improvement to their previously proposed algorithm in [57].

Zhang et al. [61] propose a Permission Set Mining heuristic for generating roles that cover the largest portion of user assignments. The proposed approach uses efficient search algorithms to identify candidate roles without the role to permission and user to role constraints inherent in existing approaches. The a priori principle of pattern extraction is used for role extraction to produce a natural ordering on roles to assist the hierarchy construction process.

2.4 Metric for Adding Business Attributes During the Process of Role Mining

There is some work in the role mining literature that discusses the use of business attributes so that the resulting roles are more accurate. Some of these are discussed in this section.

Business information, such as Business Processes and Organization Structure, is used to implement role mining algorithms in the work in [11] by Colantonio et al. The authors show that a role is likely to be meaningful from a business perspective when it involves activities within the same business process or organizational units within the same branch. Centrality indices are used to measure the spreading of a role among business processes or organization structure.

Molloy et al. [36] study the problem of mining roles with user-attribute information. In this paper, they formally define the problem and propose a new WSC metric that takes attributes such as job positions, work departments, and job responsibilities into consideration. Using this metric, they propose an optimization algorithm called Attribute Miner.

Frank et al. [20] provide statistical measures to analyze the relevance of different kinds of business information for defining roles. They also discuss an approach that incorporates relevant business information into a probabilistic model with an associated algorithm for hybrid role mining. This approach yields roles that both explain the given user-permission assignments and are meaningful from the business perspective.

2.5 Algorithms to Generate Updated Roles as Close as Possible to Previously Deployed Roles

In most previous role mining literature, it is assumed that the organization does not have an existing RBAC structure, but that may not always be the case. This section discusses some of the work that takes into consideration an existing RBAC and tries to generate updated roles as close as possible to deployed ones.

Vaidya et al. [55] define the Minimal Perturbation RMP as the problem of discovering an optimal set of roles from existing user permissions that are similar to the currently deployed roles. The minimal perturbation RMP is defined as:

Minimal Perturbation RMP. *Given a set of users U, a set of permissions PRMS, a user-permission assignment UPA, and a deployed set of roles DROLES, find a set of roles ROLES, a* *user-to-role assignment UA, and a role-to-permission assignment PA consistent with UPA such that it minimizes the combination function of the number of roles and the distance between ROLES and DROLES (i.e., minimizing $CF(|ROLES|, d(ROLES, DROLES)))$.*

Here the CF is a combination function defined in [55]. The combination function is defined as follows:

Combination Function: *Given some number of roles k, and a distance score d between two sets of roles, a combination function $CF(k, d) = (1 - w)k + wkd$ where w is a user defined weighting coefficient for the similarity.*

Candidate roles are generated and compared to the deployed roles using the Jaccard Coefficient as a similarity metric and accordingly optimal roles are generated as close as possible to the deployed roles.

Takabi and Joshi [50] formally define the problem of mining a role hierarchy with minimal perturbation as follows:

Given an RBAC state $\gamma = \langle R, UA, PA, RH \rangle$, find a new RBAC state that is consistent with access control configuration $\rho = \langle U, P, UP \rangle$ and is as close as possible to the existing RBAC state.

The paper by Takabi and Joshi introduces two measures: a measure for goodness of an RBAC state and a measure for minimal perturbation. Based on these measures they develop StateMiner, a heuristic solution to find an RBAC state as close as possible to both the deployed RBAC state and the optimal state.

2.6 Visual Elicitation of Roles

Visualization is usually considered useful to interpret information. There are certain works in the role mining literature that apply visualization in role mining.

Zhang et al. [62] propose RoleVAT, a Role engineering tool for the Visual Assessment of user and permission tendencies. RoleVAT produces intensity images that represent the natural groupings of users and permissions. Permission similarity images show role tendencies and user similarity images show possible data partitioning. This visual assessment of permission clusters allows for immediate identification of the practical need for RBAC. The tool helps in determining if roles can be identified, how many roles can be identified and if the user permission data can be partitioned.

The work on visualization in [9] by Colantonio et al. is referred to as visual role mining. Here user-permission assignments are graphically represented to enable quick analysis and elicitation of meaningful roles. They introduce a metric for the quality of the visualization and propose two algorithms: ADVISER and EXTRACT. The former is a heuristic used to best represent the user-permission assignments of a given set of roles. The latter is a fast probabilistic algorithm that, when used in conjunction with ADVISER, allows for a visual elicitation of roles even in the absence of predefined roles.

While visualization is not the prime topic in [53], Vaidya et al. do discuss one of the features of the RoleMiner role engineering tool as visualization. Based on their priority metric, the top cluster roles are graphed and the relationship between these roles are distinguished to form a hierarchy between the roles. Given the results of RoleMiner, the authors determine the hierarchical relationship between the top N clusters, where N is a user parameter. To determine the hierarchical relationships, a start is made by making the largest candidate cluster a top node in the cluster hierarchy. For each next sized cluster, *clus* , all the previous clusters that contain *clus* are found and *clus* is added as a child to a cluster already in the hierarchy only if that cluster does not have a descendent that also contains *clus*. Thus, *clus*, a candidate role, may be a candidate subrole of more than one candidate role. Once the relationship between the candidate clusters is determined, the results are graphed. The cluster labels are the ranked order of the candidate clusters. Alternatively, or in addition, the labels indicate the number of users who were found to be members of the candidate roles or even list these users.

Giblin et al. [22] describe a role engineering platform reflecting the dual importance of top-down and bottom-up data collection and analysis. The tool, IBM Tivoli Role Modeling Assistant (RMoA) imports role, organizational and identity data from multiple sources. Subsequent data analysis employs interactive attribute filters, descriptive statistics and data mining techniques to identify role discriminators and automatically infer roles. The use of histograms, charts and a score indicator enhances the exploration of data and enables the detection of patterns during role design. RMoA is a design-time modeling environment that ultimately produces role definitions which are then consumed by operational components such as provisioning systems.

2.7 Addition of Special Attributes to Role Mining Algorithms

It is assumed that the data for role mining is noise free and does not contain any constraints. But this type of assumption does not reflect real life data. In real life situations, data contains noise and has policy constraints (such as separation of duty constraints, cardinality constraints, etc.) associated with it. This section discusses the previous work that takes these into consideration.

The majority of role mining algorithms assume the provided input data is clean and correct. In reality, however, in any large organization this data is likely to contain noise. Molloy et al. [39] advocate a two-step process for role mining with noisy data: first the data is cleaned, then mined. The authors propose nonbinary matrix decomposition as a solution to cleaning the user-permission relation. Their experiments indicate that singular value decomposition, non-negative matrix factorization, and logistic Principle Component Analysis perform well, while several other techniques, including two state-of-the-art probabilistic algorithms, yield high false negative rates. To evaluate the impact noise has on the candidate roles they introduce a new distance measure between two role sets that avoids the pitfalls of the average maximum Jaccard used previously. Using the distance measure, they found that first cleaning the noisy data produces candidate roles that more closely resemble the candidate roles mined from the clean data than alternatives such as selecting the top roles or allowing exceptions. Finally, they found the role minimization objective can reduce the quality of candidate roles, causing them to overfit the noisy data.

Another work that discusses noise during the role mining process is [56] by Vaidya et al. This paper discusses the many problems noise can cause if it is not accounted for within the role mining process. They experimentally show that the algorithm developed for δ-approx RMP in [54] can be used to reduce the effect of noise. Their experimental results indicate that if one sets the level of δ equal to the level of noise, the impact of noise is minimal in discovering roles. The presence of noise essentially means that errors have occurred in the data and the paper categorizes these errors into three types: General noise, Additive noise and Subtractive noise. Preliminary experiments show that the algorithm developed for δ-approx RMP does indeed reduce the effect of noise. However, it is only able to handle additive noise.

Now, if we look into the literature on dealing with constraints, in [52], Uzun et al. use an Extended Boolean Matrix Decomposition (EBMD) to consider separation of duty constraints (SOD) and exceptions by introducing negative assignments. They define the Extended Role Mining Problem and its variants and present their optimization models. The paper also proposes a heuristic algorithm that is capable of utilizing these models to find good decompositions.

Another interesting work on constraints during role mining is [34], where Ma et al. propose constraint mining in migrating a non-RBAC system to an RBAC system. Constraints are defined and then a relationship is created between the conventional data mining technology and the constraints. The authors define a variety of constraints and also propose an anti- association rule mining algorithm to generate the mutually exclusive permissions. It can find mutually exclusive permissions by just scanning the database once, while the traditional association rule mining method needs to scan the database many times. The authors carry out experiments to illustrate the effectiveness of the proposed techniques. The current implementation for anti-association rule mining generates mutually exclusive permissions that are weighted purely on frequency. Hence, if a permission set is only given together to a small number of users, it may not be identified by the algorithm.

An example of a work that considers role-usage cardinality constraints during role mining is [24] where John et al. propose two approaches. The first approach prioritizes the roles based on their size (i.e., number of permissions within the role) and then limits the number of roles assigned to a user based on the priority order. The second approach chooses roles by iteratively picking the role with the largest number of permissions that are not yet assigned to that user by any other role, and then imposes the role usage cardinality constraint.

Another type of policy, event-driven RBAC policy, is discussed in the context of role mining in [48]. Here, the authors examine the problem of conflict identification and resolution in the context of event-driven RBAC policies and develop an integer programming-based framework to solve the problem. The authors propose a verification framework for detection and resolution of inconsistencies and conflicts in policies modeled through event-driven RBAC, an important subset of generalized temporal RBAC applicable to many domains, such as industrial control systems (e.g., SCADA) and financial workflow management systems. In such systems, occurrence of some events may trigger certain actions that need to be executed by authorized users. The authors define the conflict resolution problem and propose an integer programming-based heuristic. The proposed approach is generic and can be tuned to a variety of optimality measures.

2.8 Miscellaneous Topics

This section surveys some of the miscellaneous work done on role mining.

A probabilistic approach is taken to role mining by Frank et al [21]. They develop a probabilistic model for structured Boolean data, by examining various application-specific noise processes that account for the irregularities in the data and theoretically investigating the relationships among these variants. The experiments show that multiassignment clustering computes more precise parameter estimates than state-of-the art clustering approaches. Their model defines a novel and highly competitive solution to the role mining problem. They apply a Gibbs sampling algorithm in a disjoint decomposition model. The method determines the probability of the assignment of users to a particular role.

Molloy et al. [41] are the first to discuss the mining of roles from usage of permissions. The authors used generative machine learning models such as Latent Dirichlet Allocation (LDA) and Author-Topic Model (ATM) for mining roles from both usage records as well as from entitlements; thus meaningful roles are mined. Given data on usage of entitlements, they build a generative RBAC model, i.e., a model which best explains the observed usage pattern. In the resulting models, users who have the same entitlements but with different usage will, typically, have different roles. Extending the definitions of the core role mining problem, the authors also formally define the generative role mining problem as well as many variants, and provide efficient algorithms for these problems. An important variant is attributive generative models, i.e., models where the roles assigned to a user are causally correlated with user attributes (and their combinations) when such information is available. Besides usability, there are a number of other advantages of generative models. From a security standpoint, they are conservative: policy reflects actual usage, thereby reducing operational risk.

While much of the work on role mining is based on the process, the work in [58] by Verde et al. proposed a six step methodology to effectively reduce the computational workload of large databases when role mining is applied. The authors focus on making the role mining problem tractable by reducing the problem size without sacrificing utility and accuracy. The proposed approach does this by effectively compressing the original access control dataset, analyzing the compressed dataset to identify interesting portions, and reconstructing the portions of the original dataset that are worth investigating. Scalability is assured by targeted partitioning that ensures that the created sub-problems focus only on the interesting portions of the original dataset and are therefore orders of magnitude smaller than the original one. At this point, any existing role mining algorithm can be used to analyze user-permission assignments within these subsets. Thus, the approach is agnostic to the specific role mining methodology used, namely it can be used in conjunction with them to enhance their scalability.

Molloy et al. [38] were the first to classify role mining algorithms into two classes based on their outputs: Class 1 algorithms output a sequence of prioritized roles while Class 2 algorithms output complete RBAC states. Nine role mining algorithms were compared using the WSC metric and the differences in the performance of the different algorithms were discussed. Their results indicate that algorithms that strive to minimize the number of roles often generate RBAC states with a larger number of edges, resulting in increased complexity and likely increased administration costs.

In [40], Molloy et al. discussed the difference in structure and distribution of permissions between public and client datasets. The success of research in analyzing of access control data hinges on the availability of high-quality real-world data. Thus far, little access control data has been released to the public. The authors analyze eight publicly released access control datasets and contrast them with three client policies. The analysis indicates there are many differences in the structure and distribution of permissions between the public and client datasets, including sparseness, permission distributions, and cohesion. The client datasets also revealed a wide range of semantics and granularities of permissions, ranging from application-specific rights to general accounts on systems that could not be observed in the public data due to anonymization. Finally, the distribution of user attributes was analyzed, which the public datasets lack. They found that techniques that work well on some datasets do not work equally well on others

The work by Xu et al. in [60] was the first to describe algorithms for the migration from lower-level access control policy representations, such as access control lists (ACLs) to an RBAC model. The algorithms all begin with a phase that constructs a set of candidate roles. Two strategies are considered for the second phase: start with an empty policy and repeatedly add candidate roles, or start with the entire set of candidate roles and repeatedly remove roles. In experiments with publicly available access control policies, the elimination approach produces better results, and for a policy quality metric that reflects size and interpretability, the elimination algorithm achieves significantly better results than previous work. As an extension to this work, in [59], Xu et al. discuss algorithms that take user and permission attributes into consideration when migrating from access control policy representations to RBAC.

Temporal Role Mining is discussed in [35] by Mitra et al. In this type of role mining a role can be enabled during a certain set of time intervals and remains disabled for the rest of the time. The authors define the conflict resolution problem and propose an integer programming-based heuristic. The proposed approach is generic and can be tuned to a variety of optimality measures. The temporal aspects of roles are captured using several extensions of RBAC such as Temporal-RBAC (TRBAC) proposed in [2] and Generalized Temporal Role-Based Access Control (GTRBAC) proposed in [25]. The set of time intervals during which each role can be enabled is specified in a Role Enabling Base (REB).

2.9 Risk in Access Control

The literature on risk in access control is quite rich and here we discuss some of the work. These can be roughly categorized into papers that discuss risk in access control in general, papers that discuss risk in Role Based Access Control and finally, papers that discuss risk in Role Mining.

In [7], the authors propose an analytics-based method which attempts to ensure that the usage of permissions is consistent with the configured policy and that the configured policy follows the principle of least privilege. The principle of least privilege requires that a user be given no more privilege than necessary to perform a job [17]. The least privilege principle limits the damage that can result from an accident or error. It also reduces the number of potential interactions among privileged programs to the minimum for correct operation, so that unintentional, unwanted, or improper uses of privilege are less likely to occur [17]. The key analytic is to ensure that the group definitions in the policy correspond to actual roles mined from usage. Chari et al. [7] discuss two types of risk to an organization and define them as follows:

- **Definition 1.** *Operational risk is an adverse outcome or harm that is the result of failed* *processes or other activity. Such risks could be the result of misuse or abuse of privileges, intentional or otherwise.*

- **Definition 2.** *Administrative risk is an adverse outcome or harm caused by a discrepancy between an administrator's intention and the policy specification. Such risk may be the result of a misinterpretation or an oversight of the policy language or policy model.*

A framework for risk-based adaptive access control using attribute-based access control is proposed in [26]. The risk-based adaptive access control model adjusts security risk for providing access to resources accounting for operational needs, risk factors and situational factors.

In [8], the authors propose a dynamic, multidecision access control model based on quantified risk estimates and risk tolerance. The risk scale represents the range of quantified risk estimates that is further divided into multiple bands of risk. The quantified risk estimate for any access falls into one of these risk bands. Each band is associated with a decision and an action; the decision, the action and band boundaries are all determined according to risk tolerance and can be changed when risk tolerance changes.

The difference between the traditional constraint-based risk mitigation and the recent quantified risk-aware approaches in RBAC is identified in [4]. The authors propose a framework for introducing risk awareness in RBAC models that incorporates quantified risk. Also in [3], the authors incorporate risk in RBAC sessions and propose an extension to the core RBAC model by incorporating risk awareness in sessions where the risk is bounded by a session-based risk threshold.

In [23], Huang et al. discuss the least privilege principle and how it is a threat issue if users get access to more permissions than they are normally entitled to. They propose two greedy approximation algorithms for the least privilege problem during role mining. One of these considers static separation of duty constraints while the other does not.

The authors of [6] describe a system that implements a general model of risk based on any arbitrary set of properties of permissions and users. They use a fuzzy inferencing system to aggregate sensitivity of a group of permissions given the notional sensitivity levels of individual permissions or the aggregate user access risk level and infer the risk of this assignment.

In [46], the authors, using insights from the insurance literature in economics, propose a formal model where the value of resources is explicitly defined and an RBAC policy is only used as a reference point to determine the price each user has to pay for access, as opposed to representing rules that are always rigidly applied.

The work in [37] comes close to the work above. They propose the Quantified Risk Based Access Control System that evaluates the request for an access and quantifies the risk associated with the access. The system then provides the user with an access ticket describing the access, and indicating the request price in risk tokens. The user evaluates the price; if they have enough risk tokens and agree to the price, they will purchase access to the information from the database which will return the information requested in the Access Ticket.

Colantonio et al. propose a framework to evaluate the risk incurred when managing users and permissions through RBAC in [12]. The proposed approach highlights users and permissions that markedly deviate from others, and that might consequently be prone to error when roles are operating. They introduce the stability concept that states that if an assignment can only be managed by roles with few users and permissions, it represents an outlier. Also, the continuous modification of the access control system typically introduces noise within the data, namely, permissions exceptionally or accidentally granted or denied, thus increasing the risk of making mistakes when managing the access control system. Since these permissions are errors and exceptions, they can be tracked as outliers.

Colantonio et al. also propose two risk metrics during role mining in [13]. In the first metric, referred to as similarity, values are proportional to the number of permissions a given set of users share. The second, minability, measures the complexity of selecting candidate roles given a set of user-permission assignments.

2.10 Research Gap

As discussed in the previous section, we see that there are only a few examples of work in role mining literature on risk, specifically, [6], [12] and [13]. In [12], an attempt is made to determine risk before roles have been mined. However, risks may still remain after roles have been mined. In [13], the authors propose two metrics for the determination of risks. However, other risk metrics such as compliance to different security policies and abuse of existing valid permissions are not taken into consideration. In [6], the authors only consider user security levels and do not consider other risk factors. Our work is specific to the fact that we aggregate the various possible risk factors proposed in the literature and based on their applicability during the role engineering phase, categorize them into four general categories. To show how these risk metrics can be determined before and after role mining, we propose a framework that incorporates specific examples of each of these risk metrics both before and after role mining. We also show that risk awareness based on these risk metrics is possible during the role engineering phase by implementing a proof of concept prototype tool that dynamically detects these risk factors before and after role mining.

Chapter 3

Risk Metrics

In this Chapter, we discuss the four general categories of risk factors (henceforth referred to as risk metrics) and provide some examples of risk factors that fall under each category.

3.1 Similarity

The first risk metric that we propose is similarity. The similarity metric was also defined in [12] as the number of permissions that a given set of users share. The similarity metric was also indirectly used in [13] as the stability of a role based on its weight. Here, the weight refers to a function of the role composed of a product of the users and permissions associated with the role. The authors propose that if an assignment can only be managed by roles with a limited weight, it represents an outlier, as there are obviously few users and permissions associated with it. Also, the continuous modification of the access control system typically introduces noise within the data, namely, permissions exceptionally or accidentally granted or denied, thus increasing the risk of making mistakes when managing the access control system. Since these permissions are errors and exceptions, they can be tracked as outliers.

The stability concept is used to find the ranking of users and permissions, highlighting those that most deviate from others in comparison to available user-permission relationships. While these outliers may not be an error, from the system administrator's point of view, they are riskier than others because the risk of making mistakes when managing roles with a limited weight is higher: they are roles that are not used frequently, and are in some way obscure to administrators [13].

The similarity metric can be easily determined using the Jaccard Coefficient [13]. The Jaccard coefficient is a well known statistic used for comparing the similarity and diversity of sample sets. Specifically, the Jaccard coefficient (measure of similarity) and the Jaccard distance (measure of dissimilarity) are measurements of asymmetric information on binary (and nonbinary) variables. For non-binary data, the Jaccard coefficient (JC) for sample sets A and B can be computed as follows:

$$JC = \frac{|A \cap B|}{|A \cup B|}$$

The Jaccard coefficient has been used in various forms in the role mining literature. In [55], Vaidya et al. use it in their minimum perturbation problem to find the similarity between deployed roles and the optimal set of roles after role mining.

The Jaccard Coefficient has been used as a metric in quite a few works in the literature; these are mentioned below:

- The Jaccard coefficient is used as a metric in the evaluation of role mining algorithms by comparing the output roles from the role mining algorithms with the original roles in [38]

- Huang et al. in [23] use the Jaccard coefficient to measure the distance between the original role-set and the role-set generated from their approximate algorithm.

- Colantonio et al. use the Jaccard coefficient in [13] to find the similarity between two users by comparing the similarity between their permission sets. We will be using this similarity index to find the similarity between users and then visualize the similarity between the users to highlight potential outliers.

- Molloy et al. [39] use a maximum bipartite matching algorithm to find the similarity between two sets of roles and use the Jaccard coefficient to calculate the weight along each path. In the case of the availability of an existing RBAC system as input, this bipartite matching algorithm can be used to find the similarity between the existing and optimal set of roles.

Outliers are hard to detect without the use of a proper visualization method. Visualization has already been proposed in the role mining literature ([62] and [9]) as an effective means to highlight relevant patterns and isolate outliers leveraging cognition capabilities. We believe that the visualization of the similarity metric is an effective method to highlight potentially risky users.

3.2 Compliance to Policies

We saw in the background Chapter that there are two types of risk to an organization and the second definition refers to administrative risk. A central part of administrative risk is a discrepancy between an administrator's intention and the policy specification which can result from a misinterpretation or an oversight of the policy language or policy model [7]. For example, suppose a user is granted access to permissions $p1$ and $p2$ and then later on the access needs to be revoked based on a policy. Now suppose the administrator revokes access to only one of the permissions then the user still retains access to one of the permissions. This is an error in the data and a risky situation. Thus we will be considering it as a metric for risk. Policy is a very broad term that encompasses a very wide range. In the context of role mining, we will discuss two examples of applicable policies, namely separation of duty constraints and principle of least privilege.

Separation of duty in itself is also a broad topic that includes static and dynamic separation of duties. As the name suggests, static separation of duty policies are predefined in the context of a particular organization. We have already defined static separation of duty as a pair

$(roleset, n)$ where no user is assigned to n or more roles from the role set, where n is a natural number ≥ 2. [18]. It is important to mention that throughout this thesis, we consider this n to be equivalent to 2 and thus we will be looking at static separation of duty policies between pairs only. In [29] it is proved that 2-role conflicts can fully capture k-role conflicts.

There is a static separation of duty constraint between two roles if no one user is ever allowed to be assigned to both of these roles. In other words, the two roles have no shared assigned users. These policies do not change and must be strictly adhered to. Though it has the advantage of simplicity, static separation of duty is often too rigid to give satisfactory control.

Dynamic separation of duty was proposed as a suitable alternative in situations where static separation of duty seems too rigid. There are many different types of dynamic separation of duty and many variations have been defined. In layman's terms, dynamic separation of duty occurs when restricted roles may have common members, but users may not assume both roles at the same time in a session. For example, suppose user $U1$ has access to both table $T1$ in the form of a role $R1$ and $T2$ in the form of a role $R2$ but is not allowed to access them at the same time. Now, when user $U1$ logs in and clicks on table $T1$ they should be shown some of the contents of the table. However, now if user $U1$ clicks on $T2$ it should give an error and not show the contents.

In [42], the authors discuss different categories of RBAC constraints which we will not be discussing here further. There are quite a few papers, such as [4], [48], etc. that discuss the risk awareness with regards to adherence to separation of duty constraints, be it static or dynamic.

In [44], Nyanchama and Osborn provide a taxonomy for different kinds of separation of duties and discuss them with respect to the role graph model and how it can be applied to an RBAC model. The following kinds of conflicts are a part of what is discussed in the paper:

- User-User conflicts. User-User conflicts arise if two users together should never be assigned to something in particular such as the same privilege or the same role.

- Role-Role conflicts. Role-Role conflict means that two roles must never occur together. Thus a user must not have two roles that are in a Role-Role conflict. A Role-Role conflict corresponds exactly to a static separation of duty.

- Privilege-Privilege conflicts. Privilege-Privilege conflict means that two privileges must not appear together. This means that these two privileges should never be in the same role.

- User-Role assignment conflicts. A User-Role assignment conflict can be either static or dynamic. In a static User-Role assignment conflict between user *u* and role *r*, user *u* may never be assigned to role *r*. In a dynamic User-Role assignment conflict between user *u* and role *r*, user *u* may get some session-based access to role *r* under certain conditions.

- Role-Privilege assignment conflicts. A Role-Privilege assignment conflict can also be either static or dynamic. A static conflict says that a certain privilege should never be assigned to a certain role. Dynamic Role Privilege conflicts could specify that two operations on the same object must not appear in one role, but might appear in separate roles.

The work in our framework will be based on the separation of duty conflicts' taxonomy as discussed in [44].

The least privilege principle means a user should not be assigned more privileges than they really need in order to finish a task. The dynamic assignment of permissions in sessions has been shown to be a useful means to enforce the principle of least privilege [3]. According to the user's task or job, the privileges that are assigned to a user are divided into two parts: target privileges and non-target privileges. Target privileges are necessary to complete a task or job, and non target privileges are extra privileges that are not vital to complete a task. Since the user needs all target privileges, roles authorized for the user must cover all the target privileges. At the same time, the non-target privileges, that can sometimes be present in the user role set should be as little as possible. Otherwise, the user may use these non-target privileges to do unauthorized work [23]. Huang et al. prove in [23] that the least privilege problem for user-role assignment is NP-hard and provide approximate algorithms for solving the problem for an

RBAC and during role mining.

Discrepancies in policies may occur due to noise or error in the user-role, role-permission or user-permission assignment datasets. This can happen due to incorrect/ incomplete revocation of permissions. These discrepancies are potential risks in the RBAC and thus compliance to policies is considered to be an important risk metric.

3.3 Trust and Sensitivity

There is a lot of work in access control that discusses the trustworthiness of the user and the sensitivity of the permission to be accessed by the user. A risk-aware system must be able to make sure that a user only gets access to a permission that they are entitled to access. This means that the system should be dynamically be able to detect any changes in the data that could result in user getting access to permissions that they are not entitled to access.

Here we will discuss some of the examples from the literature that have discussed trust and sensitivity.

To determine whether the user access risk levels are on par with the permission sensitivity levels, the authors in [6], use a fuzzy logic risk inferencing system. They use this to determine permission sensitivity levels and user access risk levels and infer the risk of a particular role by looking at the aggregate sensitivity of the role and aggregate user access risk level.

In both [8] and [27], the authors discuss the principle of threat increasing if trustworthiness decreases or sensitivity of a permission increases, or if the difference between the trustworthiness and sensitivity increases.

The definition of risk as mentioned in the NIST standard [49] is used to propose a function for risk in [27]; and the function is as follows:

$Risk(s,o,a) = Threat(s,o) * Vulnerability(o) * Impact(o,a)$ where $Threat(s,o)$ represents the threat that a subject (threat source) s may present towards an object (threat target)

o, *Vulnerability*(o) represents the weakness within the existing controls for protecting (threat target) o and *Impact*(o,a) represents the adverse impact on the satisfaction of security objectives that results from successfully performing action a on o.

In [27], the authors focus on the determination of threat and defined three principles for the measurement of threat:

- **Principle 1:** Threat increases as the object sensitivity score increases.
- **Principle 2:** Threat increases as the subject trustworthiness score decreases.
- **Principle 3:** Threat increases as the difference between the object sensitivity score and subject trustworthiness score increases.

A risk graph consisting of risk bands that consist of permissions ordered according to their level of security risk is proposed in [43]. A risk distance between two permissions is the distance between the risk bands of the two permissions. If the risk distance between permissions in a particular role is too high, then this can be considered as a threat. For example, in a role $R1$ there are three permissions $P1, P2, P3$. Suppose, using a risk scale of 1 to 10 where 10 represents the highest risk, the permissions have security level 1, 2 and 8, respectively. The difference between the permission level of $P3$ is more than double of that of the other permissions and thus it could be a potential risky situation.

Dynamic properties of the user and system such as for example, time, place and currently activated roles, are used in [4] to influence the dynamic risk threshold in a session.

The authors in [8] describe how security of the physical environments and properties of information delivery channels can affect risk estimates and that taking these factors into account will result in a more holistic and realistic model for dynamic environments found in mobile settings.

In the context of role mining, it is important to check that users do not have access to permissions that they are not entitled to before the roles are defined. Ma et al. in [34] define context as a mapping from a set of attributes of the system to a set of values. Examples of context attributes include the time of access, the location of access or the system used for access. However, we will not be considering context as a part of this metric.

3.4 Permission Misuse or Abuse

While a user may have legitimate access to a permission, if they misuse that permission, then this is also an important risk factor. Thus, our final general risk metric is the misuse and abuse of permissions. If an organization is not in a preliminary stage and has existing user permission usage information then it would be particularly useful for finding out whether the permission granted to the user is being misused. In the context of role mining, if an organization is migrating from an existing access control system to RBAC (such an algorithm has been described in [59]) or is updating their current deployed system (such an algorithm has been described in [55]) then the permission misuse and abuse metrics can be used to find potential risky users. There are many examples of work in access control literature that discuss the misuse and abuse of permissions by users such as [5], [41], [3], etc. However, we will be looking at only a few examples in the context of role mining.

Determination of permission misuse and abuse is an important means to find potentially risky users before the roles have been mined and so in the context of role mining this is an important risk metric.

In [41], Molloy et al. use actual usage of entitlements to determine roles during role mining, thereby reducing operational risk. Usage provides a large amount of information such as how provisioned roles are used, e.g., the evolution of role definitions when users acting in certain roles begin to use some permissions more than others. Such analysis can also be used to identify entitlement provisioning errors such as unrevoked permissions which are never used.

Risk threshold can vary from organization to organization. Even within an organization, the risk threshold could be periodically updated based on necessity if the behavior of the users seem suspicious. For example, if a user $U1$ accesses permission $P1$ on an average of 10 times but one day suddenly accesses the permission 70 times and the permission does not fall under the category of user $U1$s target privileges, then this may be a suspicious activity. Within a session, a continuous monitoring process of users' activities and an anomaly detection mechanism can be used [4]. If the system identifies any abnormal/malicious behaviors, the session risk threshold should be decreased appropriately to stop those malicious activities.

In [5], the authors incorporate a failure modes and effects analysis (FMEA) scheme for measuring user risks. They combine components such as credentials, queries, role history logs and expected utility for a probabilistic formulation of risk. They use the risk priority number (RPN) that is defined as part of FMEA. The RPN is calculated as:
*Risk Priority Number (RPN) = Occurrence Rating (OR) * Severity Rating (SR) * Detection Rating (DR)*

The occurrence rating is a means to determine the extent of role abuse. An example would be the number of occurrences of a particular query that is submitted by the same users. Query logs of the user can be used to determine this. This frequency can be placed in a standard scale where the highest frequency rate is indicated with a 10 and the lowest rate is indicated with a 1.

Once we have found the number of occurrences of a query, we also need to find the sensitivity of the data that is being extracted using the query. The severity rating is a rating of the sensitivity of the data being accessed. Just as for the occurrence rating, for the SR a scale of 10 is used. A high SR indicates highly sensitive data and a low SR indicates less sensitive data. We will use the permission sensitivity level to determine the severity rating.

Unlike occurrence and severity rating, a detection rating is used to determine the level of role misuse. In their definition of role misuse in [5], the authors describe the situation where a user under role A submits queries under the role definition of role B. For example, let us assume that Laura works as an assistant of student affairs and primarily approves grade changes as part of her work. Now the Dean of the university is primarily responsible for authorizing university expenditure and approving late withdrawal but also has access to approval of grade changes. When we look at the usage of permissions, we suddenly find that the Dean has been approving grade changes more than Laura instead of approving late withdrawals. This is an example of abuse of legitimate permission. K-means clustering is used to find the query clusters under each role definition of each user and then the distance between these clusters is measured. If this distance is below a specific threshold then this can be considered as a threat.

Chapter 4

Framework

In this Chapter, we present a framework that incorporates some specific examples of the four risk metrics defined in the previous chapter in the context of role mining. Before role mining can be implemented, the minimum possible information that must be present and provided as input is the information about users and their corresponding permissions. Aside from this, information may be available in various degrees in different organizations. Any additional information other than the user permission assignment information can be very useful in the better determination of risk in implemented roles.

In [36], the authors propose a roadmap for role mining that describes the different types of data that may be available during role mining and what type of problems these data may be able to address. While we do believe that the availability of each of these kinds of information mentioned in [36] improves the determination of risk during role mining, we also feel that the availability of some extra information could also help in the better determination of risk. Here we discuss some of the different information types mentioned in [36] and some extra types of data that may help in alleviating risk. We also mention the particular risk metrics that we have defined that the particular type of information will help alleviate.

- **User-Attribute information:** In [36], the authors mention how user attribute information such as job positions, work departments, and job responsibilities could be used to provide a semantic meaning for roles. An access control configuration may contain noise or outliers. Once an outlier has been detected, it may not be easy to determine which particular cluster this user originally belongs to. However, if attributes of the user are available, they may help in the determination of the cluster they may belong in. Thus, in addition to comparing permissions, we can now compare attributes. Permission parameter information for permissions and permission update information as mentioned in [36] is also considered as a part of the user-attribute information mentioned here. The user attribute type of information helps in finding the similarity metric discussed in the previous chapter.

- **Permission usage data:** Permission usage data refers to the number of times a user has accessed each of the permissions that they have been assigned to within a certain time period. This type of information is also mentioned in [36]. This is a very important type of information and the presence of this information can help in the determination of risk in a number of ways. For example, for systems that require role activation, it may be desirable to group commonly-used permissions and rarely-used permissions into separate roles so as to enforce least privilege while minimizing the number of roles one has to activate for daily tasks. If a permission has never been used or has not been used for a long time by a user, then the permission assignment may be unnecessary and may be considered to be a non-target privilege. Thus the compliance of policies metric can be dealt with using this information [7]. In [41], the authors discuss how access information, if available, can help in assigning roles that are actively used by the users. This way, a user is not assigned a role that they rarely use or a user is not assigned a role that they are not supposed to have; i.e. over assignment is prevented. If a user's permission usage pattern is available then this can be compared to the user's set of permissions to check whether the permissions assigned to the user falls under the set of permissions that the user actually uses. This information is also useful to maintain the least privilege principle so that a user is assigned the minimum number of permissions that are essential for their functioning and not more than that.

Another risk metric that could use the permission usage data is the permission misuse or abuse metric. The permission usage data can be used to find the occurrence rating and the detection rating in the risk priority number calculation. This is primarily what we are doing in our framework. We are using the permission usage data to get some calculations for the risk priority number done before role mining and finding the overall risk priority number after role mining.

- **Organization Chart and Permission Sensitivity:** An organization chart provides a high level idea about the security level of the user whereas the permission sensitivity provides an idea about the security level of the permission. This type of information is not specifically mentioned in [36]. Most organizations have a well defined organization chart that has information about the hierarchy level of different job positions. If user trust level is not defined in an organization, then this organization chart can be used to get a basic idea about the security level of all the different users. Thus in our framework, the organization chart and permission sensitivity information is used to find the trustworthiness and sensitivity risk metric respectively.

- **Existing RBAC:** This type of information is also not specifically mentioned in [36]. Sometimes, an organization may have an already existing RBAC and may wish to optimize it further. In such a case, the existing RBAC or just the role set may be useful to find more accurate roles and can act as a form of policy risk metric. In [55], the authors propose an approach that identifies a set of roles that are as close as possible to both the deployed roles and the optimal set of roles using a similarity metric based on the Jaccard coefficient. In [39], the authors mention the use of maximum bipartite matching to find the difference between roles mined from a clean version and a noisy version of the same dataset. In our tool, we will be finding the difference using the Jaccard Coefficient.

- **Policy:** Again, this type of information is also not specifically mentioned in [36]. An important piece of information that an organization should provide is their security policy. A company may have many different types of security policy; however, in this framework, we will be taking into consideration two specific types of policies: separation of duty constraints and least privilege. The two types of information that are needed for these are the constraints and target privileges. The target privileges refer to permissions that must be present in each role. As discussed in [23], this information is needed to apply the least privilege principle. We also specifically take into consideration static separation of duty constraints based on the taxonomy in [45].

4.1 Structure of the Framework

Figure 4.1 shows the structure of our proposed framework. In this framework, the entire process during role mining is divided into three stages: Pre Role Mining, Role Mining and Post Role Mining. Primarily, the data discussed in the previous subsection is used as inputs to the framework to help alleviate the risk at different stages during the whole process. For the sake of understanding how the framework works, let us assume a company that has all the discussed data available. Also, let us assume that in each stage, all the information produced is stored so that the administrator can return back to it and use it as necessary at a later stage. We shall now discuss how each type of data is used in this framework for the determination of the risk metrics.

4.1.1 Pre Role Mining Stage

Starting off with the user-permission assignment, the first step consists of checking the assignment against any static user-permission constraint. This is to make sure that the current user-permission assignments are correct and that there has been no violation of constraints at this stage. Next, the similarity index is found for all combinations of users. The similarity index is found using the Jaccard coefficient.

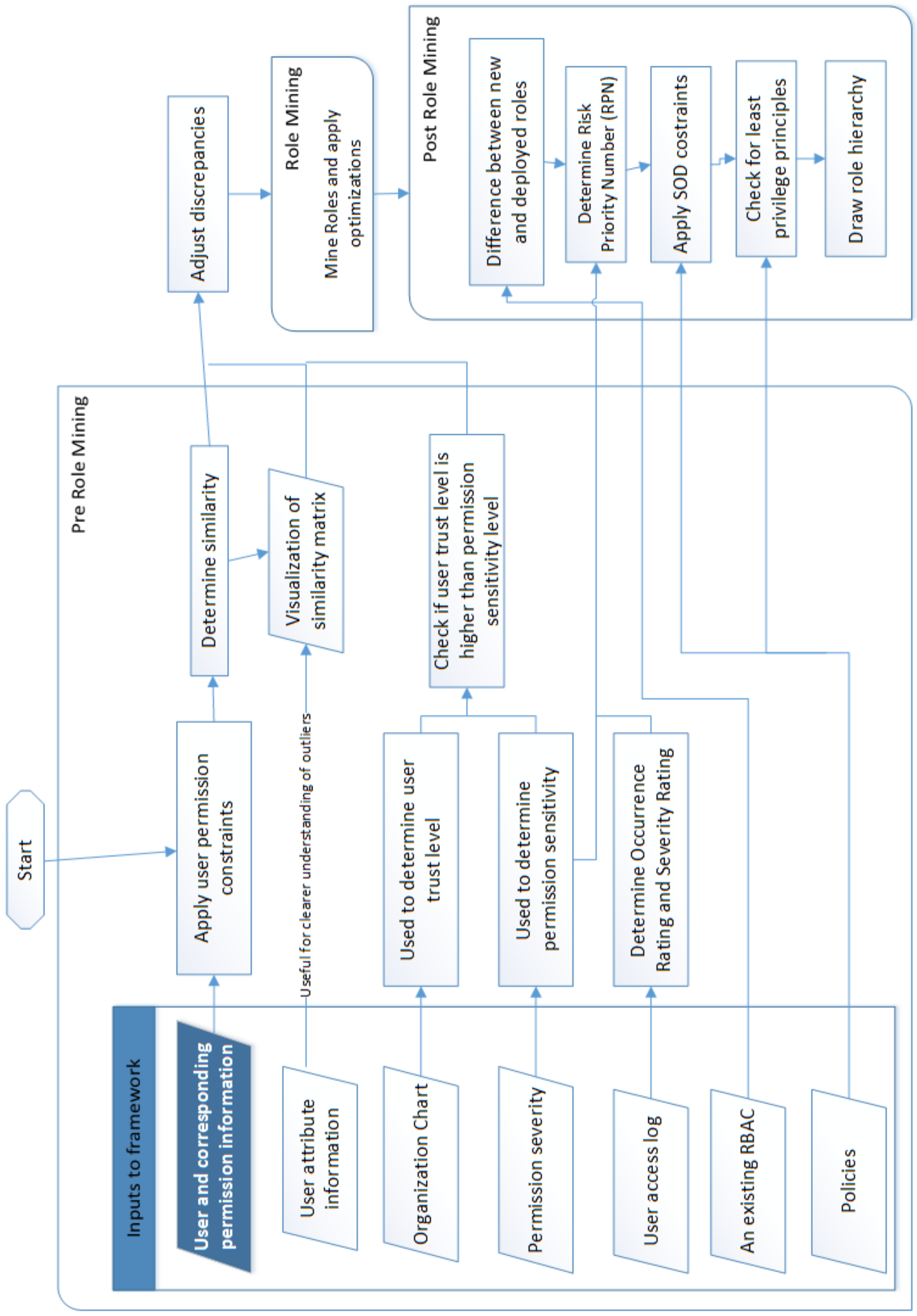

Start
Pre Role Mining
Inputs to framework
User and corresponding permission information
User attribute information
Organization Chart
Permission severity
User access log
An existing RBAC
Policies
Apply user permission constraints
Useful for clearer understanding of outliers
Used to determine user trust level
Used to determine permission sensitivity
Determine Occurrence Rating and Severity Rating
Determine similarity
Visualization of similarity matrix
Check if user trust level is higher than permission sensitivity level
Adjust discrepancies
Role Mining
Mine Roles and apply optimizations
Post Role Mining
Difference between new and deployed roles
Determine Risk Priority Number (RPN)
Apply SOD costraints
Check for least privilege principles
Draw role hierarchy

Figure 4.1: *An outline of the framework for role mining*

Now using this similarity index and the user attribute information, an aggregated visualization is made of all the users, the number of permissions they share and the attributes that they share as well. Visualization of both these types of data will help in the determination of outliers among the different users. Figure 4.2 and Figure 4.3 shows the visualization of the similarity between users and the attributes shared, respectively. In the similarity visualization figure, both the vertical and horizontal axes show all the users and the colored circles represent the amount of permission similarity between the users. The legend on the side of the charts shows the range of similarity values that the different colors represent. The attribute diagram as shown in Figure 4.3 has the different users in the horizontal axis and each circle represents whether or not the user has this attribute.

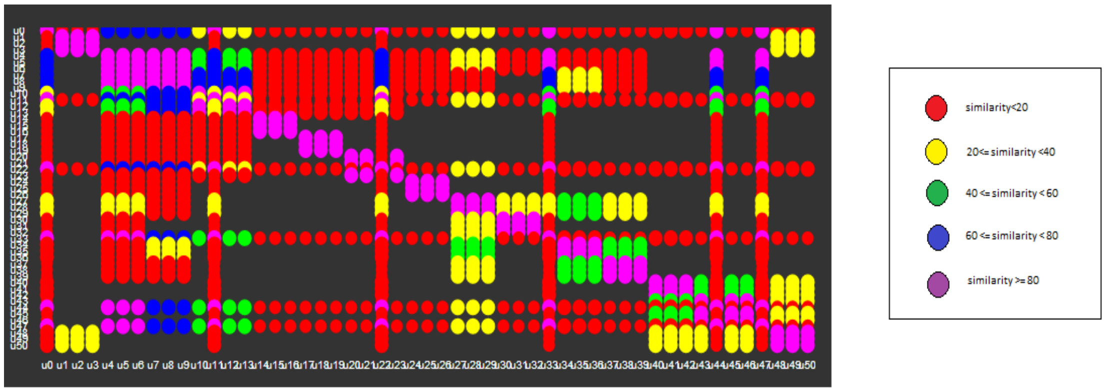


Figure 4.2: *Sample Visualization of Similarity: Here both the rows and the columns shows the different users and the different colored circles represent the amount of similarity between them. On mouseover to the circle, the user will be able to view the exact amount of overlap in permissions between the users of the particular row and column value.*

Using the organization chart information and the permission sensitivity level information, we can compare and check whether the principles discussed in Chapter 3.3 have been met. This is necessary to make sure that the difference between the trust level of the user and the sensitivity of the permission assigned to the user is acceptable or within a particular risk threshold.

The user access log information is next used to determine the product of the Occurrence Rating (OR) and the Severity Rating (SR). The occurrence rating for a particular user and permission assignment is given by the percentage the permission contributes to the frequency of

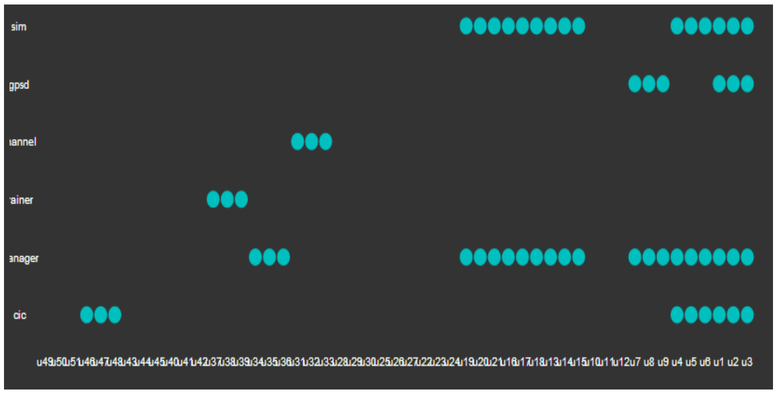


Figure 4.3: *Sample Visualization of Attributes: Here the horizontal axis represents the different users and the vertical axis represents the attributes. A circle indicates an attribute shared by a user of that particular value in the row.*

the total set of permissions accessed by the user. Unlike the method discussed in [5], we assume the severity rating of each of the permissions to be equivalent to the permission's sensitivity level. The Detection Rating is determined after the roles are mined. However, finding the product of OR and SR before role mining helps in monitoring whether sensitive permissions are accessed very frequently.

Once the OR and SR have been determined, the administrator is given the option to look at all the values generated in all the previous stages and make any adjustments necessary. Once adjustments have been made we move on to the Role Mining Stage. The level of automation of the generated values will vary from organization to organization. In a rapidly growing organization where users and permission information is frequently changed, it may be better to automatically run the tool and provide any discrepancy as notification to the administrator. In the case of an organization where user permission information does not change that frequently, it may be more suitable for the administrator to manually run the tool whenever necessary for audit purposes.

4.1.2 Role Mining Stage

There are many well known role mining algorithms in the Role Mining literature. However, these role mining algorithms do not take into consideration risk factors for the process of role mining. Instead of proposing another new role mining algorithm, we propose a first of its kind independent role mining platform. This platform ensures that upon selection of user and permission assignment information, any role mining algorithm selected by the user may be used for mining. This way we are providing the user with extra customization so that they can tailor the framework according to the needs of the company. This is because the concept of the best set of roles may vary. Some may prefer minimum role generation such as the CompleteMiner [55] while others may prefer a minimum number of edges such as in [16], while others may prefer a role mining algorithm such as Attribute Miner [36] that minimizes their weighted structural complexity metric, etc. In our framework, we are minimizing risk both before and after role mining, thus, during role mining, any type of role mining algorithm preferred by the user should be fine.

According to the work in [38], the authors categorized all the different role mining algorithms into two classes:

- **Class 1 Algorithms: Outputting Prioritized roles** Algorithms in this class output a prioritized list of candidate roles, each of which is a set of permissions. These algorithms can be divided into two phases: candidate role generation and candidate role prioritization. The candidate role generation phase identifies a set of candidate roles from the user-permission assignment data. This phase usually outputs a large number of candidate roles. The candidate role prioritization phase assigns a priority value to each candidate role; roles with a larger priority value are more important and useful.

- **Class 2 Algorithms: Outputting RBAC states** Algorithms in the second class output a complete RBAC state. Examples include ORCA [47], graph optimization [63], role/edge minimization algorithms [16], and HierarchicalMiner [36]. These algorithms take as input a configuration $\rho =< U, P, UPA >$ and output an RBAC state $\gamma =< R, UA, PA, RH, DUPA >$ that is consistent with ρ. In the state, R is a set of roles, $UA \subseteq U \times R$ is the user-role assignment relation, $PA \subseteq R \times P$ is the role-permission assignment relation, $RH \subseteq R \times R$ is a partial order over R, which is called a role hierarchy, and $DUPA \subseteq U \times P$ is the direct user-permission assignment relation. The RBAC state is consistent with $< U, P, UP >$, if every user in U has the same set of authorized permissions in the RBAC state as in UP.

These algorithms often aim at generating an RBAC state that minimizes some cost measure, such as minimizing the number of roles or the number of user-assignments and permission-assignments.

The work in [38] discusses the performance of the different popular role mining algorithms and how they are suited to different situations. For example, Hierarchical miner performed well in terms of coverage and edge minimization while other algorithms performed better in terms of role minimization and the allowance for noise. The algorithm to be used could be based on the choice of the administrator and not any predefined fixed algorithm. In the aRARM prototype that we have implemented, we have worked with four Class 2 role mining algorithms. This is because at the end of the framework the role hierarchy is generated and knowing the RBAC model makes it easier to generate the role hierarchy. In the role mining stage, we generate the user-role and role-permission assignments and a set of roles.

4.1.3 Post Role Mining Stage

Once the roles have been generated, we can use information about previous RBAC designs to check for any discrepancy between the new roles and the currently deployed roles. In this stage, we determine the Detection Rating (DR) value using the Risk Priority Number (RPN) formula as mentioned in Section 3.4. The DR value is found as discussed in [5]. Though there could be a lot of policy-based information, for the sake of this framework, we only take into account the separation of duty policies based on the taxonomy in [44] and Target Privileges. Thus, in this stage we check whether the separation of duty constraints have been met and whether the principle of least privilege has been met as closely as possible. Finally the role hierarchy is generated and saved.

Chapter 5

Proof of Concept Prototype

We have implemented a proof of concept prototype, a Risk Awareness system for Role Mining (aRARM) based on the framework described in the previous chapter. The aRARM prototype was developed to automatically detect any discrepancies or violations during the role mining process to alleviate risk. Using aRARM we are able to identify those users and permissions that represent the most likely dangerous and error prone situations from an administration point of view. Before discussing the features of the aRARM tool, we will discuss the scope of the tool.

5.1 Scope

While many important risk factors have been described in the framework, not all of them (such as dynamic separation of duty constraint) have yet been included in the aRARM tool due to time constraints. Here we define the overall scope of the factors that were implemented in the prototype.

In our prototype, static separation of duty constraints can be crosschecked for verification. Dynamic separation of duty and sessions are out of the scope. The different separation of duty constraints and the way that they are dealt with based on the taxonomy discussed in [44] are shown in Figure 5.1.

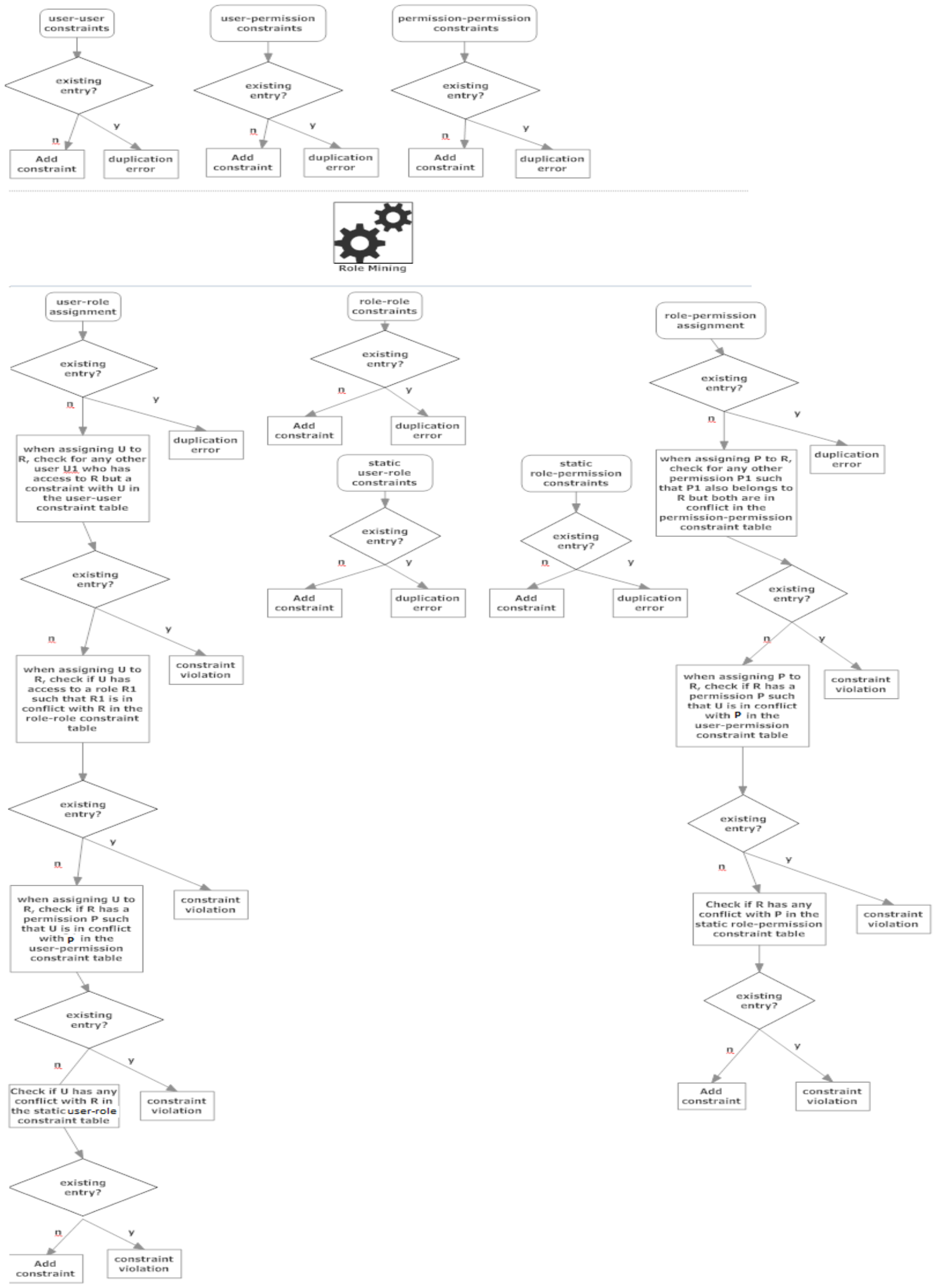

user-user constraints
user-permission constraints
permission-permission constraints
existing entry?
n
y
Add constraint
duplication error
Role Mining
user-role assignment
role-role constraints
role-permission assignment
static user-role constraints
static role-permission constraints
when assigning U to R, check for any other user U1 who has access to R but a constraint with U in the user-user constraint table
when assigning U to R, check if U has access to a role R1 such that R1 is in conflict with R in the role-role constraint table
when assigning U to R, check if R has a permission P such that U is in conflict with p in the user-permission constraint table
Check if U has any conflict with R in the static user-role constraint table
when assigning P to R, check for any other permission P1 such that P1 also belongs to R but both are in conflict in the permission-permission constraint table
when assigning P to R, check if R has a permission P such that U is in conflict with P in the user-permission constraint table
Check if R has any conflict with P in the static role-permission constraint table
constraint violation

Figure 5.1: *Flowchart for checking separation of duty constraints*

In Figure 5.1, before role mining, the user-user constraints, permission-permission constraints and the user-permission constraints, if available, are applied to the user and permission information. To input the constraint information into the database, the tables are checked for any existing entry and rows are added only when they are not already there.

After roles have been mined, system updates could be made for maintenance purposes and users can be assigned to roles based on the following checklist, as shown in the lower part of the framework in Figure 5.1.

- Check whether the user-role assignment is already existing to prevent duplication
- Check whether the user U being assigned to role R has a conflict with any other user $U1$, in the user-user constraint table, such that $U1$ has access to R. This is a constraint violation and a risky situation.
- Check whether the user U being assigned to role R has a conflict with permission P in the user-permission constraint such that P is already in role R.
- Check whether user U being assigned to role R has access to a role $R1$ such that role R and role $R1$ are in conflict in the role-role constraint. While this is not directly possible right after role mining because we do not yet know the names of the roles that have been generated, it is possible if the tool is used for RBAC management after roles have been defined and role-role constraints are defined later on. Further role mining can be skipped and the user can directly move on to the separation of duty constraint verification stage to check for any discrepancy if any new user is added to the existing RBAC.
- Also, if static user-role constraints are defined after role mining, then these can also be cross checked.

After roles have been mined, system updates could be made for maintenance purposes and permissions can be assigned to roles based on the following checklist, as shown in the lower part of the framework in Figure 5.1.

- Check whether the role permission assignment is already existing to prevent duplication.
- Check whether the permission P being assigned to role R has a conflict with any other permission $P1$, in the permission-permission constraint table, such that $P1$ has access to R. This is a constraint violation and a risky situation.
- Check whether the permission P being assigned to role R has a conflict with any user U in the user-permission constraint table, such that U has access to R. This is a constraint violation and a risky situation.
- If static role-permission constraints are defined after role mining, then these can also be crosschecked.

In [23], the authors mention that the principle of least privilege requires that the users are assigned with non-necessary privileges as little as possible in order to complete a task or job. A target privilege is necessary to complete a task or job, and a non-target privilege is not. Since the user needs all target privileges, roles authorized for the user must cover all the target privileges. At the same time, the non-target privileges should be assigned to the user as little as possible. Otherwise, the user may use these non-target privileges to do unauthorized work. While we are dealing with target privileges, we primarily focus on whether target privileges have been met and not the number of extra (non-target) privileges that are added because of noise.

We have tested our tool using four role mining algorithms: CompleteMiner by Vaidya [57], the Graph Optimization algorithm by Zhang et al. [63], the algorithm by Ene et al. [16] and the algorithm mentioned in [45]. Each of these algorithms are described in more detail in a later Chapter. While these are well-known role mining algorithms, they have not been designed to deal with noise. As noise reduction was not a prime objective in this work, noise reducing algorithms were not used.

Finding real world access control data along with usage, policy, trust and sensitivity is difficult. We instead simulated two small data examples based on past project experiences and the university database mentioned in [60] and tested the tool on that. The rows of data were randomly generated based on Vaidya's simulation in [57]. This is discussed in more detail in the next Chapter.

As the data was simulated, there were no pre-defined values for the risk threshold. We argue that the risk threshold of what is acceptable and what is not varies from organization to organization and our tool is open to that. We sort and show the highest discrepancies at the top of the report. However, based on multiple readings, we did make an assumption about the acceptable value of detection rating for the risk priority number value which we shall discuss next in Prototype Design.

5.2 Prototype Design

The aRARM prototype was implemented in Java with a MySQL database. The interface of the tool is designed in a software wizard style with each window providing the option to input each type of additional information as shown in the framework in Figure 4.1. Here we summarize the several features of the prototype.

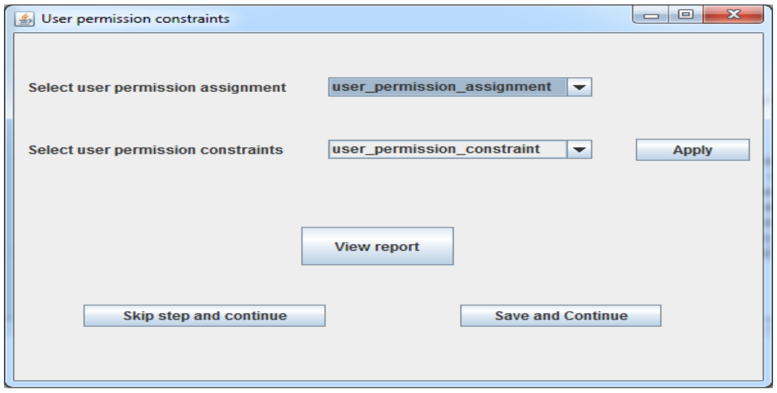
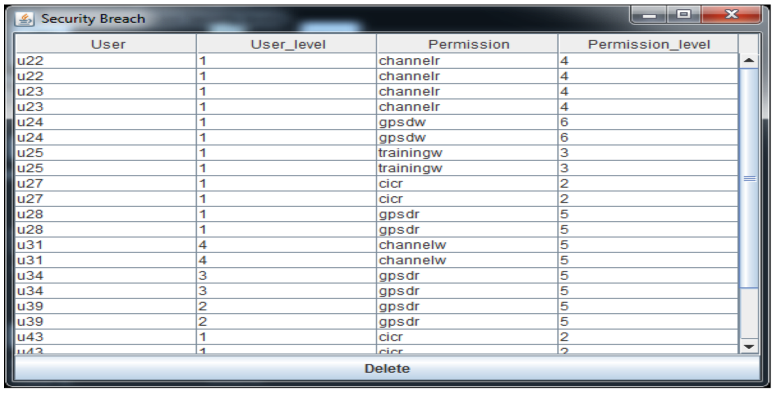

User permission constraints
Select user permission assignment
user_permission_assignment
Select user permission constraints
user_permission_constraint
Apply
View report
Skip step and continue
Save and Continue

Figure 5.2: *Sample Window: Here the user can select from a list of all the tables in the dataset that are shown in the drop down menu to get the table that contains the information. For example, here the user can look at all the tables to find the table that contains the user permission assignment information and user constraint information respectively.*

- **Flexibility of input:** Other than the user-permission assignment information, all other information is optional and the user can either input the information in the step or skip the step. For example, in the first step, the user is asked to select any table that may contain any information regarding user-permission static separation of duty constraints. If the user does not have this information, they may skip this and proceed to the next step. If the user does have this information, then this information can be formatted according to the tool necessity and then the user can select the table and the tool will check for any discrepancy.

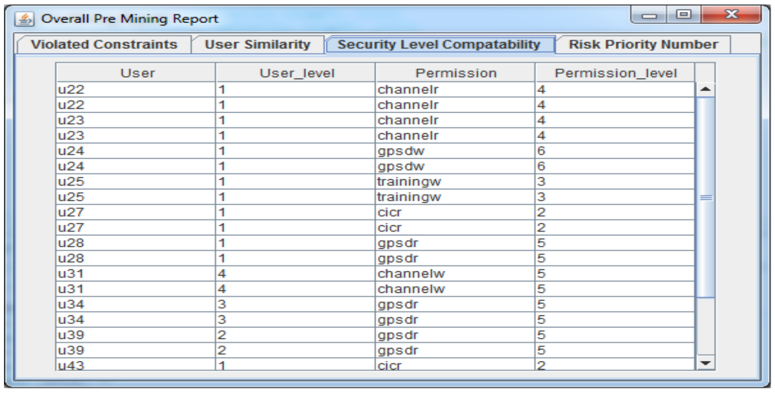


User	User_level	Permission	Permission_level
u22	1	channelr	4
u22	1	channelr	4
u23	1	channelr	4
u23	1	channelr	4
u24	1	gpsdw	6
u24	1	gpsdw	6
u25	1	trainingw	3
u25	1	trainingw	3
u27	1	cicr	2
u27	1	cicr	2
u28	1	gpsdr	5
u28	1	gpsdr	5
u31	4	channelw	5
u31	4	channelw	5
u34	3	gpsdr	5
u34	3	gpsdr	5
u39	2	gpsdr	5
u39	2	gpsdr	5
u43	1	cicr	2
u43	1	cicr	2

Figure 5.3: *Sample Report*

- **Detailed report in each step and an Overall Report:** In each step, any discrepancy in the data is found and recorded. The user may view the discrepancy report in each step or alternatively save the information and proceed to the next step. Once all the pre-role mining steps have been completed, the next window allows the user the option to view the different reports in all the previous steps in an overall dashboard (please see Figure 5.4).

- **Visualization of the similarity index and attributes shared:** In the second step in the pre-role mining phase, the user has the option to visualize the similarity between the users as shown in Figure 5.5 as well as the attributes that they share as shown in Figure 5.6. The different colors in the visualization refers to different ranges of similarity. A similarity of less than 20 is represented by a red circle. A similarity of greater than or equal to 20 and less than 40 is represented as a yellow circle. A similarity of greater than or equal to 40 and less than 60 is represented with a green circle. A similarity of greater



User	User_level	Permission	Permission_level
u22	1	channelr	4
u22	1	channelr	4
u23	1	channelr	4
u23	1	channelr	4
u24	1	gpsdw	6
u24	1	gpsdw	6
u25	1	trainingw	3
u25	1	trainingw	3
u27	1	cicr	2
u27	1	cicr	2
u28	1	gpsdr	5
u28	1	gpsdr	5
u31	4	channelw	5
u31	4	channelw	5
u34	3	gpsdr	5
u34	3	gpsdr	5
u39	2	gpsdr	5
u39	2	gpsdr	5
u43	1	cicr	2

Figure 5.4: *Overall Pre Mining Report*

than or equal to 60 and less than 80 is represented with a blue circle. Finally a similarity of greater than or equal to 80 is represented by a purple circle. The user will also be able to view a report that tabulates any users that do not have a 100% permission match with any of the other users i.e., the outlier users.

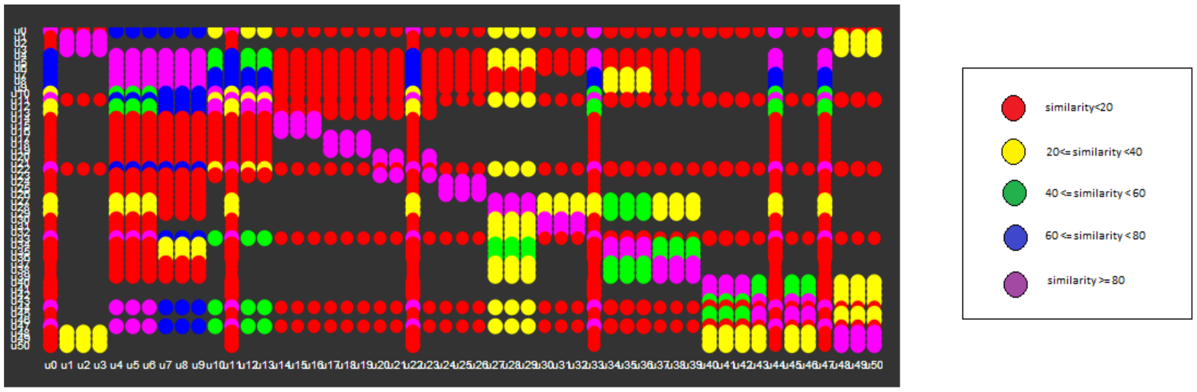


Figure 5.5: *Sample Visualization of Similarity: Here both the rows and the columns show the users and the different colored circles represent the amount of similarity between them. On mouseover to the circle, the user will be able to view the exact amount of overlap in permissions between the users of the particular row and column value. The legend on the side shows the similarity range of each of the colors.*

- **Actions regarding discrepancy:** We propose two variations regarding actions. In our current prototype, in each step, if any discrepancy is found and when the user views it in a report, there is an option to delete one or more of these discrepancies. When a row is deleted, then the user-permission assignment of that particular row is deleted from the database. However, this action may seem inappropriate and total deletion of the

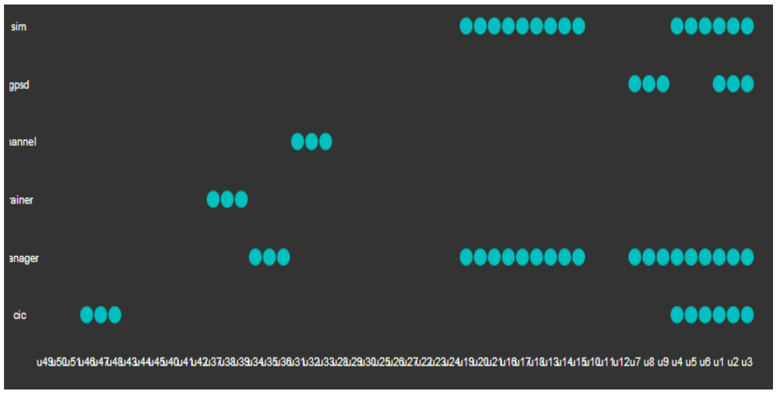


Figure 5.6: *Sample Visualization of Attributes: Here the rows represent the users and the columns represent the attributes. A circle indicates an attribute assigned to a user of that particular value in the row.*

assignment would be too harsh. We thus also propose the option to 'flag' the particular user-permission assignment so that the flagged assignments can be dealt with later on.

- **Role Mining Platform:** A role mining platform is also embedded within the prototype that is unbiased to the type of algorithm to be used. As the prototype is open to the role mining algorithm used, any algorithm based on the preference of the administrator can be applied.

- **Risk priority number calculation:** We have already seen that the risk priority number is calculated as: $OR * SR * DR$. In this prototype, the sensitivity level of each permission is the SR. The OR is the percentage of usage of each permission. Thus, if a user u has used permission p_1 as 50% of their total usage of all permissions at a particular time, then their occurrence rating for permission p_1 is 50%. The initial DR before role mining is done based on the assumption that a set of users that have a 100% match of permissions (i.e. similarity of 100% with each other) could be considered as a set of users having the same set of roles. Thus, the detection rating is found for each permission of each user from this cluster.

  In the detection rating in [5], $k - means$ clustering is used between the different users in each role. We use $k - means$ clustering after role generation. More on this is discussed in detail in Section 7.5.

- **Policy Comparison:** For our prototype, we have taken into consideration static separation of duty constraints and the principle of least privilege. The tool checks for violation of separation of duty constraints both before and after role mining and also checks for conformation to the principle of least privilege. The latter is done by checking whether roles generated contain the target privileges.

The entire process of the tool is given in the flowchart in Figure 5.7.

We describe the procedure in the tool by describing two particular processes in the tool: the process for constraint validation and the process for the RPN calculation during the timeline of the tool.

5.2.1 Sequence for constraint validation in the tool

In the sequence of events for the constraints as shown in Figure 5.8, we save the data for the user, permission and user-permission information into the database. Now if the constraint values are available, then the value for the constraints are first saved in the database as well. We validate the constraint values in the constraints validator both before and after role mining according to the flowchart shown in Figure 5.1. The values needed for validation are retrieved from the database and the results are returned. Now if there is any discrepancy in the reported results then the erroneous rows can be selected and these selected rows of relationships are deleted from the database. If there is no discrepancy then no further action is necessary.

5.2.2 Sequence for RPN in the tool

The sequence of events for RPN as shown in Figure 5.9 is more complicated than the one for constraint validation mentioned in the previous section. We are assuming that the minimum possible information that must be present in an organization is the user permission assignment

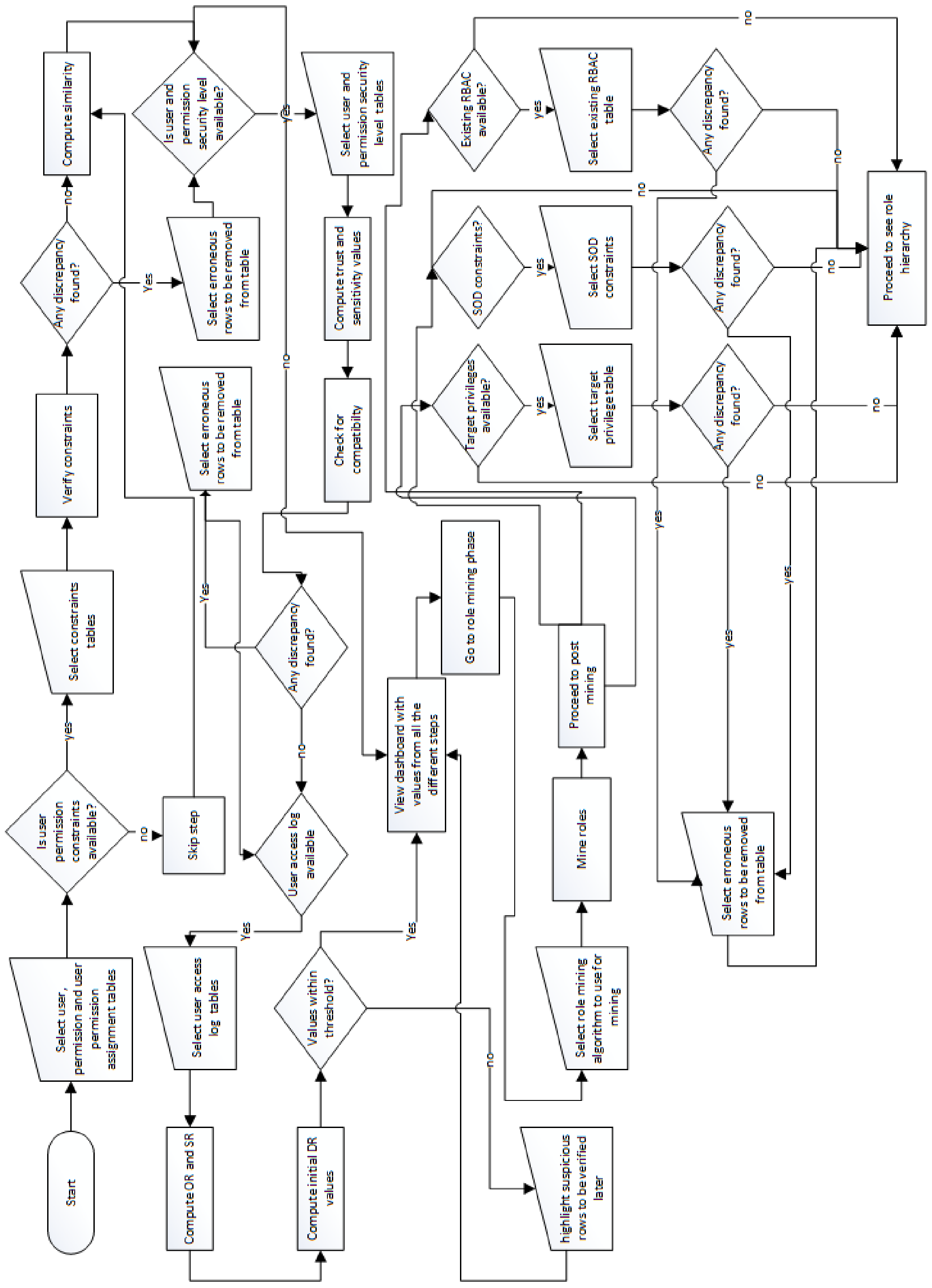

Start
Select user, permission and user permission assignment tables
Is user permission constraints available?
yes
no
Skip step
Select constraints tables
Verify constraints
Any discrepancy found?
no
Yes
Compute similarity
Select erroneous rows to be removed from table
Is user and permission security level available?
yes
no
Select user and permission security level tables
Compute trust and sensitivity values
Check for compatibility
Select erroneous rows to be removed from table
Any discrepancy found?
Yes
no
User access log available
Yes
Select user access log tables
Compute OR and SR
Compute initial DR values
Values within threshold?
Yes
no
View dashboard with values from all the different steps
highlight suspicious rows to be verified later
Go to role mining phase
Select role mining algorithm to use for mining
Mine roles
Proceed to post mining
Existing RBAC available?
yes
no
Select existing RBAC table
Any discrepancy found?
no
SOD constraints?
yes
Select SOD constraints
Any discrepancy found?
no
Target privileges available?
yes
no
Select target privilege table
Any discrepancy found?
no
yes
yes
yes
Select erroneous rows to be removed from table
Proceed to see role hierarchy

Figure 5.7: *Flowchart for aRARM*

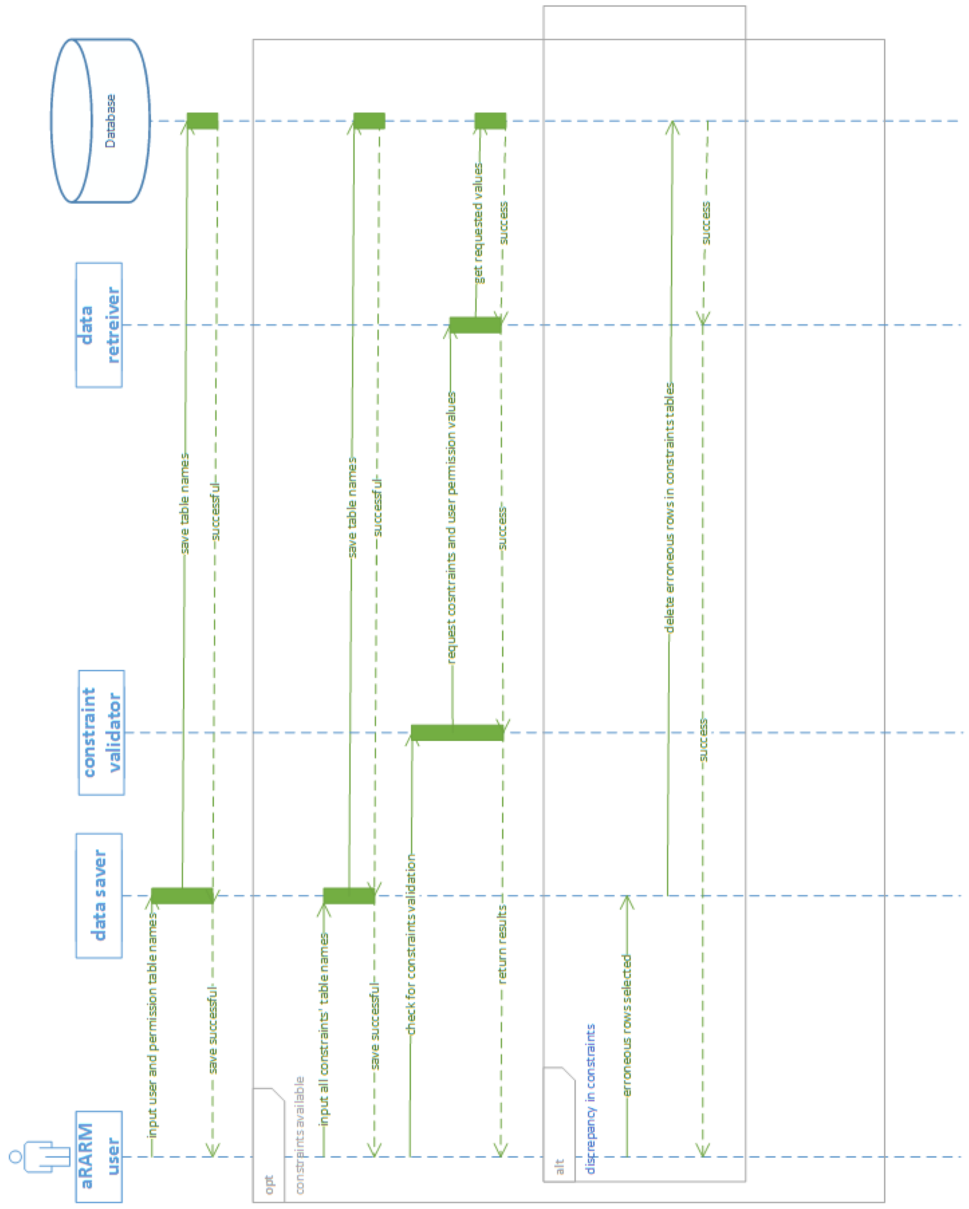


Figure 5.8: *Sequence for Constraints*

information. Let us also assume for the sake of this section that the user trust and user sensitivity level are also present. Consideration for RPN calculation is only possible when the trust and sensitivity information is also present. Now if the access logs are available, then we first save the access log information in the database. We calculate the Occurrence Rating and Severity Rating using the user-permission assignment information, the permission severity and the access log information. To determine the initial Determination Rating, we first find the clusters that have users that share 100% similarity with respect to permissions. Using the clusters we find the initial DR values and the OR, SR and initial DR values are returned. If the company has a particular threshold with regards to these values, then we need to verify that the values have not gone beyond this threshold. If some values have gone beyond this threshold then these values are highlighted for further assessment. After the role mining phase, we calculate the actual DR and the overall RPN value and the values are returned. If the overall RPN value is beyond a threshold value then this value is highlighted for further assessment.

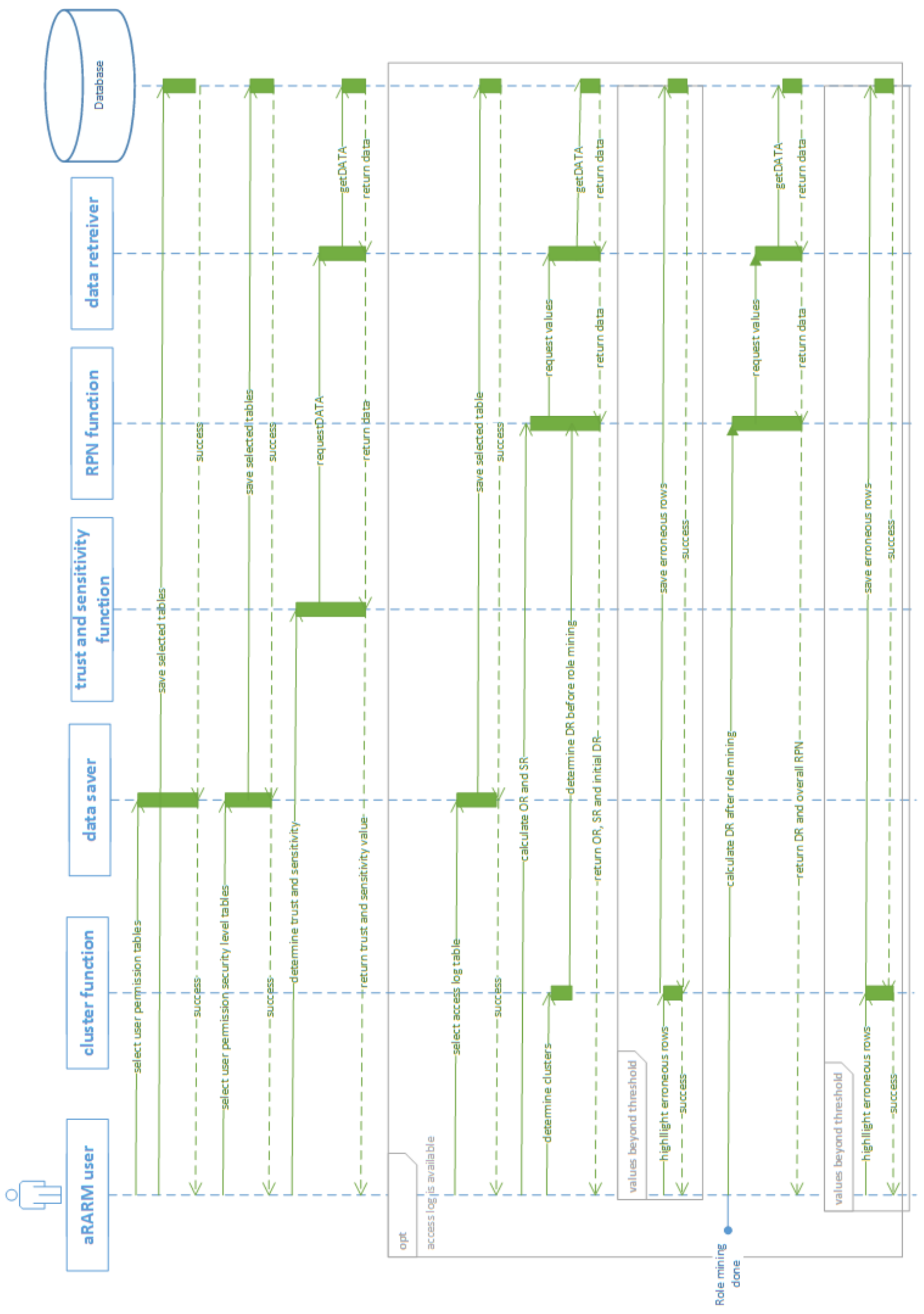


Figure 5.9: *Sequence for RPN*

Chapter 6

Case Studies

To determine the validity of our prototype, we carried out two case studies on two different examples. In this Chapter we will discuss the two case studies and the data collection method details before we move on to the results and interpretation of the results of the case studies. The first case study was based on a real RBAC model implemented in a small project involving a telecommunication company in Bangladesh. The second case study is based on a University RBAC model described in [60].

6.1 Case Study 1: Project RBAC Model

Our first case study is based on a real RBAC model implemented in a small project involving a telecommunication company in Bangladesh. The original role graph of the project is given in Figure 6.1. Face validity is the extent to which a test is subjectively viewed as covering the concept it purports to measure [51]. The data generated for the role hierarchy has been face validated [51] by consulting previous members of the project.

This RBAC model originally consisted of 17 roles, 51 users, 16 permissions and 231 user-permission assignments. This project was introduced to set up new businesspeople, known as entrepreneurs to set up telecenters known as CIC (Community Information Centres) in their local area. As a part of the set up, the entrepreneurs need to be identified, selected, trained, a

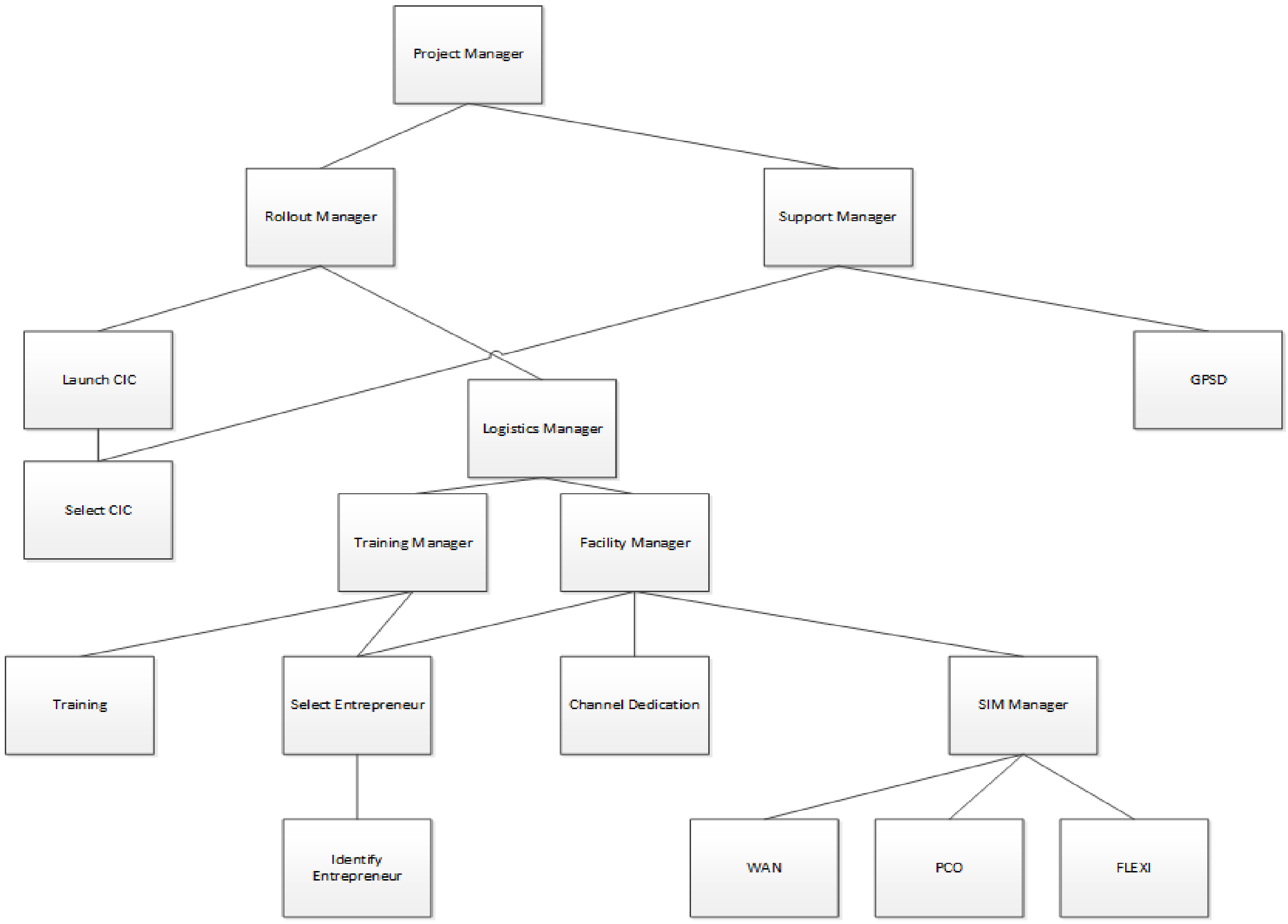


Figure 6.1: *Role Hierarchy in Community Information Centre project team*

channel dedicated for the three SIM (Subscriber identity module) cards (PCO, Flexi, WAN) to be given to them and finally their shops need to be launched and rolled out. A nearby service desk, known as GPSD, is for any kind of customer support needed. Different team members of the project are responsible for different tasks regarding the entrepreneurs. For example, the training manager is able to select particular entrepreneurs and provide their training. A WAN team member maintains information regarding the WAN SIM to be handed out to the entrepreneurs.

Much information has been added to this data for the determination of risk factors. These are given below:

- 57 attribute-based pieces of information regarding the users.
- User trust level and permission sensitivity level for all the 51 users and 16 permissions
- query log-based information regarding all the 231 user-permission assignments
- 66 target privileges set up for all the 51 users
- 50 separation of duty constraints

6.2 Case Study 2: University RBAC Model

For our second case study, we used a University RBAC model as described in [60]. The University RBAC has been modified and additions have been made based on the need to study risk factors. However, the basic structure is the same as the one mentioned in the paper. The RBAC model of the university database is shown in Figure 6.2.

The university RBAC model consists of 104 users, 49 permissions, 31 roles and 916 user permission assignments. The model is specific to a single unknown department and does not expand to all the different departments in the university. The model consists of a separation of the students, staff, faculty and other departmental individuals and the permissions that they are granted. While on first view of the model it may seem that the student and the university employee models are distinct but actually there are certain student roles such as TA and grader that share certain permissions such as view and assign student grades with other roles such as faculty.

Much information has been added to this data for the determination of risk factors. This is given below:

- User trust level and permission sensitivity level for all the 104 users and 49 permissions
- query log based information regarding all the 916 user-permission assignments
- 197 target privileges set up for all the 104 users
- 211 separation of duty constraints

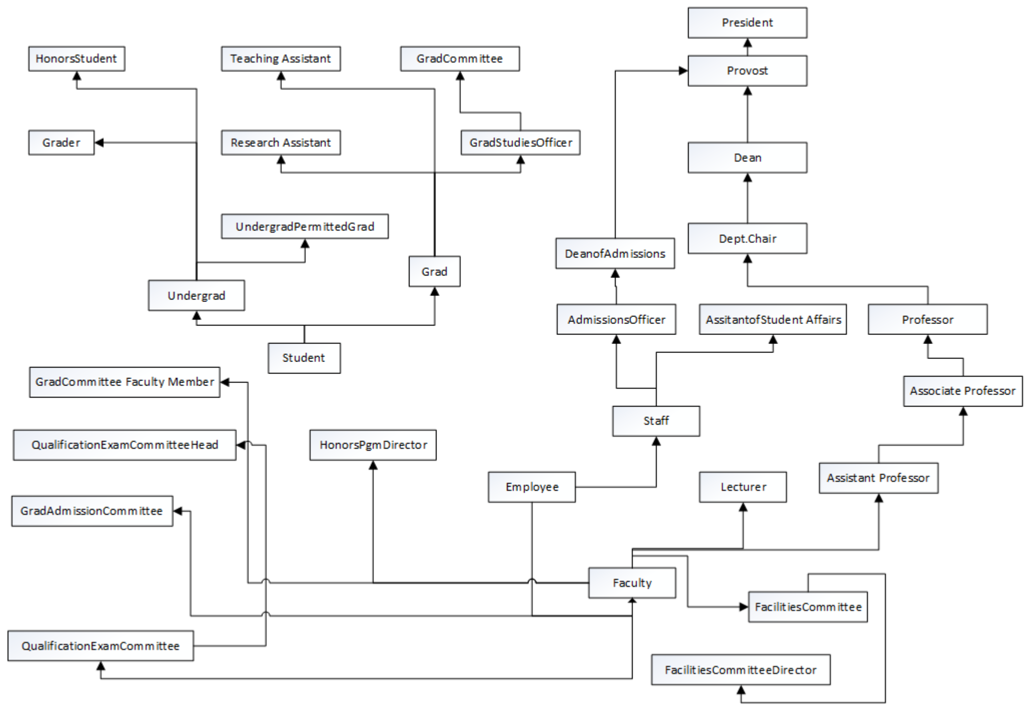


Figure 6.2: *University RBAC model*

Here we will describe some examples of the constraints that have been added.

- We had to make sure that normal students with no special privileges do not under any circumstance get access to view other students' information or assign student grades
- Anyone other that the dean, provost and the president should not be able to modify the departmental budget
- No employees other than the president should be able to authorize university account expenditure
- No employees other than the president, provost and dean should be able to authorize the college account expenditure
- Only the facility's committee director should be able to authorize equipment purchases.

Here we will describe some examples of the different target privileges that have been specified.

- The president must be able to assign a provost
- A grad studies officer must be able to reserve rooms
- An honors student must be able to register and withdraw from honors classes
- An admissions officer must be able to register students
- A faculty member must be able to revise and submit grades

6.3 Data Analysis

We conducted two case studies on the telecommunication company and university data to understand the behavior of the different risk factors with different levels and types of noise. Determination of risk for role mining requires data from a mature company that may contain different types of errors. An example of this is if a user was given access to a particular permission in a user group based on a project requirement and then the user finished work on that project and left but their permissions were not revoked. Another example could be if a user required access to more than one permission for a role and not all the permissions were revoked. Thus real world data for the determination of risk would not be completely clean and would have some errors in it. However, in the absence of such data, it needs to be simulated. We simulate real world data by artificially introducing noise into the data.

First, the information regarding the original role graph was saved. Next, noise was added to the original user-permission assignment (UPA) data according to the format mentioned in [56]. The entire user-permission mapping is converted to an $m \times n$ boolean matrix where a 1 in cell (i, j) indicates the assignment of permission j to user i. Because the data set for the project was very small compared to the one in [56], the range of noise added over the entire UPA assignment was between 1% and 5%. However, we increased the noise addition to 10% for the university database so noise was added between 1% to 10% for the university database.

The noise addition was carried out in the following three ways:

- **Affect the zeros only:** A random number of zeros within the range of 1% to 5% or 1% to 10% of the Boolean matrix is flipped and converted into a 1. This is similar to the additive noise mentioned in [56].

- **Affect the ones only:** A random number of ones within the range of 1% to 5% or 1% to 10% of the Boolean matrix is flipped and converted into a 0. This is similar to the subtractive noise mentioned in [56].

- **Affect both the ones and the zeros:** A random number of ones and zeros within the range of 1% to 5% or 1% to 10% of the Boolean matrix are flipped. Thus, if a 1 is chosen it is converted into a 0 and if a 0 is chosen it is converted into a 1. This is similar to the general noise mentioned in [56].

Once the noise has been added, the prototype is run to check whether any discrepancy in each step is detected. Each run is repeated 5 times and a floor value of the average is chosen.

Our data variables are of type continuous interval [51]. Our research is of type experimental research where we are manipulating the independent variable and then examining the change that it has on the dependent variable. In our case, the independent variable is the noise that we are adding. The dependent variables that we are analyzing are:

- **(M1:)** Number of constraints violated
- **(M2:)** Number of outliers
- **(M3:)** Number of security breaches
- **(M4:)** Number of target privileges not met

For analyzing such data, nonparametric statistical methods are used that make very few assumptions about the distribution of the population [51]. As all our risk factors are measured in a continuous interval scale, we used a nonparametric statistical method such as Spearman rank order correlation test. The statistical software StatsDirect [30] is used for the ease of statistical analysis. The Spearman rank correlation test was used to calculate the risk factors. The Spearman rank correlation test calculates the Spearman rank correlation coefficient (ρ) between two sets of paired data. We do various paired combinations of the risk factors and find the correlation between the pairs. If the ρ value is in between 0.4 and 0.7 then the correlation is considered to be moderate whereas ρ value more than 0.7 is considered as a high correlation. The 95% confidence intervals (CI) and the p-values for the correlations were calculated. If the 95% confidence intervals do not include zero (i.e., either the upper and lower bound of CI are positive or both of them are negative) then the values are considered statistically reliable [51]. A p-value < 0.05 were considered as statistically significant evidence for rejecting the null hypotheses in accordance with [51].

Because we are unsure about the effect that the different types of noise will have on the different risk factors, we define our research hypothesis for the two case studies as follows:

Research Hypothesis: *When the amount of noise applied increases, the value of every dependent variable, i.e., number of constraints violated, number of outliers, number of security breaches and number of target privileges not met increases as well*

In the next Chapter we will look into the results of the two case studies and determine the validity of our research hypothesis.

Chapter 7

Results and Interpretations

In this Chapter we will discuss the results from the two case studies and their interpretations. First, we will discuss the results and interpretations of the two case studies separately and then do a combined analysis of the results. We also discuss the results of the two case studies with respect to the four algorithms that we applied to the case studies and discuss the interpretations of that. Finally we discuss the Risk Priority Number calculation that we do before and after role mining.

7.1 Case Study 1 Results

In this section, we will discuss the results of the first case study. The results shown in Tables 7.1, 7.2 and 7.3 are based on the different types of noise added as already discussed. In each table, the violated constraints value represents a percentage of the total number of constraints that have been violated. The security level breach represents a percentage of the total number of user-permission assignments that are security breaches and where the user has gained access to permissions that are beyond their level. The target privileges not met represents a percentage value of the total number of target privileges that have not been met. The outlier column represents as a percentage the fraction of the total users that do not have any other users that share the same set of permissions, i.e. if a user does not have atleast one 100% similarity value

Table 7.1: Table for additive noise for project case study

	Violated Constraints	Security Level Breach	Target Privileges Not Met	Outlier
1%	0%	1.7%	0%	11.8%
2%	0%	3.5%	0%	27.5%
3%	4.8%	4.8%	0%	39.2%
4%	9.5%	6.1%	0%	51%
5%	9.5%	9.5%	0%	64.7%

Table 7.2: Table for subtractive noise for project case study

	Violated Constraints	Security Level Breach	Target Privileges Not Met	Outlier
1%	0%	0%	3%	11.8%
2%	0%	0%	7.6%	25.5%
3%	0%	0%	10.6%	29.4%
4%	0%	0%	13.6%	39.2%
5%	0%	0%	21%	50.98%

with one other user, then it is considered an outlier. Each row represents the percentage of noise of the particular type added. Readings have been taken for the case study dataset in the absence of noise but it is not included here because it did not show any significant reading in any of the risk metrics.

Table 7.3: Table for general noise for project case study

	Violated Constraints	Security Level Breach	Target Privileges Not Met	Outlier
1%	0%	0.9%	1.5%	11.8%
2%	0%	2.2%	1.5%	29.4%
3%	4.8%	3.5%	3%	39.2%
4%	4.8%	5.7%	1.5%	51%
5%	4.8%	4.8%	4.6%	60.8%

If we look at the constraints violated column in the additive, subtractive and general noise tables we will see that the subtractive noise has no impact on the violated constraints whereas both in the case of additive and general noise the violated constraints value increases with increase in noise upto a certain point and then becomes constant. In summation these results do not help us to make a conclusive deduction about the effect of noise on violated constraints. This makes sense because the constraints implemented for this prototype are static, thus the probability of their violation is equally distributed in all the different types of variations applied. This value could be different if dynamic constraints were applied. The outlier column is not affected by the type of noise applied. The value indiscriminately increases as the noise increases.

There appears to be an interesting relationship between the Security Level Breach and Target Privileges not met. When the ones are flipped to zeros, there is no noticeable security level breach but there is a consistently high number of target privileges that have not been met. It is the other way around when the zeros are flipped to ones: there is no noticeable number of users that have not had their target privileges met but there is an increasingly higher number of security level breaches. This phenomenon is explainable by the fact that when many of the available privileges are converted to zero, it is still not a security breach, but rather an under assignment. However, under assignment carries with it the risk of depriving users from their rightful privilege, hence the high number of target privileges not met. On the contrary, during over assignment of privileges in additive noise, even though users are getting their target privileges, it is risky because users are getting access to permissions that they do not deserve. The randomness of the noise is noticeable in the general noise table in Table 7.3 where the security breach and target privileges not met values sometimes decreases and then sometimes increases again.

The overall results of the readings were explained in [39]. These are Type I (false positive) and Type II (false negative) errors and as the authors mention that balancing the allowed and denied actions is a very delicate process with many tradeoffs. Aggressively reducing Type I errors will likely result in an increase in Type II errors, and vice versa.

As our data size for the project is too small it is difficult to come up with specific inferences for the rejection of the null hypothesis. The results shown in Tables 7.4, 7.5 and 7.6 show that the results from the small case study data is not statistically significant. We did not apply greater than 5% noise because it would be unrealistic for such a small dataset. However, based on the results that we found we decided to experiment on the correlation between the various risk factors at a fixed high noise value of 5%. We applied Spearmans Rank Correlation

Coefficient [32] and found that a reliable positive correlation only exists between the outliers and the security breach. When we say reliable we mean statistically significant. If we look at the correlation coefficient values in Tables 7.4, 7.5 and 7.6 , we see that many of the ranges between the upper and lower limit consists of the value 0, which signifies statistical insignificance. The empty cells represent undefined values that occurred due to the presence of the 0 values in the readings.

Table 7.4: Spearmans Rank Correlation and 95% and 99% Confidence Interval for additive noise for the first case study

		95%CI		99%CI	
		lower	Upper	lower	Upper
Outlier	Violated Constraint	-0.136	0.403	-0.222	0.475
Outlier	Security Breach	0.305	0.706	0.222	0.748
Outlier	Targets not met				
Violated Constraint	Security Breach	-0.342	0.205	-0.418	0.289
Violated Constraint	Targets not met				
Security Breach	Targets not met				

Table 7.5: Spearmans Rank Correlation and 95% and 99% Confidence Interval for subtractive noise for project case study

		95%CI		99%CI	
		lower	Upper	lower	Upper
Outlier	Violated Constraint				
Outlier	Security Breach				
Outlier	Targets not met	-0.107	0.428	-0.194	0.498
Violated Constraint	Security Breach				
Violated Constraint	Targets not met				
Security Breach	Targets not met				

However, if we look carefully we see that for the additive and general noise the correlation between security breach and outlier is statistically significant. We will be using this significance to find the correlation for the security breach and outlier for general and additive noise between the ranges of 1% and 5%. The details of the tables are provided in Appendix B.

Our results are shown in table 7.7. The table shows the values for general and additive noise only. We omitted the results for subtractive noise because it gave undefined values. The 95%

Table 7.6: Spearmans Rank Correlation and 95% and 99% Confidence Interval for general noise for project case study

		95%CI		99%CI	
		lower	Upper	lower	Upper
Outlier	Violated Constraint	-0.152	0.389	-0.238	0.462
Outlier	Security Breach	0.045	0.544	-0.044	0.603
Outlier	Targets not met	-0.012	0.503	-0.1	0.566
Violated Constraint	Security Breach	-0.335	0.213	-0.411	0.297
Violated Constraint	Targets not met	-0.313	0.237	-0.391	0.318
Security Breach	Targets not met	-0.03	0.489	-0.118	0.554

Table 7.7: Table for correlation between security breach and outlier for project case study

	General Noise				Additive Noise			
	95% CI		99% CI		95% CI		99% CI	
	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper
1%	0.088	0.574	-0.001	0.63	0.493	0.802	0.423	0.831
2%	0.026	0.53	-0.063	0.591	0.306	0.707	0.223	0.748
3%	0.05	0.547	-0.039	0.607	0.148	0.613	0.06	0.665
4%	0.105	0.585	0.016	0.64	0.126	0.599	0.038	0.653
5%	0.114	0.591	0.026	0.646	0.205	0.648	0.118	0.697

and 99% confidence interval values are also shown. The results do not indicate a statistically significant relationship between the two variables. Only a few of the results do not include a 0 value and thus no explicit inference can be made based on this result.

7.2 Case Study 2 Results

In this section, we will discuss the results of the University case study. The University case study was designed as a bigger study to get answers to questions that remained unresolved in the first case study. Let us first look at the results.

The results shown in Tables 7.8, 7.9 and 7.10 are based on the different types of noise added as already discussed. In each table, the violated constraints value represents a percentage of the total number of constraints that have been violated. The security level breach represents a percentage of the total number of user-permission assignments that are security breaches and permissions that the user has gained access to that are beyond their level. The target privileges

Table 7.8: Table for additive noise for University case study

	Violated Constraints	Security Breach	Targets not met	Outlier
1%	1.422%	1.638	0%	46.154%
2%	1.422%	3.821%	0%	73.077%
3%	2.37%	6.223%	0%	91.346%
4%	3.318%	6.550%	0%	94.231%
5%	4.265%	7.533%	0%	98.077%
6%	5.687%	10.59%	0%	100%
7%	9.953%	10.808%	0%	100%
8%	9.953%	15.502%	0%	100%
9%	15.64%	16.594%	0%	100%
10%	13.744%	19.542%	0%	100%

Table 7.9: Table for subtractive noise for University case study

	Violated Constraints	Security Breach	Targets not met	Outlier
1%	0%	0%	6.599%	33.654%
2%	0%	0%	11.168%	59.615%
3%	0%	0%	16.751%	75.962%
4%	0%	0%	22.335%	76.923%
5%	0%	0%	24.366%	83.654%
6%	0%	0%	34.518%	85.577%
7%	0%	0%	40.609%	87.5%
8%	0%	0%	42.132%	87.5%
9%	0%	0%	44.67%	87.5%
10%	0%	0%	50.254%	88.462%

Table 7.10: Table for general noise for University case study

	Violated Constraints	Security Breach	Targets not met	Outlier
1%	1.422%	1.638%	1.015%	44.231%
2%	1.896%	3.057%	3.553%	65.385%
3%	2.37%	4.586%	3.0457%	88.462%
4%	3.318%	5.786%	2.538%	94.231%
5%	3.792%	7.205%	6.091%	100%
6%	6.635%	8.515%	4.569%	98.077%
7%	7.583%	10.590%	4.061%	100%
8%	7.109%	11.354%	6.0914%	100%
9%	8.531%	12.445%	6.599%	100%
10%	9.005%	15.066%	9.137%	100%

not met represents a percentage value of the total number of target privileges that have not been met. The outlier column represents the fraction of the total users that do not have any other users that share the same set of permissions as a percentage, i.e. if a user does not have at least one 100% similarity value, then it is considered an outlier. Each row represents the percentage of noise of the particular type added. Again for the university case study the results for the readings when no noise is added is not included because there were no significant reading for any of the risk metrics.

Here we discuss the results from these tables. The violated constraints value fluctuates a little in both general and additive noise as shown in column one of Tables 7.10 and 7.8 but generally increases with the increase in noise in much the same manner. However, the value constantly stays zero when subtractive noise is added as shown in column one of Table 7.9. The same trend is also noticed in the security breach value as the value increases with the increase in noise in much the same manner in both general and additive as shown in column two of Tables 7.10 and 7.8. However, the value constantly stays zero when subtractive noise is added as shown in column two of Table 7.9. This can be explained by the fact that when the 1's are changed to zeros, the users are deprived of some of their privileges but there have been no violations of privilege in the form of getting access to something that they are not supposed to; thus no matter how many violated constraints are added, and no matter what the security levels are, it does not place any impact on the security breach and the violated constraints value. The fluctuations in the value for violated constraints is explainable by the fact that noise is added randomly to the data and thus may sometimes 'accidentally' help a user to get back their rightful privilege.

The target privileges not met value increases with the increase in noise in subtractive noise as shown in column three of Table 7.9 and fluctuates slightly but generally increases with noise as shown in column three of Table 7.10. However, the value constantly stays zero when additive noise is added as shown in column three of Table 7.8. This can be explained by the fact that when the 0's are changed to ones, there is an over assignment of privileges and the users are getting access to the privileges that they need plus more. Thus, no matter how many over assignments of privileges are made, because it does not affect the deprivation of target privileges, they will always stay zero. The fluctuating value in the general noise is due to the randomness of the noise added.

As the data for the University database is bigger than the telecommunication company data, we are adding more noise. In this study, we are extending our noise level to 10%. We again used Spearmans Rank Correlation on the different risk factors and checked for the 95% and 99% Confidence Interval for the correlation values. Tables 7.11, 7.12 and 7.13 show the correlation and p-value results that we obtained for the different noise types and values. Because the values in the first case study were quite insignificant, only the confidence interval values were calculated and the p-value results were not calculated.

Table 7.11: Correlation for additive noise for University case study

			95%CI		99%CI		
		ρ	lower	Upper	lower	Upper	p-value
Outlier	Violated Constraint	0.87	0.54	0.97	0.35	0.98	0.0005
Outlier	Security Breach	0.92	0.71	0.98	0.57	0.99	6.484E-05
Outlier	Targets not met						0.5
Violated Constraint	Security Breach	0.97	0.84	0.99	0.81	0.81	1.82E-06
Violated Constraint	Targets not met						0.5
Security Breach	Targets not met						0.5

Table 7.12: Correlation for subtractive noise for University case study

			95%CI		99%CI		
		ρ	lower	Upper	lower	Upper	p-value
Outlier	Violated Constraint						0.5
Outlier	Security Breach						0.5
Outlier	Targets not met	0.7755	0.285	0.944	0.061	0.964	0.0042
Violated Constraint	Security Breach						0.5
Violated Constraint	Targets not met						0.5
Security Breach	Targets not met						0.5

The correlation results in the second case study are more statistically significant compared to the first one. None of the results in the general noise category have a 0 value. The additive and subtractive noise have only a few values that are not undefined. This is because of the 0

Table 7.13: Correlation Table for General Noise for University case study

			95%CI		99%CI		
		ρ	lower	Upper	lower	Upper	p-value
Outlier	Violated Constraint	0.90	0.63	0.98	0.47	0.99	0.00018
Outlier	Security Breach	0.90	0.62	0.98	0.46	0.98	0.0002
Outlier	Targets not met	0.85	0.47	0.96	0.27	0.98	0.0009
Violated Constraint	Security Breach	0.96	0.84	0.99	0.75	0.99	5.08E-06
Violated Constraint	Targets not met	0.77	0.28	0.94	0.05	0.96	0.0045
Security Breach	Targets not met	0.83	0.41	0.96	0.202	0.97	0.0016

values of security breach and violated constraints in the additive noise and the 0 values in the target privileges not met part of the subtractive noise. However, based on our results, we can draw some particular inferences. These inferences are made in combination in the next section.

7.3 Overall Performance Comparison of the Two Case Studies

If we compare the results that we found in both case studies we will see that many of the portions give consistent results. This basically shows that the performance of the aRARM tool with respect to the different risk factors is consistent. There were also certain portions of the results that were inconclusive in the smaller telecommunication company data but showed more conclusive results in the University data. Let us discuss all the different results from the telecommunication company data and university data that are either common or disjoint.

- Except for certain exceptions in the university dataset, in both the case studies, the outliers generally increase with the amount of noise, irrespective of the type of noise that is applied.

- With respect to subtractive and additive noise, we can say that the security breach and the target privileges behave opposite to each other. This behavior is seen in both the case studies. Thus the problems with both over assignment and underassignment of privileges remain the same regardless of the case applied.

- While in the first case study, no conclusive results were found regarding the violated constraints, in the university case study we found that the results are similar to the results from the security breach. This means that for violated constraints, an overassignment of privileges is more of a problem than an underassignment of privileges and that when users are overassigned privileges, there is a higher probability of constraints being violated as well.

- In almost all the cases, whenever there is no discrepancy found in the data, no discrepancy is found irrespective of the amount of noise that is added. For example, when 0's are converted to 1's for over assignment of privileges, the target privileges are always met and the over assignment of privileges means that users are getting more privileges than they need but are also always getting the target privileges that they need to work. In the case of underassignment of privileges, the users are getting fewer privileges than they need to function but they are also not having any constraints violated and there is no breach of security.

- There are fluctuations in the results in many cases in both the first and second case study. This is due to the randomness of the noise applied to the case studies.

Based on the combined results of the two case studies we can draw the following null hypotheses:

$H1_0$**:** *The number of outliers is not affected by an increase of noise irrespective of the type of noise applied*

$H2_0$**:** *The number of security breaches is not affected by an increase of additive noise*

$H3_0$**:** *The number of violated constraints is not affected by an increase of additive noise*

$H4_0$**:** *The number of target privileges not met is not affected by an increase of subtractive noise*

We have seen that the p-value for the outlier column is always less than 0.05 with respect to all the other risk factors in all the different reported noise cases. This allows us to reject the null hypothesis $H1_0$ that the number of outliers is not affected. Instead, we can now accept the following alternate hypothesis:

$H1_a$**:** *The number of outliers is affected by an increase of noise irrespective of the type of noise applied*

The p-value for the security breach is less than 0.05 both in the general noise table as well as the additive noise table. This allows us to reject the null hypothesis $H2_0$ that the number of security breaches are not affected. Instead we can now accept the following alternate hypothesis:

$H2_a$**:** *The number of security breaches is affected by an increase of additive noise*

Similarly, the p-value for the violated constraints is less than 0.05 both in the general noise table as well as the additive noise table. This allows us to reject the null hypothesis $H3_0$ that the number of violated constraints is not affected. Instead we can now accept the following alternate hypothesis:

$H3_a$**:***The number of violated constraints is affected by an increase of additive noise*

The p-value for the target privileges not met is less than 0.05 both in the general noise table as well as the subtractive noise table. This allows us to reject the null hypothesis $H4_0$ that the number of target privileges not met is not affected. Instead we can now accept the following alternate hypothesis:

$H4_a$**:** *The number of target privileges not met is affected by an increase of subtractive noise*

7.4 Performance of Case Studies with Different Role Mining Algorithms

We tested both our case studies with four role mining algorithms. These are:

- CompleteMiner [57]
- Fast Exact Heuristics [16]

- Graph Optimization [63]
- Osborn Reid Wesson [45]

The first three algorithms mentioned are well known role mining algorithms. These algorithms have been used extensively in many papers for evaluation ([38]) and other purposes. However, the last algorithm mentioned is from a paper not specifically known for role mining. This paper describes two experiments in interfacing a role based access control model and a relational database. In the first experiment, a role based front end is built as a front end to a relational database with discretionary access control. In the second, a role graph to show the roles present in a standard relational database was built by consulting the privileges relation in the database. The role mining algorithm was implemented as a part of the second experiment. The explicit algorithm used for the implementation for the prototype tool is summarized in Algorithm 1. We will dedicate a subsection to the description of the results that we found from the two case studies.

Algorithm 1 Osborn Reid Wesson Algorithm for role mining

```
Require: Set of users $U$;
 1: Set of permissions $P$;
 2: Set of user permission assignments $u \rightarrow p$;
 3: $clusters$=collection of user permission assignments clustered by user;
 4: while $clusters \neq \emptyset$ do
 5:     $currentCluster = clusters.remove(0)$;
 6:     $matched = false$;
 7:     for each $cluster$ in $clusters$ do
 8:         if $isEqualClusters(currentCluster, cluster)$ then
 9:             merge $currentCluster$ and $cluster$ as a single cluster;
10:             $matched = true$;
11:         end if
12:     end for
13:     if $matched \neq true$ then
14:         add $currentCluster$ to $tempCluster$;
15:     end if
16: end while
17: add $tempCluster$ to $clusters$
```

7.4.1 Results of Different Role Mining Algorithms with the Two Case Studies

The first role mining algorithm that we implemented is the CompleteMiner [57]. This algorithm consists of three phases:

- Identification of Initial Set of Roles: In this phase, we group all users who have the exact same set of permissions. These groups form the initial set of roles, say InitRoles.
- Subset Enumeration: In this phase, all of the potential roles are determined by computing all possible intersection sets of all the roles created in the initial phase. Let this set be GenRoles. Thus if $p1, p2, p3$ are owned by several users, but no users have any subset of those permissions, then only $p1, p2, p3$ is reported as a role.
- User Count Computation: In this phase, for each generated role in GenRoles, a count of the number of users who have the permissions associated with that role is also made. Two sets of counts are made: (i) the original count i, the original number of users who have exactly the set of permissions corresponding to role i and nothing else, and (ii) count(i), an updated count of users whose permissions are a superset of the permissions associated with this role i.

The performance of Complete Miner in both the case studies with regards to noise was poor. Its performance is worst during underassignment of privileges when subtractive noise is added. This is not unexpected as the same thing was reported in [38]. In the evaluation of all the different algorithms as mentioned in [38], for noise and direct assignments, the Complete Miner algorithm just performed better than ORCA [47].

The second role mining algorithm that we implemented is the Fast Exact Heuristics algorithm mentioned in [16]. The authors propose a greedy algorithm that builds a biclique cover by identifying and including one biclique at a time in the cover until all edges are covered. Bicliques are constructed as follows. Take a vertex v with fewest uncovered incident edges, find its neighbors, $\Gamma(v)$, and then find the set $\Xi(\Gamma(v))$ of vertices that are adjacent to all of $\Gamma(v)$. (If v is a user, we find his or her permissions and then find all other users who also have all of these permissions; if on the other hand v is a permission, we find all users that have this permission and then all permissions that all of these users have.) $v \epsilon \Xi(\Gamma(V))$, so $\Xi(\Gamma(V))$ is nonempty and the vertex subset $\Xi(\Gamma(V)) \cup \Gamma(v)$ is a biclique.

The fast exact heuristics algorithm outperformed all the other algorithms in terms of producing the least number of roles. It always produced the least number of roles irrespective of the type or amount of noise that is applied to it. This is in accordance with the evaluation mentioned in [38]. In their experimentation, this algorithm got the highest ranking with respect to noise and direct assignment.

The third role mining algorithm that we implemented is the Graph Optimization algorithm. The algorithm identifies pairs of roles to analyze and performs merge or split operations if they improve the optimization metric of the graph. Pairs of analyzed roles can be separated into four scenarios depending on the permission set overlap of the two roles:

- The roles' permission sets are equal. When two roles are the same, they are merged into one role with their relations updated.
- One role's permission set is a subset of the other role's permission set. When this situation occurs, a link from the superset to the subset is created and outgoing links from the superset role are updated.
- There is overlap between the two role's permission sets and one permission set is not a subset of the other permission set. In this scenario, a role containing the overlap of the two roles is created and two links to this new role are created.
- The two roles' permission sets are distinct. No action is preformed when roles with no overlapping permissions are dealt with.

The paper mentions that the roles must be merged until they are stable. However, this is a rather vague term and elaboration is not made in the paper. As a measure for stability, we have introduced a try limit in the algorithm and the clusters are merged until the number of merges reaches the try limit. In our experiment, we noticed that when the algorithm has reached a particular try limit, increasing the limit further does not improve the number of roles generated and we can assume that the roles generated are stable. However, when the algorithm reaches stability, it generates the same number of roles as the fourth algorithm, i.e. the OsbornReidWesson algorithm. Thus, irrespective of noise amount and type, at its best performance, the graph optimization algorithm usually performs similar to the OsbornReidWesson algorithm. We have seen that with a lower optimization try value, the graph optimization algorithm produces more roles than OsbornReidWesson. This can be seen in Table 7.14. For our experiments, for best performance of the graph algorithm we used an optimization try value of 7000. We also tested the algorithm with a lower optimization try value of 100 and 1000 and with and without noise.

Table 7.14: The performance of the OsbornReidWesson and Zhang algorithms with different try limits. The third and fourth columns represent the number of roles that were generated using the Zhang algorithm and the OsbornReidWesson algorithm, respectively, for different noise types and amounts.

Noise Type and amount	Try Limit (for zhang algorithm only)	Zhang	OsbornReidWesson
No Noise	100	25	17
No Noise	1000	17	17
5% additive noise	100	51	41
5% additive noise	1000	41	41
5% subtractive noise	100	43	30
5% subtractive noise	1000	30	30
5% general noise	100	47	42
5% general noise	1000	45	42

Table 7.15: The number of roles generated with additive noise in Case 1

	CompleteMiner	Ene	Zhang	OsbornReidWesson
1%	27	18	23	23
2%	38	21	29	29
3%	49	24	32	32
4%	58	25	37	37
5%	71	26	38	38

Tables 7.15, 7.16 and 7.17 show the results for the different role mining algorithms for case study 1 on the telecommunication company project. Tables 7.18, 7.19 and 7.20 show

Table 7.16: The number of roles generated with subtractive noise in Case 1

	CompleteMiner	Ene	Zhang	OsbornReidWesson
1%	47	20	22	22
2%	85	23	26	26
3%	147	26	28	28
4%	269	27	29	29
5%	211	30	30	30

Table 7.17: The number of roles generated with general noise in Case 1

	CompleteMiner	Ene	Zhang	OsbornReidWesson
1%	28	19	23	23
2%	51	23	29	29
3%	68	24	33	33
4%	72	25	38	38
5%	90	30	43	43

Table 7.18: The number of roles generated with additive noise in Case 2

	CompleteMiner	Ene	Zhang	OsbornReidWesson
1%	165	54	65	65
2%	306	66	88	88
3%	503	87	100	100
4%	783	111	102	102
5%	860	114	103	103
6%	1308	167	104	104
7%	1868	146	104	104
8%	2410	180	104	104
9%	2838	200	104	104
10%	3686	202	104	104

Table 7.19: The number of roles generated with subtractive noise in Case 2

	CompleteMiner	Ene	Zhang	OsbornReidWesson
1%	385	54	56	56
2%	839	70	79	79
3%	884	66	88	88
4%	911	70	89	89
5%	924	79	93	93
6%	625	87	93	93
7%	628	88	93	93
8%	522	78	91	92
9%	405	73	92	92
10%	327	63	85	85

Table 7.20: The number of roles generated with general noise in Case 2

	CompleteMiner	Ene	Zhang	OsbornReidWesson
1%	147	52	66	66
2%	315	69	82	82
3%	682	93	98	98
4%	794	114	102	102
5%	960	115	104	104
5%	960	115	104	104
6%	2148	150	104	103
7%	1817	162	104	104
8%	2847	153	104	104
9%	2627	176	104	104
10%	3310	170	104	104

the results for the different role mining algorithms for case study 2 on the university RBAC.

In case study 1, all the different algorithms generated fairly lowest number of roles when additive noise was added. This is obvious because a cluster of user permissions is easier to form when there are more permissions available for all. On the contrary, the number of roles generated from each algorithm are the most when the noise is subtractive. This is because now the user permission assignments are dispersed and it is more difficult to cluster them into fewer roles. This is interesting when related to the previous results because it echoes the NP hardness of finding roles. When target privileges are met, the roles may be of an optimum number but there may still be risk of other risk factors such as constraints that have been violated and breaches in security. However, when we do take into account these risk factors, the number of roles increases and the target privileges may not be met. It is a matter of finding the right balance so that all the target privileges are met and the constraints are not violated and security not breached which is quite difficult.

The performance of the CompleteMiner algorithm is usually the worst irrespective of the amount of noise applied. The performance of Zhang and OsbornReidWesson algorithms are usually similar whereas the Ene algorithm performs better than the others in case study 1 results and only sometimes performs better than the others in the case study 2 results.

7.5 Risk Priority Number Calculation Before and After Role Mining

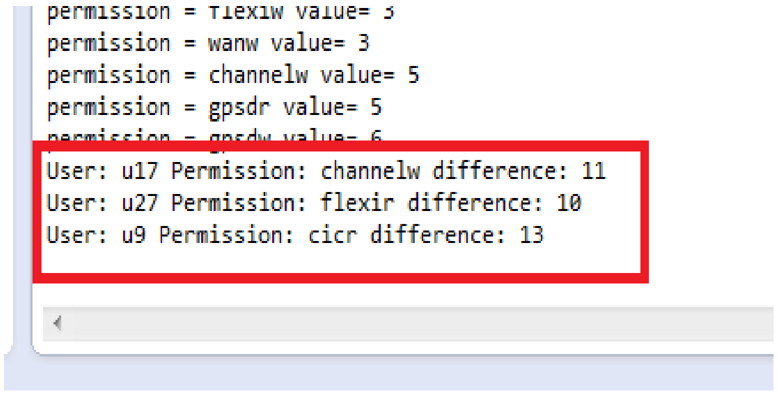
In this section we will discuss the RPN calculation that we determine in our prototype before and after role mining. When the user log information and permission sensitivity level value are available, they are being used to determine the occurrence rating and the severity rating respectively. However, we are also using these two values and the similarity value of the users to get an earlier approximate idea about the Detection Rating (DR). The actual detection rating value is calculated after the roles have been defined.

In our prototype, we calculate an approximation of the DR even before the roles are generated. This ′initial DR′ before role mining is done based on the assumption that a set of users who have a 100% match of permissions (i.e. similarity of 100% with each other) could be considered as a set of users having the same set of roles. Thus, the ′initial DR′ is found for each permission of each user from this cluster. The algorithm for the initial DR calculation before role mining is shown in Algorithm 2. We also calculate the DR after role mining using the method mentioned in [5]. In [5], k-means clustering is used between the different users in each role. Here we describe how 'initial DR' is calculated. Three users $u1$, $u2$ and $u3$ all share the same set of permissions $p1$, $p2$, $p3$. Thus, the set $u1$, $u2$, $u3$ can be considered as a cluster. Now suppose u_1, u_2 and u_3 have used permission p_1 5, 6 and 20 times respectively, the minimum distance between the usage of u_3 of p_1 and the usage of the other users is (20-6)= 14. Considering a threshold of 10, u_3 can be considered to be misusing their privilege. Thus user u_3 is highlighted before role mining.

After role mining, k-means clustering is used on each permission in each role of a particular user. This is done to determine whether the usage of a permission of a user under one role is comparable to the usage of permissions of other users in the same role. Two clusters are used and the differences between the usage of the user of the permission and the average usage in each cluster is saved. The difference values that are greater than the threshold values are highlighted. Through our experiments, we have found that some of these highlighted users are users already highlighted in the ′initial DR′ phase. Thus potentially risky users are highlighted even before role mining.

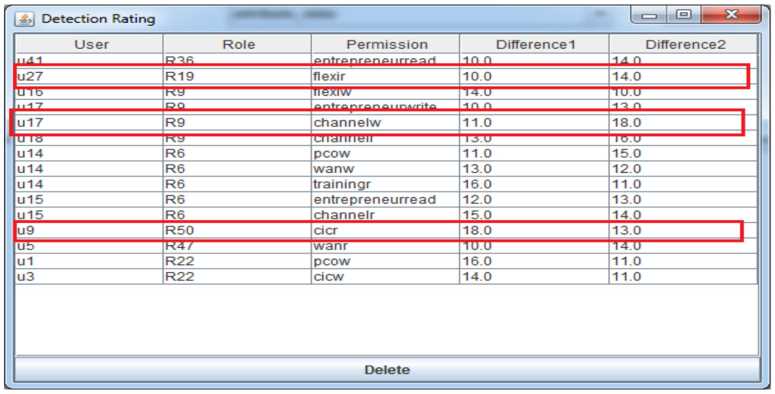
Let us look at an example from the first case study as studied in the tool. We made an assumption that a permission usage difference of greater than or equal to 10 between the different users in a cluster for initial DR calculation could be considered risky. We thus found three user-permission assignments that had a permission usage of 10 or more than the usage of others in the cluster. These are user $u17$ and permission *channelw*, user $u27$ and permission $flexir$ and finally user $u9$ and permission *cicr*. These are all shown in Figure 7.1. Now after role mining, the detection rating is calculated and we decided to look at the results where the difference between usage and cluster value is greater than 10 in both the clusters. These were shown in the report as shown in Figure 7.2. As can be seen in the report, all three user-permission assignments that where detected before role mining have shown up in the report after role mining.

The algorithm for finding the detection rating before role mining is shown in Algorithm 2.



```
permission = flexiw value= 3
permission = wanw value= 3
permission = channelw value= 5
permission = gpsdr value= 5
permission = gpsdw value= 6
User: u17 Permission: channelw difference: 11
User: u27 Permission: flexir difference: 10
User: u9 Permission: cicr difference: 13
```

Figure 7.1: Initial DR calculation before role mining



User	Role	Permission	Difference1	Difference2
u41	R36	entrepreneurread	10.0	14.0
u27	R19	flexir	10.0	14.0
u16	R9	flexiw	14.0	10.0
u17	R9	entrepreneupwrite	10.0	13.0
u17	R9	channelw	11.0	18.0
u18	R9	channelr	13.0	16.0
u14	R6	pcow	11.0	15.0
u14	R6	wanw	13.0	12.0
u14	R6	trainingr	16.0	11.0
u15	R6	entrepreneurread	12.0	13.0
u15	R6	channelr	15.0	14.0
u9	R50	cicr	18.0	13.0
u5	R47	wanr	10.0	14.0
u1	R22	pcow	16.0	11.0
u3	R22	cicw	14.0	11.0

Figure 7.2: DR calculation after role mining

Algorithm 2 Approximate DR assumption before role mining

```
Require: Set of users $U$
Require: Set of permissions $P$
Require: Set of usage frequencies F[$U$][$P$] where $u$ in $U$ and $p$ in $P$
 1: while $U \neq \emptyset$ do
 2:     $currentUser = U[0]$
 3:     $currentPermissionSet$ = permissions of $currentUser$
 4:     find set of users $currentUserSet$ where each user has the exact permissions as
        $currentPermissionSet$
 5:     for each $p$ in $currentPermissionSet$ do
 6:         $min = \infty$
 7:         for each $u$ in $currentUserSet$ do
 8:             if $F[u][p]$-$min \geq threshold$ then
 9:                 highlight $u$ and $p$
10:             end if
11:         end for
12:     end for
13:     $U = U - currentUserSet$
14: end while
```

Chapter 8

Conclusions, Contributions and Future Work

In this thesis, we have added risk awareness to role mining. We have studied the various possible risk factors and categorized them into four general types, which we refer to as risk metrics, in the context of role mining. We have also provided a framework that handles some specific cases of each of these risk metrics before and after role mining. We have implemented a proof-of-concept prototype, a Risk Awareness system for Role Mining (aRARM) based on this framework and applied it to two case studies: a small organizational project and a university database setting. Even though it is possible to detect certain risk metrics such as permission misuse and abuse in clean data; to determine the complete validity of the tool, noise was added to the data. The aRARM prototype is automatically able to detect different types of risk factors when we add different types of noise to this data. The results from the two case studies draws some correlation between the behavior of the different risk factors due to different types and amounts of noise. We draw some hypotheses based on these behaviors. While the detection rating value for calculating the risk priority number has previously been calculated after role mining, we attempt to find an initial estimate of the detection rating before role mining and find that some of the user permission usage information that shows up in the DR calculation after role mining does come up in the initial DR calculation before role mining.

We first defined the four general risk metrics specific for role mining. These risk metrics include Similarity, Compliance to Policies, Trust and Sensitivity, and Permission Misuse and Abuse. We also defined a framework that is able to incorporate the risk metrics at different phases before and after role mining in the pre and post role mining stages. The structure also discusses the different types of input that better aid in the determination of the different risk factors. Next we discussed the proof of concept prototype, a Risk Awareness during Role Mining (aRARM) system that we implemented. We also discussed the scope and the different features of the tool such as the detailed report in each step and an overall report, visualization of the similarity index, shared attributes, etc. We tested our prototype tool on two case studies namely a small project in a large organization and a university database. The value of the different risk factors were noted while we added general, additive and subtractive noise between 1% and 10%. The results and interpretations of the two case studies allowed us to conclude that the outliers always increase with increase in noise irrespective of the type of noise applied. We also noticed that the number of violated constraints and security breaches increases with the increase in both the general and additive noise. The value of target privileges not met increases with the increase in general as well as subtractive noise.

8.1 Contributions

In this section we highlight the several contributions that we have made in this research:

- There are a few examples of work in role mining algorithms that take into consideration risk, but there is no work that categorizes the various risk metrics before and after role mining. An important contribution of this work is the definition of these risk metrics for role mining.
- Based on the risk metrics that we have defined in this thesis, we have also developed a framework that incorporates the risk metrics into the role mining process.

- We have developed a proof-of-concept prototype, a Risk Awareness system for Role Mining (aRARM) based on the framework that we have defined.
- We have applied the proof of concept prototype to two case studies: a small organizational project and a university database setting to test the usability, effectiveness and validity of the tool.
- The proof of concept prototype that we have developed is unique in its flexibility towards role mining algorithms. It is one of the few role mining tools that is unbiased towards the role mining algorithm to be used. In this tool, any role mining algorithm of choice can be used to generate roles. This is because the main focus of our work was on risk minimization and not the quality of the generated roles.
- One of the metrics that we have defined is permission misuse and abuse in which we can use the Risk Priority Number formula as used in [5]. However, we have implemented an algorithm to determine an approximate value of the Detection Rating in the Risk Priority Number formula even before the roles have been generated.
- Our results from the case studies have given us some understanding of the interactions of additive and subtractive noise and the various risk factors, from a very high level. We have shown through the two case studies that in general, if a user has too many permissions then there is a risk of security breach and violation of constraints, and if they are missing permissions then there is a risk of target permissions not being met.

8.2 Practical Significance of the Research

From a practitioners' perspective, the prototype would be very useful in the reduction of risk in user-permission assignments before roles are mined and also in the management of roles after the roles have been mined. The tool can be used to periodically make checks for any potential risks in the roles due to permission misuse/ abuse, constraint violation when old inactive users have not been deleted from the system, permissions exceptionally or accidentally granted or denied, etc. It can also at the same time be used to generate a new RBAC with its built-in role mining platform and to compare the new RBAC with the previous one for any discrepancy. From a researchers' perspective, the risk metrics and the current framework could be slightly redefined for application to other access control models. The risk awareness framework could be used to carry out further research on role mining such as: the effect of different risk metrics on the access control model and whether the risk metrics are related to one another in any way.

8.3 Future Work

There is a lot of potential future research that can be conducted based on our findings. In this section we look at some of the future research that can be conducted, some of which is out of the scope of our current research, and also some potential future research based on the interpretations of our findings of the results.

- Addition of a noise filtering algorithm during role mining. In our current research we added noise to actual data just to check whether the different risk factors are effectively detected in our proof of concept prototype. However, a more stable version of the prototype could have some built in role mining algorithm that filters the noise before mining the roles such as the one in [39]. This would greatly improve the effectiveness of the tool.

- Addition of session information and dynamic separation of duty constraints. In our current prototype we only dealt with the static separation of duty constraints and did not take into consideration sessions, dynamic separation of duty constraints and many other types of policies such as those dealing with workflow. Adding these types of policies would make the tool more suitable for a realistic use in an organization where these policies are more common.

- Application of both static and dynamic risk thresholds. For a static risk threshold, the risk value is compared against a predefined threshold, while in a dynamic setting, the risk threshold may change based on activity in a session, location and other possible environmental factors. In our current research we did not take into consideration any types of risk thresholds because we assumed that this will vary based on different organizations. However, addition of a dynamic risk threshold that changes based on the users' usage would make it more effective in detecting whether a threshold has been crossed or not.

- As our current data set is quite small, we are hoping to carry out a study on our tool using real world data from a large size organization. This is a very important potential future research for this work. While we did try our best with the two case studies and even though the results seemed to be statistically significant based on the confidence interval and p values, the small sample size could mean that there could be a potential type I or type II error in the results. A possible future research could be the usage of the tool in a large industrial data and then having a domain expert make assumptions about the potential risk of the data. Such a data could be then used to do more statistical tests on the tool such as k-factor analysis.

- We plan to explore the effect of different role mining algorithms on the different risk factors and to check whether certain role mining algorithms are best at dealing with certain risk factors. In our current research, we have only implemented four role mining algorithms and used them to find the roles. However, we did not look into the effect of the different role mining algorithms in the determination of the different risk factors. This could be a potential future research. For example, say, if we found out that the Complete Miner is particularly useful to mine roles when there is additive noise as compared to other role mining algorithms and can better deal with outliers than other algorithms then this could be useful for an organization to choose an algorithm based on their need.

- In our work, we have considered the risk metrics determination and their reduction as a means to deal with potential threats. However, there is work in the literature that discusses other means to deal with risk such as the work in [37]. They propose the Quantified Risk Based Access Control System that evaluates the request for an access and quantifies the risk associated with the access. The system then provides the user with an access ticket describing the access, and indicating the request price in risk tokens. The user evaluate the price; if they have enough risk tokens and agree to the price, they will purchase access to the information from the database which will return the information requested in the access ticket. Our tool could possibly be extended to include this interesting feature for risk awareness.

- In our scope section we mentioned that we are currently only dealing with whether target privileges have been met in the system. However, if we are to consider the entire concept of least privilege principle, we must make sure that non-target privileges are assigned as little as possible. This could be a possible point of research by implementing a feature in the tool to check the extra non-target privileges that have been assigned and trying to minimize it. An algorithm has been proposed in [23] for the detection and minimization of extra non-target privileges during role mining.

The results that we found from our case studies gave us some conclusive as well as some inconclusive information. Further research could be done on these portions for more conclusive results. For example, based on our finding regarding the security breach, violated constraints and target privileges not met, we note that both security breach and violated constraints seem to have the same type of inverse relationship with the target privileges not met. We can express this as a possible hypothesis and further research to be carried out.

H5: *The violated constraints and target privileges not met are negatively correlated*

H6: *The security breach and target privileges not met are negatively correlated*

Bibliography

[1] Trusted computer system evaluation criteria (orange book). Technical report, U.S. Department of Defense, 1985.

[2] Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. TRBAC: A Temporal Role-based Access Control Model. *ACM Trans. Inf. Syst. Secur.*, 4(3):191–233, August 2001.

[3] Khalid Zaman Bijon, Ram Krishnan, and Ravi Sandhu. Risk-aware RBAC sessions. In Venkat Venkatakrishnan and Diganta Goswami, editors, *Information Systems Security*, volume 7671, pages 59–74, 2012.

[4] K.Z. Bijon, R. Krishnan, and R. Sandhu. A framework for risk-aware role based access control. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 462–469, Oct 2013.

[5] Ebru Celikel, Murat Kantarcioglu, Xiaohu Li, and Elisa Bertino. A Risk Management Approach to RBAC. *Risk and Decision Analysis*, 1(2), November 2009.

[6] Suresh Chari, Jorge Lobo, and Ian Molloy. Practical risk aggregation in RBAC models. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, pages 117–118, 2012.

[7] Suresh Chari, Ian Molloy, Youngja Park, and Wilfried Teiken. Ensuring Continuous Compliance Through Reconciling Policy with Usage. In *Proceedings of the 18th ACM*

Symposium on Access Control Models and Technologies, SACMAT '13, pages 49–60. ACM, 2013.

[8] Pau Chen Cheng, P. Rohatgi, C. Keser, P.A. Karger, G.M. Wagner, and A.S. Reninger. Fuzzy multi-level security: An experiment on quantified risk-adaptive access control. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 222–230, May 2007.

[9] A. Colantonio, R. Di Pietro, A. Ocello, and N.V. Verde. Visual Role Mining: A Picture Is Worth a Thousand Roles. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6):1120–1133, 2012.

[10] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. A cost-driven approach to role engineering. In *Proceedings of the 2008 ACM symposium on Applied computing*, SAC 08, pages 2129–2136, New York, NY, USA, 2008. ACM.

[11] 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, New York, NY, USA, 2009. ACM.

[12] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Evaluating the risk of adopting RBAC roles. In *Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*, DBSec'10, pages 303–310. Springer-Verlag, 2010.

[13] 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. *Decis. Support Syst.*, 50(4):715–731, 2011.

[14] Alessandro Colantonio, RobertoDi Pietro, and Alberto Ocello. Leveraging Lattices to Improve Role Mining. In Sushil Jajodia, Pierangela Samarati, and Stelvio Cimato, editors, *Proceedings of The Ifip Tc 11 23rd International Information Security Conference*,

volume 278 of *IFIP The International Federation for Information Processing*, pages 333–347. Springer US, 2008.

[15] Edward J. Coyne. Role engineering. In *Proceedings of the first ACM Workshop on Role-based access control*, RBAC '95, New York, NY, USA, 1996. ACM.

[16] 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, New York, NY, USA, 2008. ACM.

[17] David Ferraiolo and Richard Kuhn. Role-based access control. In *In 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[18] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST Standard for Role-based Access Control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, August 2001.

[19] Mario Frank, Joachim M. Buhmann, and David Basin. On the definition of role mining. In *Proceedings of the 15th ACM symposium on Access control models and technologies*, SACMAT '10, pages 35–44, New York, NY, USA, 2010. ACM.

[20] Mario Frank, Andreas P. Streich, David Basin, and Joachim M. Buhmann. A Probabilistic Approach to Hybrid Role Mining. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 101–111, New York, NY, USA, 2009. ACM.

[21] Mario Frank, Andreas P. Streich, David Basin, and Joachim M. Buhmann. Multi-assignment Clustering for Boolean Data. *J. Mach. Learn. Res.*, 13(1):459–489, February 2012.

[22] Chris Giblin, Marcel Graf, Günter Karjoth, Andreas Wespi, Ian Molloy, Jorge Lobo, and Seraphin Calo. Towards an Integrated Approach to Role Engineering. In *Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration*, SafeConfig '10, pages 63–70, New York, NY, USA, 2010. ACM.

[23] Hejiao Huang, Feng Shang, and Jiangtao Zhang. Approximation Algorithms for Minimizing the Number of Roles and Administrative Assignments in RBAC. In *Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*, COMPSACW '12, pages 427–432, Washington, DC, USA, 2012. IEEE Computer Society.

[24] John C. John, Shamik Sural, Vijayalakshmi Atluri, and Jaideep S. Vaidya. Role Mining under Role-Usage Cardinality Constraint. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *Information Security and Privacy Research*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 150–161. Springer Berlin Heidelberg, 2012.

[25] James B D Joshi, E. Bertino, U. Latif, and A Ghafoor. A generalized temporal role-based access control model. *Knowledge and Data Engineering, IEEE Transactions on*, 17(1):4–23, Jan 2005.

[26] S. Kandala, R. Sandhu, and V. Bhamidipati. An Attribute Based Framework for Risk-Adaptive Access Control Models. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 236–241, Aug 2011.

[27] Hemanth Khambhammettu, Sofiene Boulares, Kamel Adi, and Luigi Logrippo. A framework for threat assessment in access control systems. In *Information Security and Privacy Research,*, pages 187–198. Springer Berlin Heidelberg, 2012.

[28] Martin Kuhlmann, Dalia Shohat, and Gerhard Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *Proceedings of the*

eighth ACM symposium on Access control models and technologies, SACMAT '03, pages 179–186, New York, NY, USA, 2003.

[29] Ninghui Li, Ziad Bizri, and Mahesh V. Tripunitara. On mutually-exclusive roles and separation of duty. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 42–51, New York, NY, USA, 2004. ACM.

[30] StatsDirect Ltd. StatsDirect Statistical Software. `http://www.statsdirect.com/`, 2014. [Online; accessed 16-July-2014].

[31] Haibing Lu, J. Vaidya, and V. Atluri. Optimal boolean matrix decomposition: Application to role engineering. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 297–306, April 2008.

[32] Adam Lund and Mark Lund. Spearman's Rank-Order Correlation - A guide to when to use it, what it does and what the assumptions are. `https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php`, 2014. [Online; accessed 16-July-2014].

[33] Xiaopu Ma, Ruixuan Li, and Zhengding Lu. Role Mining Based on Weights. In *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies*, SACMAT '10, pages 65–74, New York, NY, USA, 2010. ACM.

[34] Xiaopu Ma, Ruixuan Li, Zhengding Lu, and Wei Wang. Mining constraints in role-based access control. *Mathematical and Computer Modelling*, 55(12):87 – 96, 2012. Advanced Theory and Practice for Cryptography and Future Security.

[35] Barsha Mitra, Shamik Sural, Vijayalakshmi Atluri, and Jaideep Vaidya. Toward mining of temporal roles. In *Proceedings of the 27th international conference on Data and Applications Security and Privacy XXVII*, DBSec'13, pages 65–80, Berlin, Heidelberg, 2013. Springer-Verlag.

[36] 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, New York, NY, USA, 2008. ACM.

[37] Ian Molloy, Pau-Chen Cheng, and Pankaj Rohatgi. Trading in risk: using markets to improve access control. In *Proceedings of the 2008 workshop on New Security Paradigms*, pages 107–125. ACM, 2008.

[38] 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, New York, NY, USA, 2009. ACM.

[39] 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, New York, NY, USA, 2010. ACM.

[40] Ian Molloy, Jorge Lobo, and Suresh Chari. Adversaries' holy grail: access control analytics. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, BADGERS '11, pages 54–61, New York, NY, USA, 2011. ACM.

[41] Ian Molloy, Youngja Park, and Suresh Chari. Generative models for access control policies: applications to role mining over logs with attribution. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 45–56, New York, NY, USA, 2012. ACM.

[42] Gustaf Neumann and Mark Strembeck. An approach to engineer and enforce context constraints in an RBAC environment. In *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, SACMAT '03, pages 65–79. ACM, 2003.

[43] Nimal Nissanke and Etienne J. Khayat. Risk based security analysis of permissions in RBAC. In *Proceedings of the 2 nd International Workshop on Security In Information Systems, Security In Information Systems*, pages 332–341. INSTICC Press, 2004.

[44] Matunda Nyanchama and Sylvia Osborn. The role graph model and conflict of interest. *ACM Trans. Inf. Syst. Secur.*, 2(1):3–33, February 1999.

[45] Sylvia L. Osborn, Laura K. Reid, and Gregory J. Wesson. On the interaction between role-based access control and relational databases. In *Proceedings of the tenth annual IFIP TC11/WG11.3 international conference on Database security: volume X : status and prospects: status and prospects*, pages 275–287, London, UK, UK, 1997. Chapman & Hall, Ltd.

[46] F. Salim, J. Reid, E. Dawson, and U. Dulleck. An Approach to Access Control under Uncertainty. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 1–8, Aug 2011.

[47] Jürgen Schlegelmilch and Ulrike Steffens. Role mining with ORCA. In *Proceedings of the tenth ACM symposium on Access control models and technologies*, SACMAT '05, pages 168–176, New York, NY, USA, 2005. ACM.

[48] Basit Shafiq, Jaideep S. Vaidya, Arif Ghafoor, and Elisa Bertino. A framework for verification and optimal reconfiguration of event-driven role based access control policies. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 197–208, New York, NY, USA, 2012. ACM.

[49] Gary Stoneburner, Alice Y. Goguen, and Alexis Feringa. Sp 800-30. risk management guide for information technology systems. Technical report, Gaithersburg, MD, United States, 2002.

[50] 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 Sym-*

posium on Access Control Models and Technologies, SACMAT '10, pages 55–64, New York, NY, USA, 2010. ACM.

[51] William M. K. Trochim. *The Research Methods Knowledge Base*. Atomic Dog Publishing, 2001.

[52] Emre Uzun, Vijayalakshmi Atluri, Haibing Lu, and Jaideep Vaidya. An optimization model for the extended role mining problem. In *Proceedings of the 25th annual IFIP WG 11.3 conference on Data and applications security and privacy*, DBSec'11, pages 76–89, Berlin, Heidelberg, 2011. Springer-Verlag.

[53] J. Vaidya, V. Atluri, J. Warner, and Qi Guo. Role engineering via prioritized subset enumeration. *Dependable and Secure Computing, IEEE Transactions on*, 7(3):300–314, July 2010.

[54] 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, New York, NY, USA, 2007. ACM.

[55] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Nabil Adam. Migrating to optimal RBAC with minimal perturbation. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, SACMAT '08, pages 11–20, New York, NY, USA, 2008. ACM.

[56] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Haibing Lu. Role mining in the presence of noise. In *Proceedings of the 24th annual IFIP WG 11.3 working conference on Data and applications security and privacy*, DBSec'10, pages 97–112, Berlin, Heidelberg, 2010. Springer-Verlag.

[57] 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, New York, NY, USA, 2006. ACM.

[58] Nino Vincenzo Verde, Jaideep Vaidya, Vijay Atluri, and Alessandro Colantonio. Role engineering: from theory to practice. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*, CODASPY '12, pages 181–192, New York, NY, USA, 2012. ACM.

[59] Zhongyuan Xu and Scott D. Stoller. Algorithms for mining meaningful roles. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 57–66, New York, NY, USA, 2012. ACM.

[60] Zhongyuan Xu and Scott D. Stoller. Mining Parameterized Role-based Policies. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, pages 255–266, New York, NY, USA, 2013. ACM.

[61] D. Zhang, K. Ramamohanarao, T. Ebringer, and T. Yann. Permission Set Mining: Discovering Practical and Useful Roles. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pages 247–256, Dec 2008.

[62] D. Zhang, K. Ramamohanarao, S. Versteeg, and Rui Zhang. RoleVAT: Visual Assessment of Practical Need for Role Based Access Control. In *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pages 13–22, 2009.

[63] 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, New York, NY, USA, 2007. ACM.

Appendix A

Required Relational Database Tables

We have used two different databases based on the two case studies, cic and university_database. The tables in both of these databases are identical and only the dataset is different. Here we describe each of the tables that are in each of these databases.

- **attribute_data:** This table contains information about the attributes of the users.
- **detection_rating:** This table saves the detection rating information after role mining. The table contains user id, role id, permission id and the difference values between the two clusters.
- **noise_additive_user_permission_assignment:** This table stores the user-permission assignment information after additive noise has been added to it.
- **noise_subtractive_user_permission_assignment:** This table stores the user-permission assignment information after subtractive noise has been added to it.
- **noise_general_user_permission_assignment:** This table stores the user-permission assignment information after general noise has been added to it.
- **original_roles:** This table contains information about any existing RBAC. The table contains role id and permission id information.

- **permissions:** This is the permissions table and mainly contains the permission id as well as any additional information related to permission such as creation date, modification date, status, etc.

- **permission_permission_constraint:** If there is any permission-permission constraint information available then it is saved in this table.

- **permission_sensitivity_level:** The permission sensitivity information is saved in this table. The permission id and a corresponding sensitivity rating between 1 to 10 is saved with 1 referring to lowest sensitivity and 10 referring to highest sensitivity.

- **query_log:** The user access log information is saved in this table. It has the user id, permission id and the number of times the particular permission id has been accessed in a particular time.

- **role_role_constraint:** This table is used after the roles have been mined and new roles needs to added or updates need to be made. Role-role constraints are saved in this table.

- **role_similarity:** If there is an existing RBAC, then the jaccard coefficient is used to find the similarity between new and deployed roles. This similarity value is saved in this table.

- **similarity:** This table stores the similarity between users found using the jaccard coefficient.

- **static_role_permission_constraint:** This table is used after the roles have been mined and updates need to be made. The static role permission constraints are saved in this table.

- **static_user_role_constraint:** This table is used after the roles have been mined and updates need to be made. The static user role constraints are saved in this table.

- **target_privileges:** If there is information available about target privileges then these are saved here. This table contains the user id and corresponding permission id that the user must have to function properly.
- **temp_orsr_data:** This table saves the product of the occurrence rating and severity rating that is calculated before role mining.
- **test_role_child_assignment:** Once roles have been mined, the senior junior relationship between the roles are saved in this table.
- **test_role_permission_assignment:** Once roles have been mined, the role-permission assignment information is saved in this table.
- **test_user_role_assignment:** Once roles have been mined, the user-role assignment information is saved in this table.
- **users:** This is the users table and mainly contains the user id as well as any additional information related to users such as creation date, modification date, status, etc.
- **user_permission_assignment:** This table contains the user-permission assignment information.
- **user_permission_constraint:** If there is any available constraints between the user and permissions then these are saved in this table.
- **user_trust_level:** The user trust level information is saved in this table. The user id and a corresponding trustworthiness rating between 1 to 10 is saved with 1 referring to lowest sensitivity and 10 referring to highest sensitivity.
- **user_user_constraint:** If there is any user-user constraint information available then it is saved in this table.

Appendix B

User Manual

In this Appendix, we provide a user manual for the operation of the tool. The tool was implemented in Java with a MySQL database. To run the aRARM tool, we first need to start the Apache Server. In our case we have used the XAMPP server. After starting the server we run the tool. In our case we used the eclipse integrated development environment for the tool. The screen shown in Figure B.1 will appear:

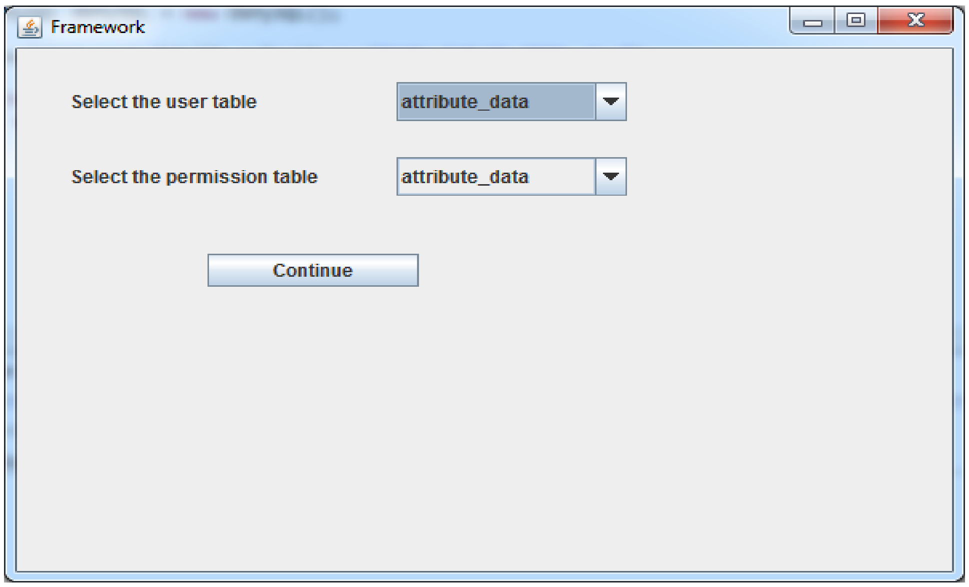


Figure B.1: Initial window of the framework

Select the user and the permission table and continue on to the next window as shown in

Figure B.2.

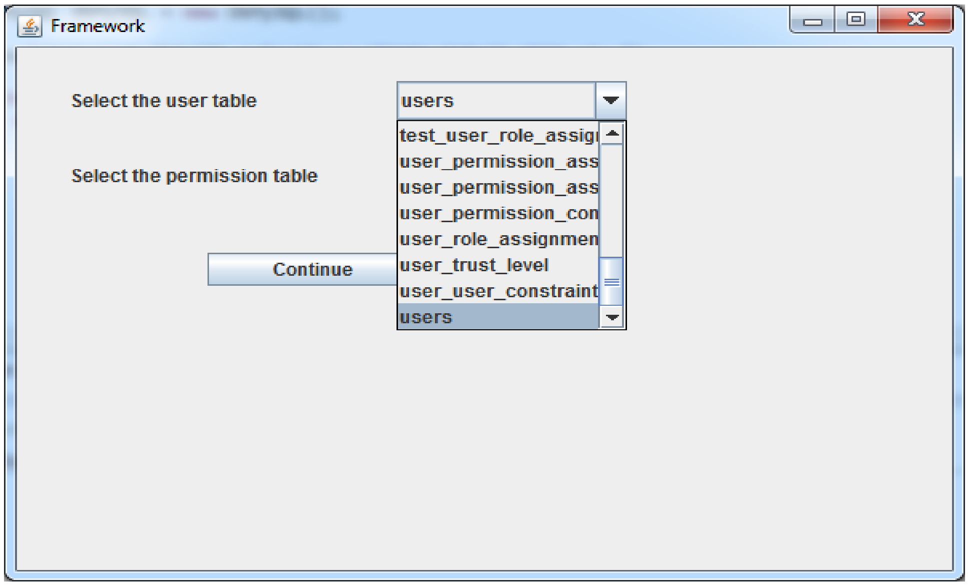


Figure B.2: Selecting users and permissions

In this window, you have to select the table that represents the user permission assignments as well as the user-permission constraints. Once selected press the ’Apply’ button. Once we press the apply button we may choose to not check for any violated constraint and just move on to the next window. In that case, the ’Skip this Step and Continue’ button needs to be pressed. Otherwise we can select the user permission constraints and press ’View Report’ button circled in red in Figure B.3.

If the violated constraints report shows empty as shown in Figure B.5, then this means that no constraints have been violated. In such a case, no action is necessary.

However, if there are a number of constraints that have been violated as shown in the Figure B.6, then action needs to be taken. Select the violations and press the delete button as highlighted in red in the figure. This way, the user-permission assignment where the user has access to the privilege that is actually a violation will be deleted.

Once this is done, move on to the next window, which is the similarity window. Select the box for computing the similarity. You may wish to either view the similarity in the visualization or save the similarity information and move on to the next window. If you view the

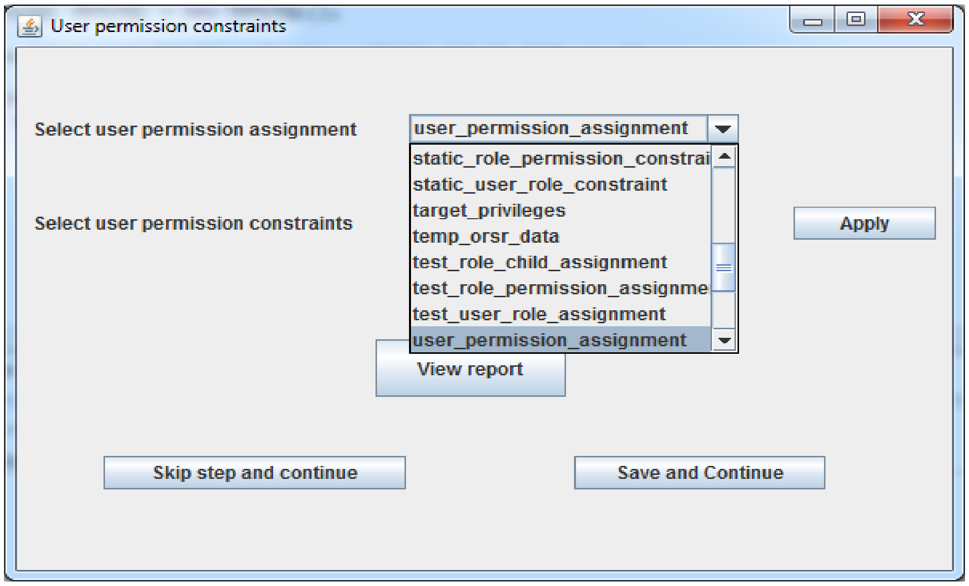

User permission constraints
Select user permission assignment
user_permission_assignment
static_role_permission_constrai
static_user_role_constraint
target_privileges
temp_orsr_data
test_role_child_assignment
test_role_permission_assignme
test_user_role_assignment
user_permission_assignment
Select user permission constraints
Apply
View report
Skip step and continue
Save and Continue

Figure B.3: Selecting user-permission constraints

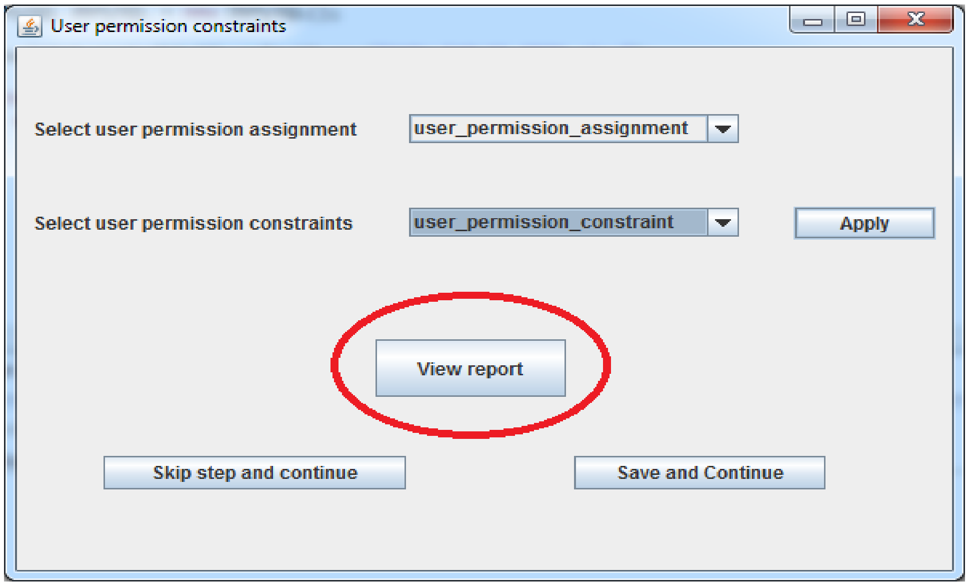

User permission constraints
Select user permission assignment
user_permission_assignment
Select user permission constraints
user_permission_constraint
Apply
View report
Skip step and continue
Save and Continue

Figure B.4: Button to click to view report of the violated constraints

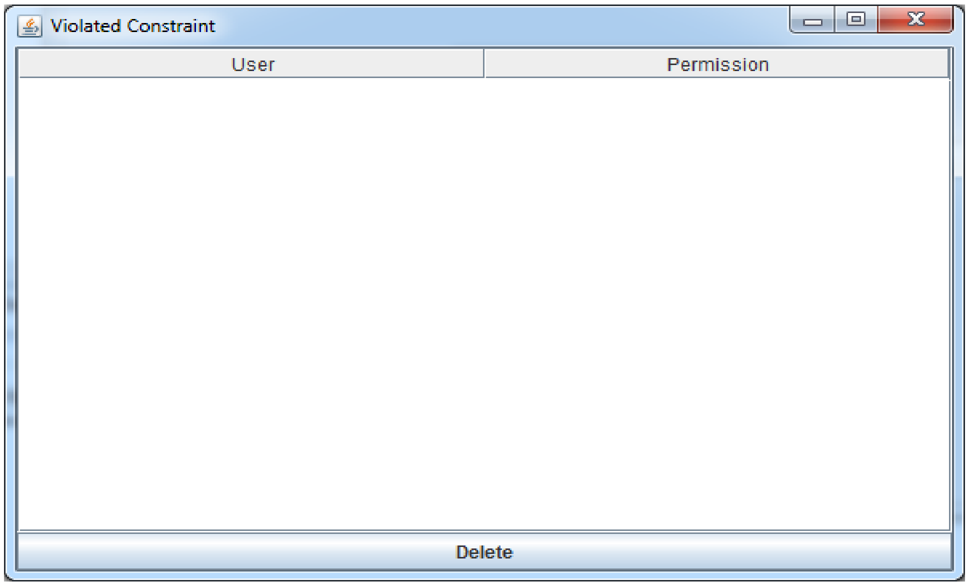
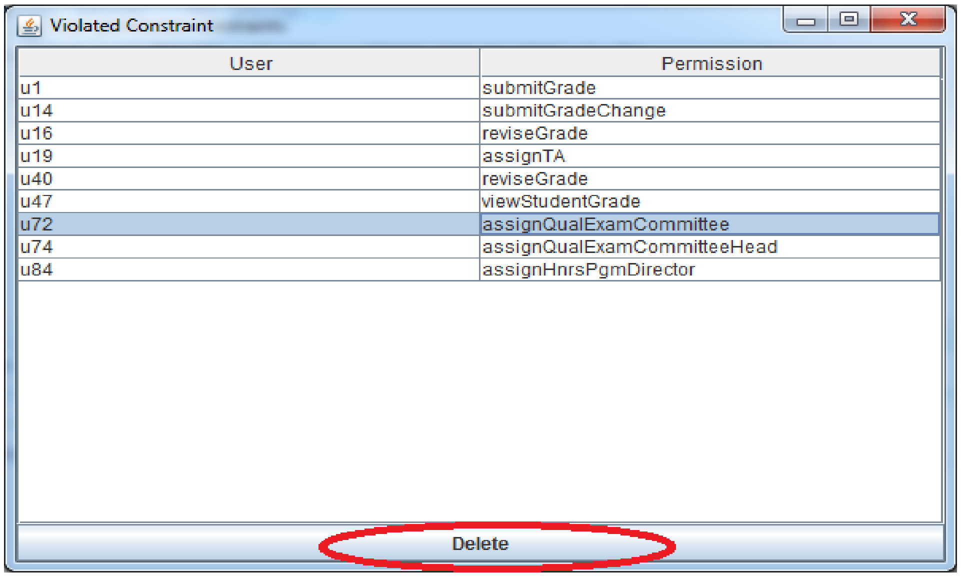


Figure B.5: Violated constraints report with no violations



User	Permission
u1	submitGrade
u14	submitGradeChange
u16	reviseGrade
u19	assignTA
u40	reviseGrade
u47	viewStudentGrade
u72	assignQualExamCommittee
u74	assignQualExamCommitteeHead
u84	assignHnrsPgmDirector

Figure B.6: Violated constraints report with violations

visualization, it will show the visualization and attributes in parallel windows. The legend for the different colors of the circles are also given in Figure B.7. Here red represents a similarity of less than 20 whereas purple represents a similarity of greater than 80.

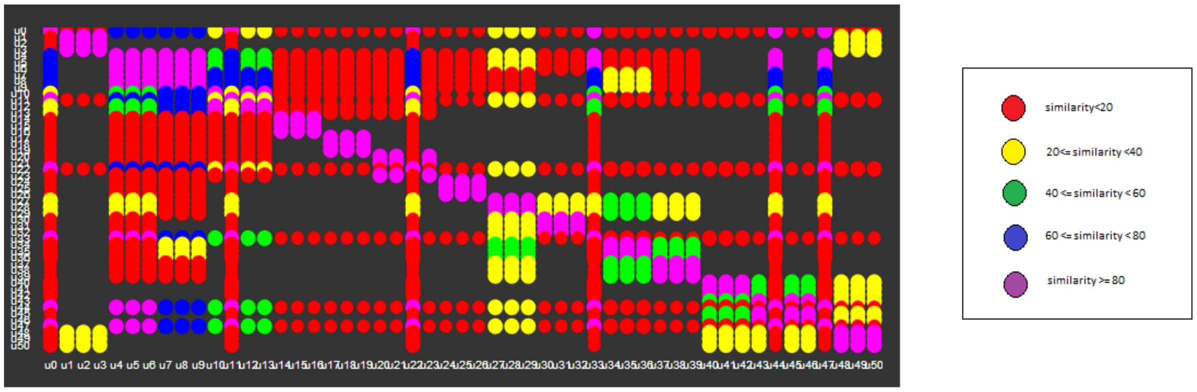


Figure B.7: Visualization of the similarity between users

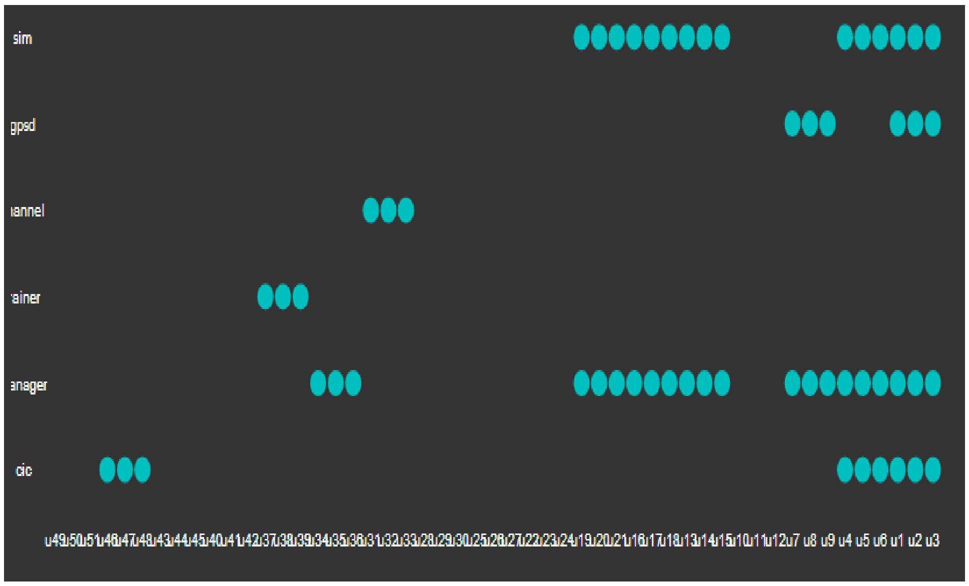
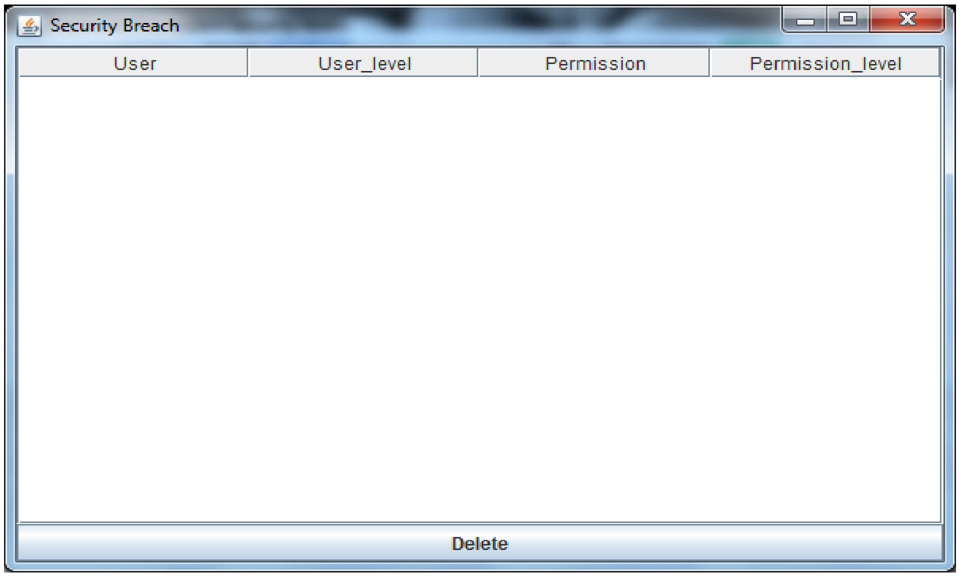


Figure B.8: Visualization of the attributes shared by the users

Next we move on to the next window that shows the user and permission security levels. If this information is available, then we need to select the table containing the user trust level and the permission security level information. Next, the boxes to compute trust and sensitivity need to be checked. Finally the box for checking the security label compatibility needs to be checked. Once this button is pressed, the report for the security label compatibility can be viewed by pressing the ’View Report’ button. If the security breach report shows empty as shown below, then this means that there has been no security breach. In such a case, no action is necessary.



User	User_level	Permission	Permission_level

Figure B.9: Security breach report with no breaches

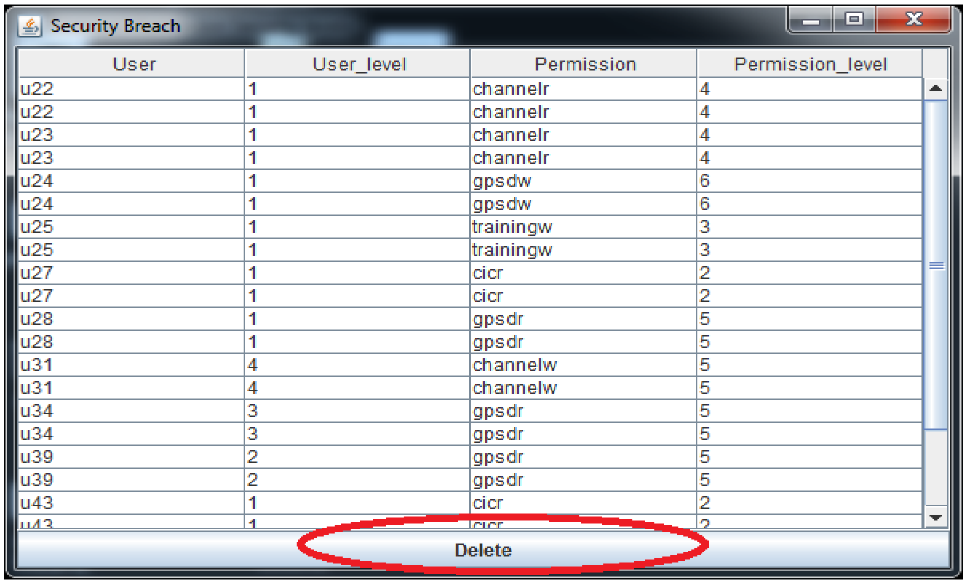
However, if there are a number of security breaches as shown in Figure B.10, then action needs to be taken. Select the breaches and press the delete button as highlighted red in the figure. This way, the user-permission assignment where the user has access to the privilege that is actually a security breach will be deleted.

If the user and permission security level information is not available, then this step can be skipped and we can move on to the next step.

In the next window, we can compute the Occurrence Rating (OR) and Severity Rating (SR) values and show them in a report. Select the table that contains the user access log information and choose to compute the OR and SR.

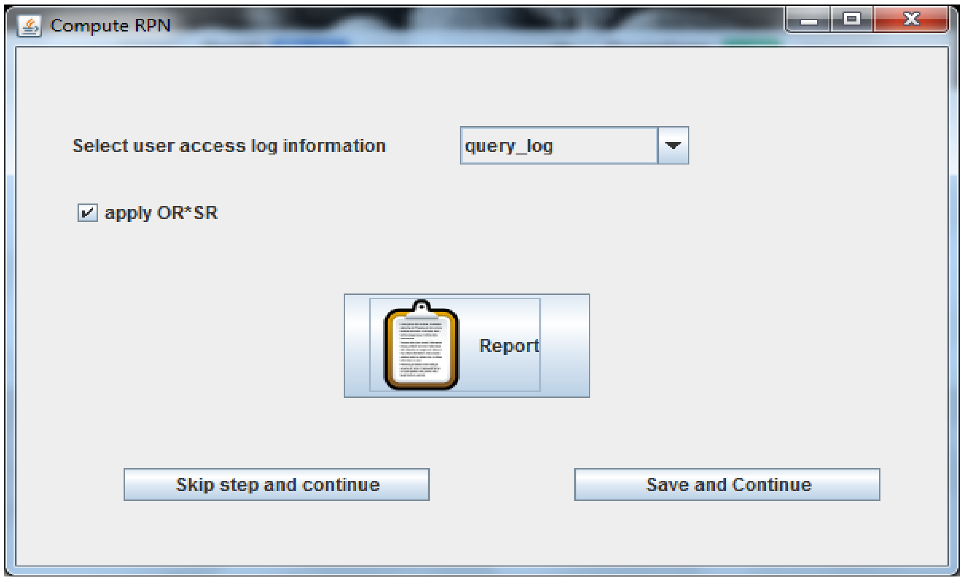
You can press the report button to view the report for the OR and SR values as shown in Figure B.12.

Once we have passed though the RPN report window, we have come to the end of the pre role mining stages. In the next window, you can take an overall look at all the different reports



User	User_level	Permission	Permission_level
u22	1	channelr	4
u22	1	channelr	4
u23	1	channelr	4
u23	1	channelr	4
u24	1	gpsdw	6
u24	1	gpsdw	6
u25	1	trainingw	3
u25	1	trainingw	3
u27	1	cicr	2
u27	1	cicr	2
u28	1	gpsdr	5
u28	1	gpsdr	5
u31	4	channelw	5
u31	4	channelw	5
u34	3	gpsdr	5
u34	3	gpsdr	5
u39	2	gpsdr	5
u39	2	gpsdr	5
u43	1	cicr	2
u43	1	cicr	2

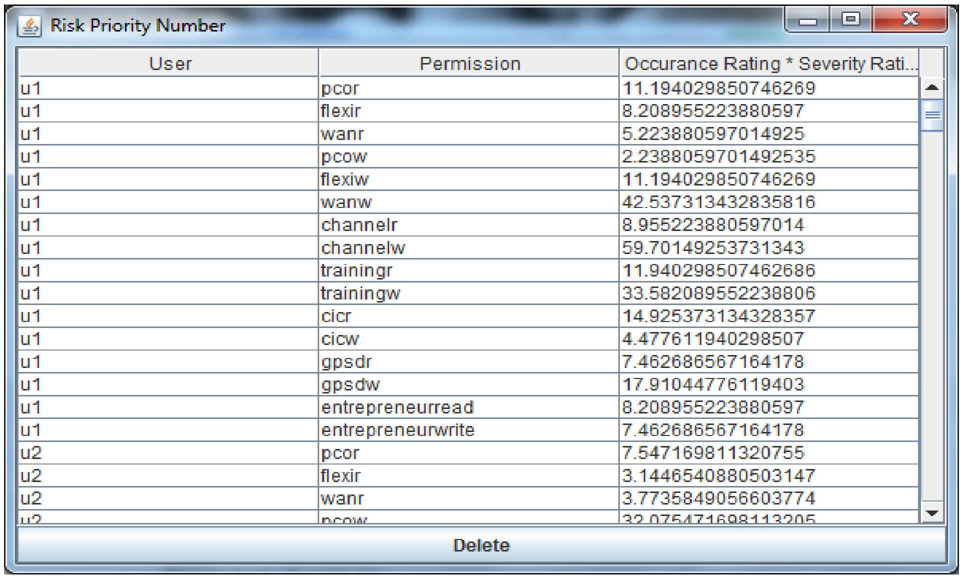
Figure B.10: Security breach report with breaches



Select user access log information: query_log

[✓] apply OR*SR

Figure B.11: Window to select table for user access log information and compute occurrence rating and severity rating values



User	Permission	Occurance Rating * Severity Rati...
u1	pcor	11.194029850746269
u1	flexir	8.208955223880597
u1	wanr	5.223880597014925
u1	pcow	2.2388059701492535
u1	flexiw	11.194029850746269
u1	wanw	42.537313432835816
u1	channelr	8.955223880597014
u1	channelw	59.70149253731343
u1	trainingr	11.940298507462686
u1	trainingw	33.582089552238806
u1	cicr	14.925373134328357
u1	cicw	4.477611940298507
u1	gpsdr	7.462686567164178
u1	gpsdw	17.91044776119403
u1	entrepreneurread	8.208955223880597
u1	entrepreneurwrite	7.462686567164178
u2	pcor	7.547169811320755
u2	flexir	3.1446540880503147
u2	wanr	3.7735849056603774
u2	pcow	32.075471698113205

Figure B.12: User, permission and their occurrence rating and severity rating product report

generated in all the previous windows in an overall dashboard. This can be done by clicking on the report button in the window shown in Figure B.13.

If you do not have access to the user access log information then you can skip this step and continue on to the next step.

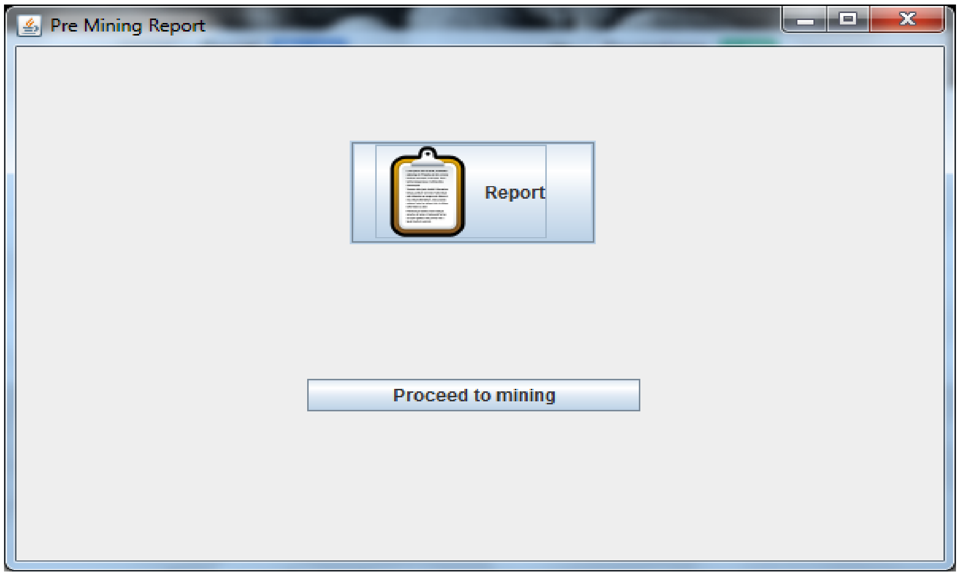
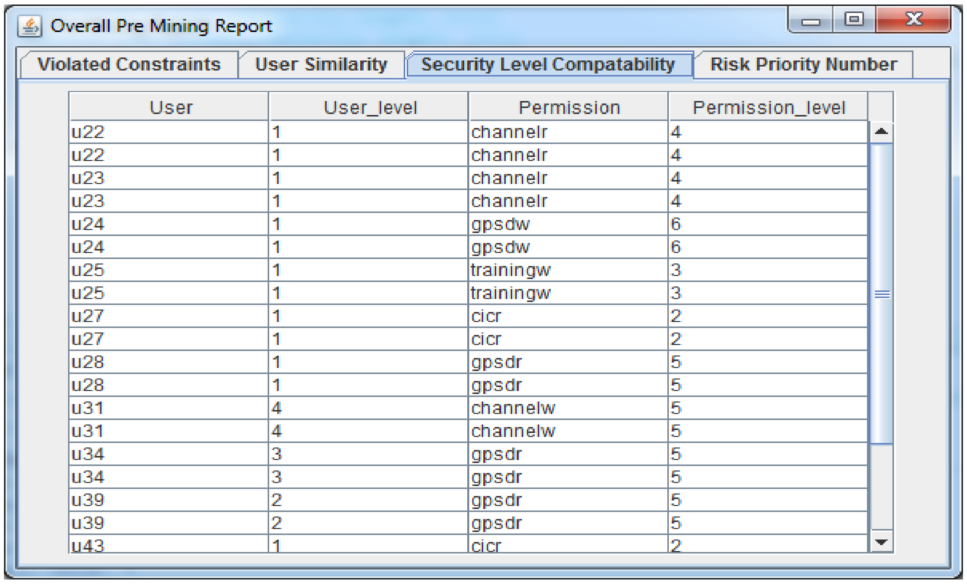


Figure B.13: Pre Mining report generation window

Clicking on the report will open a dashboard with tabs containing report of all the previous steps. You can click on the proceed to mining button to move on to the role mining phase.

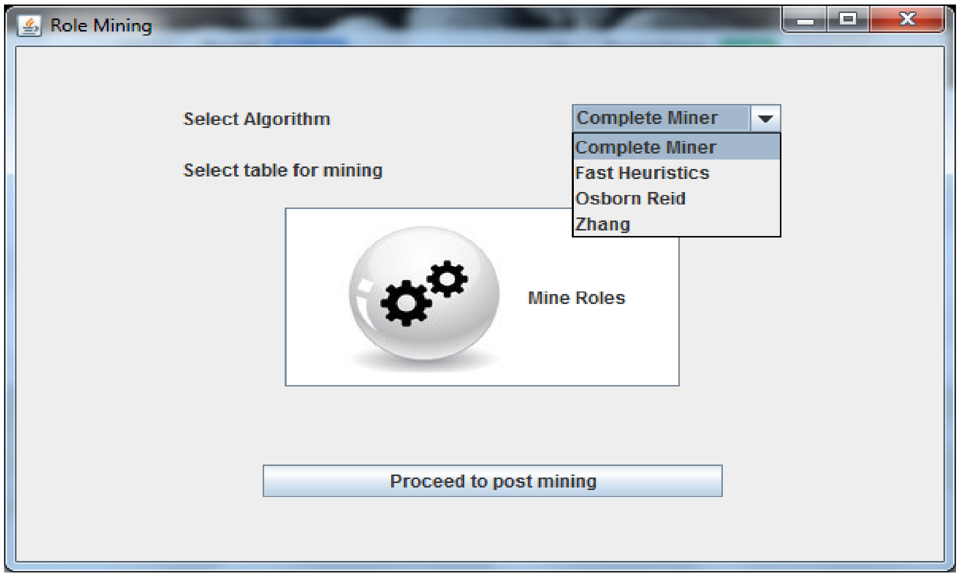

Overall Pre Mining Report

Violated Constraints | User Similarity | Security Level Compatability | Risk Priority Number

User	User_level	Permission	Permission_level
u22	1	channelr	4
u22	1	channelr	4
u23	1	channelr	4
u23	1	channelr	4
u24	1	gpsdw	6
u24	1	gpsdw	6
u25	1	trainingw	3
u25	1	trainingw	3
u27	1	cicr	2
u27	1	cicr	2
u28	1	gpsdr	5
u28	1	gpsdr	5
u31	4	channelw	5
u31	4	channelw	5
u34	3	gpsdr	5
u34	3	gpsdr	5
u39	2	gpsdr	5
u39	2	gpsdr	5
u43	1	cicr	2

Figure B.14: Overall Pre Mining dashboard

In the next window is the role mining phase. You can select from a list of algorithms the one that you want to use and also select the table that contains the user permission information needed for the mining process. Once these are selected, you can select the mine roles button to mine the roles.

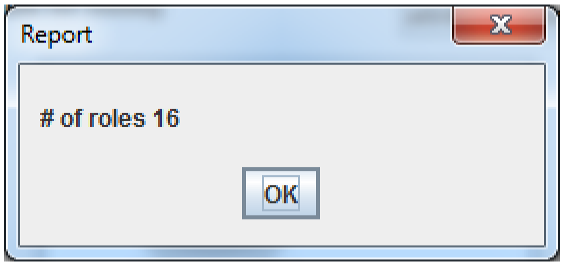

Role Mining

Select Algorithm: Complete Miner

Select table for mining

Figure B.15: Window for role mining

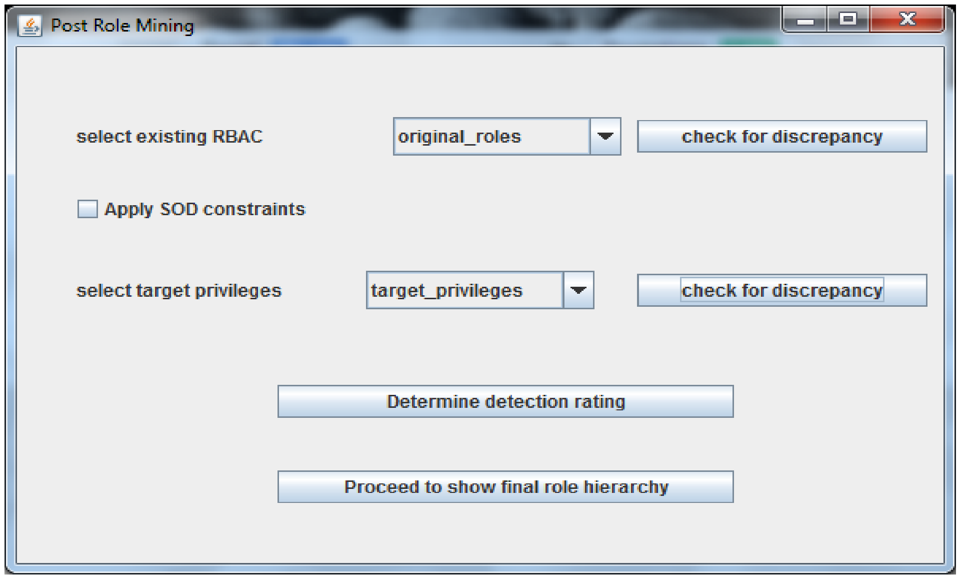
Once the mine roles button is pressed, a small report showing the number of roles generated will pop up.



# of roles 16

Figure B.16: number of roles generated

Once you decide to proceed to the post mining steps by clicking the proceed to post mining button, you will get to the following window as shown in Figure B.17.


Post Role Mining
check for discrepancy
Determine detection rating
Proceed to show final role hierarchy

select existing RBAC: original_roles

Apply SOD constraints

select target privileges: target_privileges

Figure B.17: Post role mining window

There are a number of things that can be done in the post role mining window. For example, if you decide to view whether the target privileges have been met, you may select the table that contains the target privilege information and after selecting the table, click on the button to check for discrepancy.

If the window shown in Figure B.18 is shown, then the target privileges have been met and there is no action that needs to be taken. However, if we find the type of window as shown in

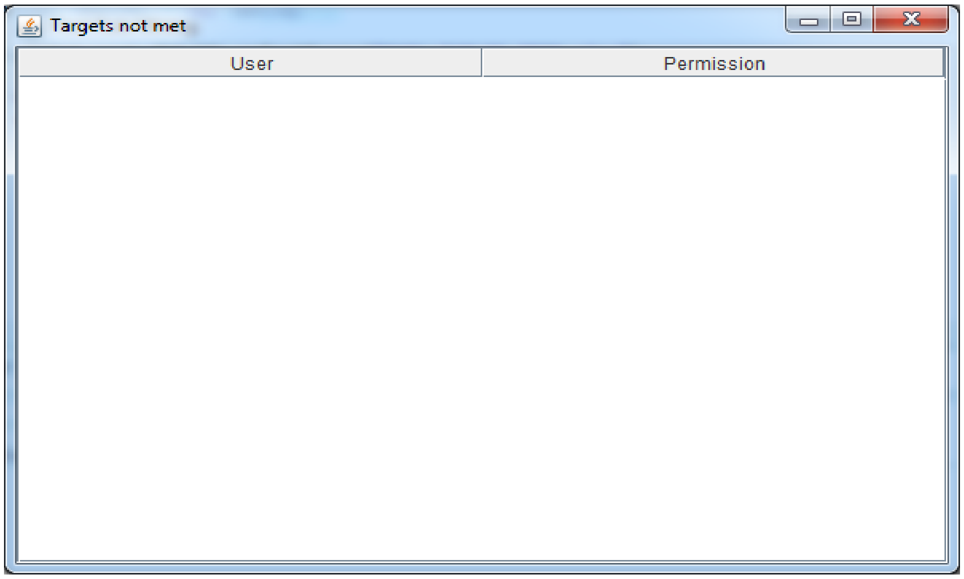
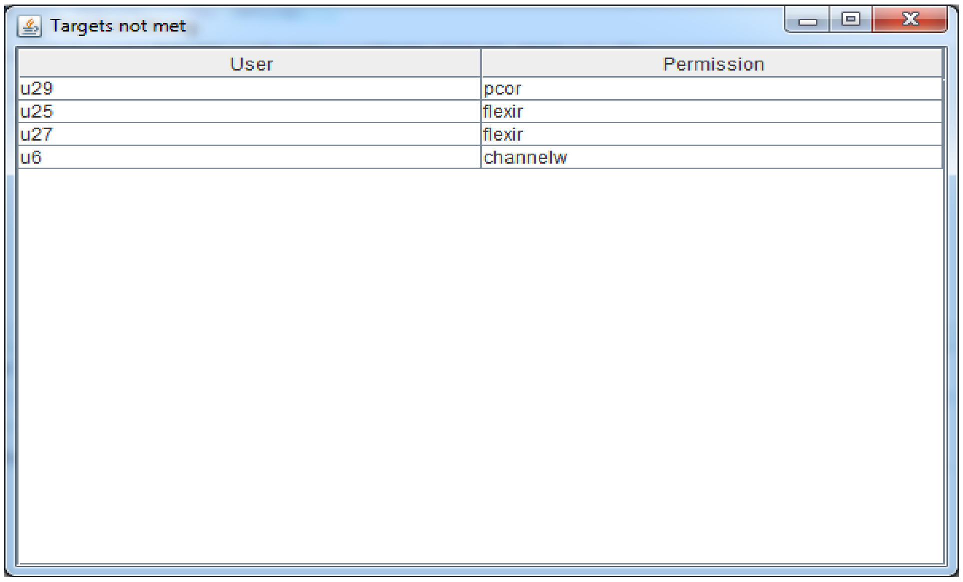


Figure B.18: Targets not met report when all targets are met

Figure B.19 then certain target privileges have not been met.

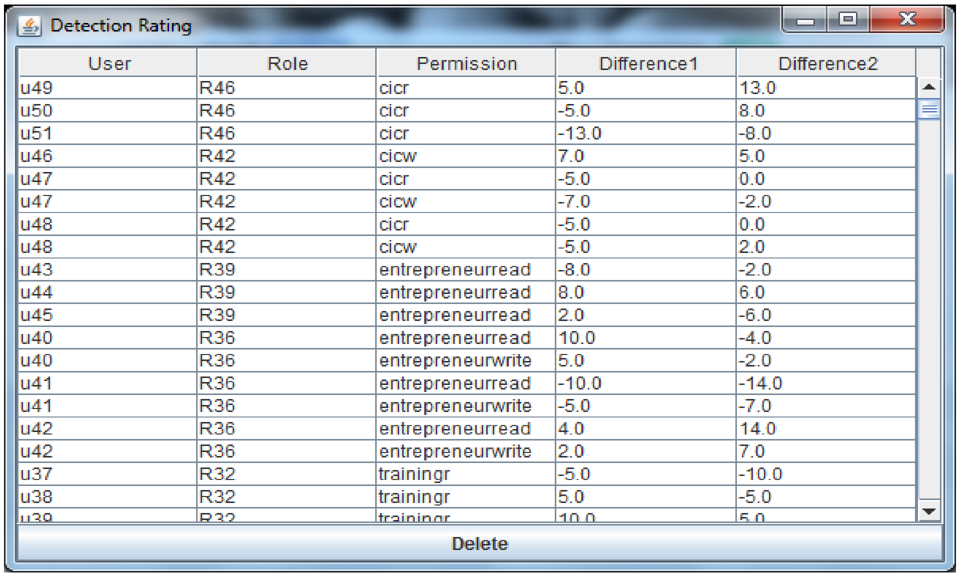


User	Permission
u29	pcor
u25	flexir
u27	flexir
u6	channelw

Figure B.19: Targets not met report with user and corresponding permissions that have not been met

If we click on the 'determine detection rating' button then the report as shown in Figure B.20 will be shown.

Now if we wish to press the button Proceed to show the role hierarchy then the next window as shown in Figure B.21 will appear.



User	Role	Permission	Difference1	Difference2
u49	R46	cicr	5.0	13.0
u50	R46	cicr	-5.0	8.0
u51	R46	cicr	-13.0	-8.0
u46	R42	cicw	7.0	5.0
u47	R42	cicr	-5.0	0.0
u47	R42	cicw	-7.0	-2.0
u48	R42	cicr	-5.0	0.0
u48	R42	cicw	-5.0	2.0
u43	R39	entrepreneurread	-8.0	-2.0
u44	R39	entrepreneurread	8.0	6.0
u45	R39	entrepreneurread	2.0	-6.0
u40	R36	entrepreneurread	10.0	-4.0
u40	R36	entrepreneurwrite	5.0	-2.0
u41	R36	entrepreneurread	-10.0	-14.0
u41	R36	entrepreneurwrite	-5.0	-7.0
u42	R36	entrepreneurread	4.0	14.0
u42	R36	entrepreneurwrite	2.0	7.0
u37	R32	trainingr	-5.0	-10.0
u38	R32	trainingr	5.0	-5.0
u39	R32	trainingr	10.0	5.0

Figure B.20: Detection rating report with difference values

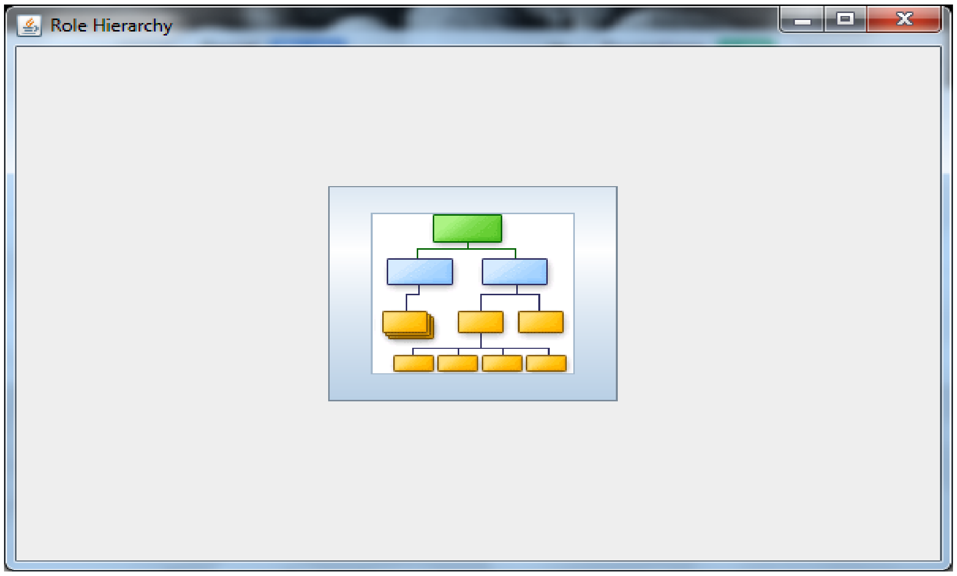


Figure B.21: Role hierarchy report window

To view the final role hierarchy, we click on the hierarchy image icon as circled in red in Figure B.22.

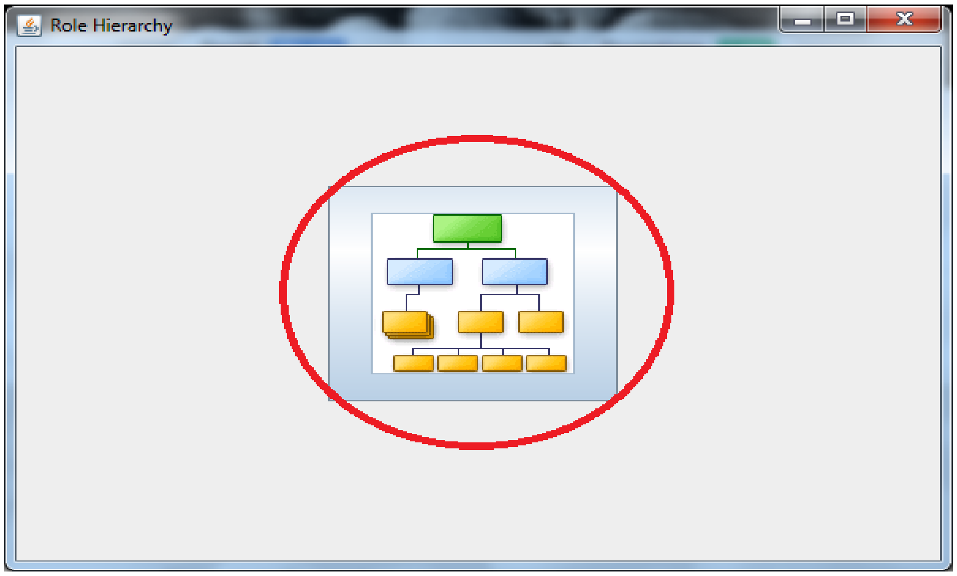


Figure B.22: Button to press to view role hierarchy

In the next window we will be able to see the final role hierarchy, the most useful thing about this hierarchy generation is the fact that the nodes are editable and can be moved around according to necessity for better visibility. The final role hierarchy is shown in Figure B.23.

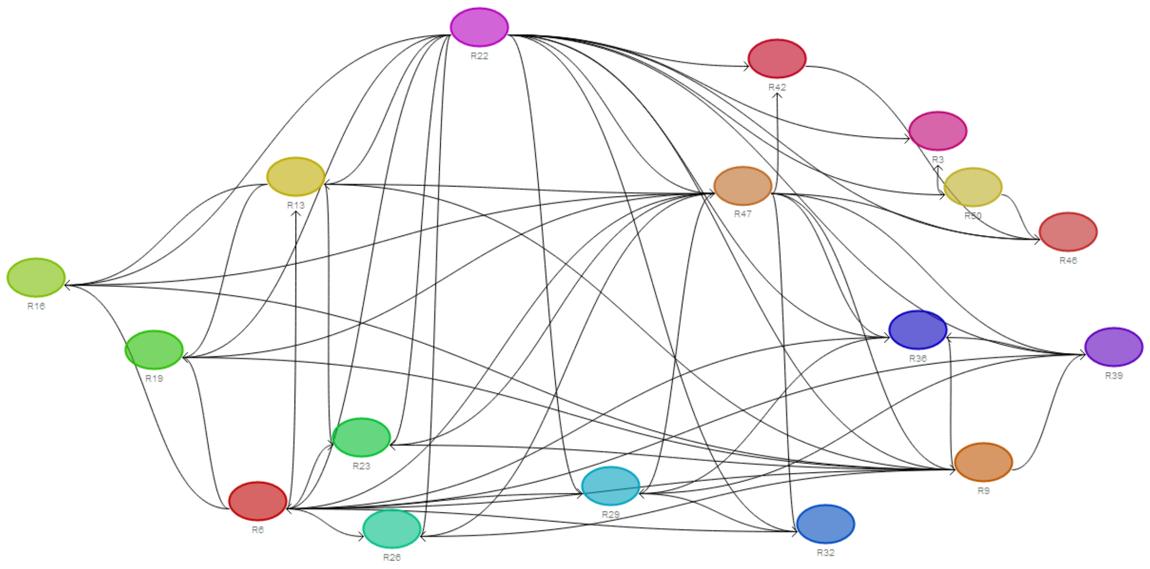


Figure B.23: Final role hierarchy

Appendix C

Correlation Data

For the case 1 study, the dataset was small and thus the correlation results were not all statistically significant. By statistical significance, we mean that the 95% confidence intervals do not include zero (i.e., either the upper and lower bound of CI are positive or both of them are negative [51]. However, the correlation of similarity and security breach values seemed to be statistically significant. To check their statistical significance more accurately, we ran the correlation between the security breach and similarity on all the 51 users for different levels of noise. In this Appendix, we have all the raw data from the correlations.

Table C.1: Raw Data for correlation with 1% General noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	0	0
u5	1	0
u6	0	0
u7	0	0
u8	0	0
u9	0	0
u10	0	0
u11	0	0
u12	1	0
u13	0	0
u14	0	0
u15	0	0
u16	0	0
u17	1	0
u18	0	0
u19	1	0
u20	1	0
u21	1	1
u22	0	0
u23	0	0
u24	0	0
u25	0	0
u26	0	0
u27	0	0
u28	0	0
u29	0	0
u30	0	0
u31	0	0
u32	1	0
u33	0	0
u34	0	0
u35	0	0
u36	0	0
u37	0	0
u38	0	0
u39	0	0
u40	0	0
u41	0	0
u42	0	0
u43	0	0
u44	0	0
u45	0	0
u46	0	0
u47	0	0
u48	0	0
u49	0	0
u50	0	0
u51	0	0

Table C.2: Raw Data for correlation with 1% Subtractive noise

	Similarity	Security Breach
u1	0	0
u2	1	0
u3	0	0
u4	0	0
u5	1	0
u6	0	0
u7	0	0
u8	0	0
u9	0	0
u10	0	0
u11	0	0
u12	1	0
u13	1	0
u14	0	0
u15	0	0
u16	0	0
u17	0	0
u18	0	0
u19	0	0
u20	1	0
u21	0	0
u22	0	0
u23	0	0
u24	0	0
u25	0	0
u26	0	0
u27	0	0
u28	0	0
u29	0	0
u30	0	0
u31	0	0
u32	0	0
u33	0	0
u34	0	0
u35	0	0
u36	0	0
u37	0	0
u38	0	0
u39	0	0
u40	0	0
u41	0	0
u42	0	0
u43	0	0
u44	0	0
u45	0	0
u46	0	0
u47	0	0
u48	0	0
u49	0	0
u50	0	0
u51	0	0

Table C.3: Raw Data for correlation with 1% Additive Noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	0	0
u5	0	0
u6	0	0
u7	0	0
u8	0	0
u9	1	0
u10	0	0
u11	0	0
u12	0	0
u13	0	0
u14	0	0
u15	0	0
u16	0	0
u17	0	0
u18	0	0
u19	0	0
u20	0	0
u21	0	0
u22	0	0
u23	0	0
u24	0	0
u25	1	0
u26	1	1
u27	1	1
u28	0	0
u29	1	1
u30	0	0
u31	0	0
u32	1	0
u33	0	0
u34	0	0
u35	0	0
u36	0	0
u37	0	0
u38	0	0
u39	0	0
u40	1	0
u41	0	0
u42	0	0
u43	1	1
u44	0	0
u45	0	0
u46	0	0
u47	0	0
u48	0	0
u49	0	0
u50	0	0
u51	0	0

Table C.4: Raw Data for correlation with 2% General Noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	0	0
u5	0	0
u6	0	0
u7	1	0
u8	1	0
u9	1	0
u10	1	0
u11	0	0
u12	0	0
u13	0	0
u14	0	0
u15	1	0
u16	1	0
u17	1	0
u18	1	0
u19	1	0
u20	0	0
u21	0	0
u22	1	0
u23	1	0
u24	1	0
u25	0	0
u26	0	0
u27	0	0
u28	0	0
u29	0	0
u30	0	0
u31	0	0
u32	0	0
u33	1	0
u34	1	1
u35	0	0
u36	0	0
u37	0	0
u38	1	0
u39	0	0
u40	0	0
u41	0	0
u42	0	0
u43	0	0
u44	1	1
u45	0	0
u46	0	0
u47	0	0
u48	0	0
u49	0	0
u50	0	0
u51	0	0

Table C.5: Raw Data for correlation with 2% Subtractive Noise

	Similarity	Security Breach
u1	1	0
u2	0	0
u3	0	0
u4	1	0
u5	1	0
u6	1	0
u7	0	0
u8	1	0
u9	0	0
u10	0	0
u11	0	0
u12	0	0
u13	1	0
u14	1	0
u15	1	0
u16	1	0
u17	1	0
u18	1	0
u19	0	0
u20	0	0
u21	0	0
u22	0	0
u23	0	0
u24	0	0
u25	0	0
u26	0	0
u27	0	0
u28	0	0
u29	0	0
u30	0	0
u31	0	0
u32	1	0
u33	0	0
u34	0	0
u35	0	0
u36	0	0
u37	0	0
u38	0	0
u39	0	0
u40	0	0
u41	1	0
u42	1	0
u43	0	0
u44	0	0
u45	0	0
u46	0	0
u47	0	0
u48	0	0
u49	0	0
u50	1	0
u51	0	0

Table C.6: Raw Data for correlation with 2% Additive Noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	0	0
u5	0	0
u6	0	0
u7	0	0
u8	0	0
u9	0	0
u10	0	0
u11	1	0
u12	0	0
u13	0	0
u14	0	0
u15	0	0
u16	0	0
u17	0	0
u18	0	0
u19	0	0
u20	1	1
u21	0	0
u22	1	0
u23	0	0
u24	0	0
u25	0	0
u26	1	1
u27	0	0
u28	0	0
u29	0	0
u30	1	0
u31	0	0
u32	0	0
u33	0	0
u34	0	0
u35	0	0
u36	1	0
u37	0	0
u38	1	0
u39	0	0
u40	0	0
u41	1	1
u42	0	0
u43	1	0
u44	1	1
u45	1	1
u46	1	0
u47	1	0
u48	1	0
u49	0	0
u50	0	0
u51	0	0

Table C.7: Raw Data for correlation with 3% General Noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	0	0
u5	0	0
u6	1	0
u7	1	0
u8	0	0
u9	0	0
u10	0	0
u11	1	0
u12	0	0
u13	0	0
u14	1	0
u15	0	0
u16	0	0
u17	0	0
u18	1	0
u19	0	0
u20	0	0
u21	0	0
u22	0	0
u23	1	1
u24	0	0
u25	1	0
u26	1	1
u27	1	0
u28	0	0
u29	0	0
u30	0	0
u31	0	0
u32	0	0
u33	0	0
u34	0	0
u35	1	0
u36	0	0
u37	0	0
u38	1	0
u39	0	0
u40	0	0
u41	0	0
u42	0	0
u43	0	0
u44	1	1
u45	0	0
u46	0	1
u47	1	0
u48	0	1
u49	0	0
u50	1	1
u51	0	0

Table C.8: Raw Data for correlation with 3% Subtractive Noise

	Similarity	Security Breach
u1	1	0
u2	1	0
u3	1	0
u4	1	0
u5	1	0
u6	1	0
u7	1	0
u8	0	0
u9	0	0
u10	0	0
u11	0	0
u12	0	0
u13	1	0
u14	1	0
u15	1	0
u16	1	0
u17	1	0
u18	1	0
u19	0	0
u20	0	0
u21	0	0
u22	0	0
u23	1	0
u24	0	0
u25	0	0
u26	0	0
u27	1	0
u28	0	0
u29	0	0
u30	0	0
u31	0	0
u32	0	0
u33	0	0
u34	1	0
u35	1	0
u36	1	0
u37	0	0
u38	0	0
u39	0	0
u40	0	0
u41	0	0
u42	0	0
u43	0	0
u44	0	0
u45	0	0
u46	0	0
u47	0	0
u48	0	0
u49	0	0
u50	0	0
u51	0	0

Table C.9: Raw Data for correlation with 3% Additive Noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	0	0
u5	0	0
u6	0	0
u7	1	0
u8	1	0
u9	1	0
u10	1	0
u11	1	0
u12	1	0
u13	1	0
u14	0	0
u15	0	0
u16	1	0
u17	0	0
u18	0	0
u19	0	0
u20	0	0
u21	1	0
u22	1	1
u23	0	0
u24	0	0
u25	0	0
u26	0	0
u27	0	0
u28	1	1
u29	1	1
u30	1	0
u31	1	0
u32	1	0
u33	1	0
u34	1	0
u35	1	1
u36	1	0
u37	1	1
u38	1	0
u39	1	1
u40	1	0
u41	1	0
u42	1	1
u43	0	0
u44	0	0
u45	0	0
u46	0	0
u47	1	1
u48	0	0
u49	1	0
u50	0	0
u51	0	0

Table C.10: Raw Data for correlation with 4% General Noise

	Similarity	Security Breach
u1	0	0
u2	1	0
u3	0	0
u4	0	0
u5	0	0
u6	0	0
u7	1	0
u8	0	0
u9	0	0
u10	0	0
u11	0	0
u12	1	0
u13	1	0
u14	1	0
u15	1	0
u16	0	0
u17	1	1
u18	0	0
u19	1	0
u20	1	0
u21	1	0
u22	1	0
u23	0	1
u24	1	1
u25	0	0
u26	1	1
u27	0	0
u28	0	0
u29	0	0
u30	1	1
u31	1	0
u32	0	0
u33	0	0
u34	1	0
u35	0	0
u36	0	0
u37	0	0
u38	0	0
u39	0	0
u40	1	1
u41	1	0
u42	1	1
u43	0	0
u44	1	1
u45	0	0
u46	1	0
u47	1	0
u48	1	0
u49	1	0
u50	0	0
u51	1	1

Table C.11: Raw Data for correlation with 4% Subtractive Noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	0	0
u5	0	0
u6	0	0
u7	0	0
u8	0	0
u9	0	0
u10	0	0
u11	0	0
u12	0	0
u13	0	0
u14	0	0
u15	0	0
u16	0	0
u17	0	0
u18	0	0
u19	0	0
u20	0	0
u21	0	0
u22	0	0
u23	0	0
u24	0	0
u25	0	0
u26	0	0
u27	0	0
u28	0	0
u29	0	0
u30	0	0
u31	0	0
u32	0	0
u33	0	0
u34	0	0
u35	0	0
u36	0	0
u37	0	0
u38	0	0
u39	0	0
u40	0	0
u41	0	0
u42	0	0
u43	0	0
u44	0	0
u45	0	0
u46	0	0
u47	0	0
u48	0	0
u49	0	0
u50	0	0
u51	0	0

Table C.12: Raw Data for correlation with 4% Additive Noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	1	0
u5	0	0
u6	0	0
u7	0	0
u8	1	0
u9	0	0
u10	1	0
u11	1	0
u12	1	0
u13	0	0
u14	0	0
u15	0	0
u16	0	0
u17	0	0
u18	0	0
u19	0	0
u20	1	1
u21	0	0
u22	0	0
u23	0	0
u24	1	1
u25	0	0
u26	1	1
u27	0	0
u28	1	0
u29	1	1
u30	1	1
u31	1	0
u32	1	0
u33	1	0
u34	0	0
u35	0	0
u36	0	0
u37	1	0
u38	0	0
u39	0	0
u40	0	1
u41	1	1
u42	1	1
u43	0	0
u44	0	0
u45	1	1
u46	1	0
u47	1	0
u48	1	0
u49	1	0
u50	1	0
u51	1	0

Table C.13: Raw Data for correlation with 5% General Noise

	Similarity	Security Breach
u1	1	0
u2	1	0
u3	1	0
u4	1	0
u5	1	0
u6	1	0
u7	1	0
u8	1	0
u9	1	0
u10	1	1
u11	1	0
u12	1	0
u13	1	0
u14	0	0
u15	0	0
u16	0	0
u17	0	0
u18	0	0
u19	1	1
u20	0	0
u21	0	0
u22	0	0
u23	1	1
u24	0	0
u25	1	1
u26	1	0
u27	1	1
u28	1	0
u29	1	1
u30	1	1
u31	1	0
u32	1	0
u33	1	0
u34	0	0
u35	1	0
u36	0	0
u37	0	0
u38	1	1
u39	0	0
u40	1	0
u41	1	0
u42	1	1
u43	1	1
u44	1	1
u45	1	1
u46	1	0
u47	1	0
u48	1	0
u49	1	1
u50	1	1
u51	1	1

Table C.14: Raw Data for correlation with 5% Subtractive Noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	0	0
u5	0	0
u6	0	0
u7	0	0
u8	0	0
u9	0	0
u10	0	0
u11	0	0
u12	0	0
u13	0	0
u14	0	0
u15	0	0
u16	0	0
u17	0	0
u18	0	0
u19	0	0
u20	0	0
u21	0	0
u22	0	0
u23	0	0
u24	0	0
u25	0	0
u26	0	0
u27	0	0
u28	0	0
u29	0	0
u30	0	0
u31	0	0
u32	0	0
u33	0	0
u34	0	0
u35	0	0
u36	0	0
u37	0	0
u38	0	0
u39	0	0
u40	0	0
u41	0	0
u42	0	0
u43	0	0
u44	0	0
u45	0	0
u46	0	0
u47	0	0
u48	0	0
u49	0	0
u50	0	0
u51	0	0

Table C.15: Raw Data for correlation with 5% Additive Noise

	Similarity	Security Breach
u1	0	0
u2	0	0
u3	0	0
u4	0	0
u5	0	0
u6	0	0
u7	1	0
u8	1	0
u9	1	0
u10	1	0
u11	1	0
u12	1	0
u13	0	0
u14	0	0
u15	1	0
u16	0	0
u17	0	0
u18	0	0
u19	0	0
u20	0	0
u21	0	0
u22	1	1
u23	1	0
u24	1	1
u25	1	0
u26	1	1
u27	1	1
u28	1	1
u29	1	1
u30	1	0
u31	1	0
u32	1	0
u33	1	1
u34	0	0
u35	1	0
u36	0	0
u37	1	1
u38	1	0
u39	1	0
u40	1	1
u41	1	0
u42	1	1
u43	1	1
u44	1	1
u45	1	0
u46	1	0
u47	0	0
u48	0	0
u49	1	0
u50	1	1
u51	1	1

Curriculum Vitae

Name:	SharminAhmed
Post-Secondary Education and Degrees:	University of Western Ontario London, Ontario 2008 - 2009 M.Sc
	University of Western Ontario London, ON 2010 - 2014 Ph.D.
Honours and Awards:	Western Graduate Research Scholarship 2008-2009, 2010-2014
Related Work Experience:	Teaching Assistant The University of Western Ontario 2008 - 2012

Publications:

Sharmin Ahmed and Sylvia L. Osborn. (2014). Application of Risk Metrics For Role Mining. (submitted for publication).

Sharmin Ahmed and Sylvia L. Osborn. (2014). A system for risk awareness during role mining. In Proceedings of the 19th ACM symposium on Access control models and technologies (SACMAT '14). ACM, New York, NY, USA, 181-184.

Sharmin Ahmed, M. I. Chowdhury, Remo Ferrari and N.H. Madhavji. (2011). Efficient Requirements Engineering through Feed-forward. International Requirements Engineering Effi-

ciency workshop (REEW11) at 17th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ11). Essen, Germany.