TACKLING INSIDER THREATS USING RISK-AND-TRUST AWARE ACCESS CONTROL APPROACHES

by

Nathalie Baracaldo

Master in Computer Sciences, Universidad de los Andes, 2008

Bachelors in Computer Sciences, Universidad de los Andes, 2006

Bachelors in Industrial Engineering, Universidad de los Andes, 2006

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2016

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Nathalie Baracaldo

It was defended on

January 7, 2016

and approved by

TACKLING INSIDER THREATS USING RISK-AND-TRUST AWARE ACCESS CONTROL APPROACHES

Nathalie Baracaldo, PhD

University of Pittsburgh, 2016

Insider Attacks are one of the most dangerous threats organizations face today. An insider attack occurs when a person authorized to perform certain actions in an organization decides to abuse the trust, and harm the organization by causing breaches in the confidentiality, integrity or availability of the organization's assets. These attacks may negatively impact the reputation of the organization, its productivity, and may incur heavy losses in revenue and clients. Preventing insider attacks is a daunting task. Employees need legitimate access to effectively perform their jobs; however, at any point of time they may misuse their privileges accidentally or intentionally. Hence, it is necessary to develop a system capable of finding a middle ground where the necessary privileges are provided and insider threats are mitigated. In this dissertation, we address this critical issue.

We propose three *adaptive risk-and-trust aware access control* frameworks that aim at thwarting insider attacks by incorporating the behavior of users in the access control decision process. Our first framework is tailored towards general insider threat prevention in role-based access control systems. As part of this framework, we propose methodologies to specify risk-and-trust aware access control policies and a risk management approach that minimizes the risk exposure for each access request. Our second framework is designed to mitigate the risk of obligation-based systems which are difficult to manage and are particularly vulnerable to sabotage. As part of our obligation-based framework, we propose an insider-threat-resistant trust computation methodology. We emphasize the use of monitoring of obligation fulfillment patterns to determine some psychological precursors that have high predictive power with respect to potential insider threats. Our third framework is designed to take advantage of geo-social information to deter insider threats. We uncover some insider threats that arise when geo-social information is used to make access control decisions. Based on this analysis, we define an insider threat resilient access control approach to manage privileges that considers geo-social context. The models and methodologies presented in this dissertation can help a broad range of organizations in mitigating insider threats.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

These past few years have been an amazing journey that has finally concluded with the completion of this dissertation. I would like to thank those who helped me through this journey and have made possible this milestone.

First, I would like to thank my adviser for his guidance, feedback and support throughout my Ph.D. studies. I also would like to thank my committee members, Dr. Prashant Krishnamurthy, Dr. Balaji Palanisamy and Dr. Heiko Ludwig, for their valuable insights and support of my research. I am also thankful for the career advise they have provided me.

I would also like to thank Dr. Marek Druzdzel for sharing his knowledge and being available to discuss about utility theory, Bayesian networks and other probabilistic models. Our discussions helped shape the utility decision model that is part of the geo-social framework presented in this dissertation. I would like to thank all faculty members, especially Dr. Paul Munro, who pointed me in the right direction when I was looking for a graduate program. I would like to thank my LERSAIS lab mates and school friends for making my Ph.D. experience more productive and enjoyable. I would like to thank Amirreza Masoumzadeh, Xuelian Long, Jesus Gonzales, Yue Zhang, Lei Jin, Hassan Takabi, Saman Taghavi-Zargar, Leila Karimi, Runhua Xu, Chao Li, Xu Jinlai and Marcela Gomez. I also would like to acknowledge all the administrative stuff, especially Mary Stewart and Kelly Shaffer, who went out of their ways to help me out. I would also like to thank Milton Quiroga, my master's adviser, for getting me interested in the security field and Jody Glider for sharing his wisdom and teaching me the importance of pragmatism and real world applicability.

I am especially thankful to my husband, Santiago Bock. Thanks for surrounding me with love, for being my number one supporter and for always making everything feel right. Without you, this journey would not have been as much fun! You are my sunshine! Last but not least, I would like to thank my parents and family. I would not be here without their support. I would like to thank them for their unconditional love, for always believing in me, for teaching me to dream big, work hard and pursue my dreams with passion and confidence. *Siempre están en mi corazón, gracias!*

This research has been supported by the US National Science Foundation award IIS-0545912.

1.0 INTRODUCTION

"The year 2013 may be the year of the insider threat. Recent incidents of intellectual property theft, exfiltration of sensitive intelligence, and international espionage concerns have risen to the legal and regulatory forefront, quickly becoming a matter of political debate and public speculation. These incidents highlight the need to improve the ability of organizations to detect, deter, and respond to insider threats".
Computer Emergency Response Team (CERT), January 2014 [52].

An insider attack is carried out by people who are legitimately authorized in the system to perform certain tasks. Without a doubt, the data exfiltration performed by Edward Snowden has been one of the most publicized insider incidents in recent history [64]. Snowden, while working as a governmental contractor, leaked an estimate of 200,000 classified documents from the US National Security Agency (NSA). This incident illustrates the significant damage that can be inflicted by insiders and the urgent need for new solutions to mitigate this type of threat.

Snowden's insider attack is not an isolated incident. Insider threats have occurred across all public and private sectors. According to the US State of Cybercrime Survey, insider attacks accounted for 28% of the total incidents reported in 2014 [29]. Additionally, 32% of the respondents reported that insider attacks were more damaging than attacks performed by outsiders and 31% of the respondents in that survey reported incidents that could not be attributed with certitude to insiders or outsiders. This indicates a lack of accountability and possibly more incidents caused by insiders. The consequences of insider attacks may be devastating, and may include financial losses, negative impact on the reputation, loss of customers, among others. According to the CERT [81], the monetary losses due to insider attacks ranged from five hundred dollars to tens of million of dollars, around 75% of the organizations had an adverse impact on their business operations, and 28% experienced a negative impact on their reputation. Moreover, according to the same survey, 60 % of the respondents reported monetary losses caused by non-malicious insiders. These statistics show that it is not wise to trust insiders blindly. For this reason, dealing with insider threats has become one of the most important issues in information security.

Deterring insider attacks and unintentional damage is a daunting task. In contrast to external adversaries, insiders already have some access to the system and have preliminary knowledge about existing defenses and about what data is valuable. Hence, these threats cannot be adequately mitigated using defenses against outsiders. Additionally, while it is necessary to provide privileges to employees so they can perform their jobs efficiently, providing too many privileges may backfire when users accidentally or intentionally abuse their privileges. Hence, finding a middle ground, where the necessary privileges are provided and malicious usages are avoided, is necessary.

Some of the insider attacks could be prevented if users are monitored to identify suspicious activities [81]. In particular, insider attack incidents could be prevented if the system had a monitoring module to evaluate how trusted a user is with respect to *technical* and *psychological* precursors used to predict insider attacks. Some of the *technical precursors* include download and use of hacker tools, failure to create backups, unauthorized access to customers' or coworkers' systems, system access after termination, inappropriate Internet access at work, and the setup or use of backdoor accounts [81]. Among the *psychological precursors*, insider attackers have shown the following symptoms: disgruntlement, bad attitude towards feedback, lack of dependability and absenteeism [58].

Despite the evidence of the predictive value of technical and psychological precursors, they are often overlooked [81], only manually examined [27, 61] or analyzed only for forensic purposes after the damage has already been done [36, 102, 88, 65, 109]. Some of the existing solutions to deter insider attacks aim at specifying and enforcing *least privilege* by uniquely providing the minimum set of required permissions to complete a task at a particular point of time, and *separation of duty* to avoid conflicts of interests that may allow fraudulent activities or personal gain [56]. Although it is crucial to enforce these security principles, they are not enough by themselves. In fact, access control systems such as role-based access control (RBAC) [50], Bell LaPadula [19], obligation-based systems [22], among others, typically incorporate these principles, yet do not adapt to negative changes in users' behavior. In these systems, as long as users can prove they have the necessary set of credentials, the access is granted – for instance, if a user can prove he works as an engineer, he can get all associated privileges, even if his behavior suggests he is attacking the system! These access control systems are appropriate in environments where users are well-behaved and can be trusted to perform actions according to their credentials. Unfortunately, as the statistics show, insiders do perform attacks. Furthermore, even if users could be trusted, malware can be inadvertently installed and a user account compromised. Thus, it is necessary to include the behavior of the users in the access control loop.

This suggests that having a more adaptive enforcement system that considers the behavior of users to make access decisions would help prevent insider attacks. In such a dynamic system, users' behavior should dictate how trusted they are. When a user's behavior falls out of the expected pattern in a suspicious manner, the trust the system has on him should be reduced. If a user is no longer trusted, the system should adapt by denying access to key resources. A different trust value needs to be enforced for each resource depending on the resource's inherent criticality and the potential risk of the resource being misused.

1.1 LIMITATIONS OF EXISTING APPROACHES AND CHALLENGES

Several researchers have recognized the advantages of having more dynamic access control models, e.g., [30, 49, 46, 84]. We refer to these as *adaptive access control systems*. Adding trust helps the system adapt to changes in the behavior of users. A trust threshold is typically used to limit access to resources based on their importance to the organization. However, existing approaches do not provide a comprehensive solution to address insider threats. We identify the following shortcomings in existing approaches that we propose to address as part of this research effort:

- It is not well understood how to adequately model the risk exposure during the access control decision-making process. Often, risk management techniques such as Octave [7], or the NIST risk management methodology [108] are used to determine which threats need to be mitigated and which need to be accepted. As a result of risk management analysis, the technical controls and procedures that need to be in place are identified. However, it is not well understood how to integrate risk analysis results into an access control framework, nor how to aggregate the risk exposure that occurs when an access to a set of resources is granted. Current approaches do not provide a comprehensive analysis to determine how risky an access is and simply assume this information is available in the form of a trust threshold. There is a need to provide clear methodologies to model the risk exposure of an access request.

- Current adaptive access control systems lack procedures to enforce *automatically* the access control policy while minimizing the risk exposure associated with each access request, limiting the power of the system to mitigate insider attacks. Salim *et. al* [99] propose a system that relies on users to minimize the risk exposure by themselves. However, in the context of insider threats, we believe that it is not wise to trust users in this respect. Furthermore, humans are known to have bounded rationality [38]; therefore, their solutions are most likely suboptimal. To close this gap, it is necessary to define a system to minimize *automatically* the risk exposure every time an access request is received.

- Access control policies are prone to errors and misconfigurations that result in a dangerous false sense of security. Research in this area has produced methodologies to analyze policies and discover, detect and resolve policy misconfigurations, e.g., [107, 69, 54, 16, 44, 17, 104]. However, these works have focused on analyzing policies of non-adaptive access control systems. For this reason, their methodologies are not fully applicable to adaptive systems. In particular, they do not cover some of the caveats that arise when policy constraints for insider threat mitigation are in place. More research in this area is needed.

- The recent proliferation of mobile devices and social media has created newer opportunities to design an adaptive access control approach that is better suited for the mitigation of insider threats. Despite the availability of geo-social information and techniques to analyze it (e.g., [35, 78, 5, 40]), there have been few research efforts that focus on leveraging geo-social information to mitigate insider threats. Nonetheless, valuable information such as interaction or relationship between users and places they frequently visit has not been considered by existing adaptive access control systems. We believe that geo-social information can help capture the risk exposure added by users in the vicinity and can also assist in analyzing patterns of misbehavior. These risk factors should influence the access control decision process to prevent insider threats.

A few recent approaches incorporate geo-social context of users as part of the access control policy [76, 63, 14]. However, they are not designed to take into consideration suspicious user behavior and thus they fail to prevent some insider attacks. We need to have a better understanding of how to use geo-social information in access control policies and how including this information impacts the security of the system. Thus, new approaches to fully leverage geo-social information for insider threat mitigation are in need.

- Despite the great predictive value of psychological indicators for insider threat detection [58], they are often overlooked because they require human input and hence are difficult to acquire and maintain up to date. Furthermore, collecting such information may violate employee's privacy [58, 111, 73]. Hence, new ways to collect this type of indicators are needed.

With these limitations in mind, in the following, we present the objectives of our research.

1.2 OVERVIEW OF THE PROPOSED RESEARCH

The *hypothesis* of this study is that some insider attacks and undesirable incidents can be avoided by designing an access control system that is able to adapt to negative changes on users' behavior as well as contextual information. In particular, we hypothesize that enhancing access control systems by including factors such as the inference of critical unauthorized information, obligation management, geo-social contextual information, collusion indicators, among other technical and psychological precursors can help evaluate the risk exposure of each access request and mitigate it accordingly. We believe that including risk management techniques that incorporate the previously mentioned enhancement components into the ac-

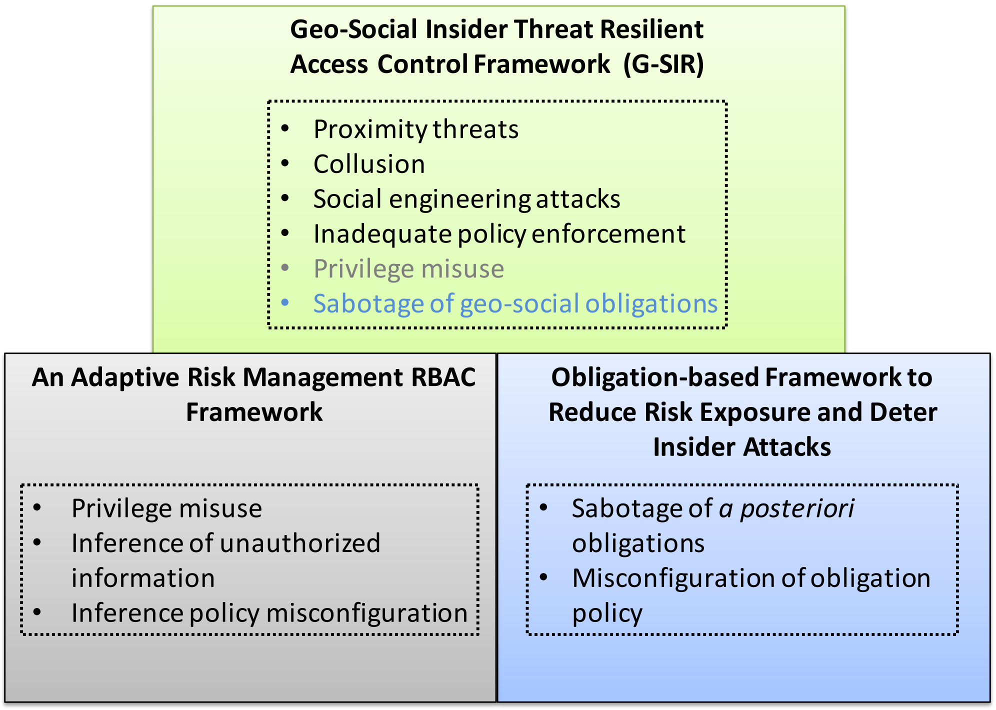

Geo-Social Insider Threat Resilient Access Control Framework (G-SIR)
• Proximity threats
• Collusion
• Social engineering attacks
• Inadequate policy enforcement
• Privilege misuse
• Sabotage of geo-social obligations
An Adaptive Risk Management RBAC Framework
• Privilege misuse
• Inference of unauthorized information
• Inference policy misconfiguration
Obligation-based Framework to Reduce Risk Exposure and Deter Insider Attacks
• Sabotage of *a posteriori* obligations
• Misconfiguration of obligation policy

Figure 1: Conceptual view of the proposed frameworks.

cess control decision process is a natural way to model the problem as different assets need different degrees of protection. We also argue that access control systems are an appropriate place to perform risk management because they are the main component controlling who can access different information, assets or resources of the system.

Towards proving our hypothesis, we develop the three frameworks depicted in Figure 1. Each of the proposed frameworks is designed to mitigate a different set of insider threats. A detailed description of these threats is presented in Chapter 3. We now briefly describe the objectives and research questions of each of the proposed frameworks.

1. *An Adaptive Risk Management RBAC Framework*

   This framework is designed for general insider threat mitigation in systems that use RBAC. We focus on extending the RBAC model because it is widely adopted and has been proven to be a promising approach for different types of organizations, as documented in [95, 85]. Our objective is to develop an adaptive risk management framework for RBAC systems to proactively incorporate the most recent behavior of users as a factor for making an authorization decision. This risk-and-trust aware framework should provide a comprehensive risk aggregation methodology and manage risk automatically for every access request.

The key *research questions* that we address while designing this framework are the following: How should an adaptive RBAC-based system be modeled to mitigate insider attacks? How should we model and compute the risk of granting an access? Can we define an optimization problem to minimize automatically the risk exposure while enforcing the access control policy? How can we help system administrators mitigate the risk exposure caused by inference of unauthorized information?

2. *An Obligation-based Framework to Reduce Risk Exposure and Deter Insider Attacks*

   The second proposed framework aims to mitigate insider threats in obligation-based access control systems. Obligation-based access control systems have emerged as an important approach to perform privilege management in multiple application domains [91, 66], for example, in health care information systems [86], digital right management [75] and privacy aware systems [18]. These systems are particularly risky because they incorporate *a posteriori* obligations. Such an obligation is an action imposed on users after an access is granted. Because the access is already granted, there is no guarantee that users will fulfill *a posteriori* obligations assigned to them, opening the door to costly threats of sabotage[1].

   Our objective is to reduce the risk exposure introduced by *a posteriori* obligations. Here, we aim to answer the following *research questions*: Can we reduce the risk exposure caused by *a posteriori* obligations during access control? How should we model a risk-and-trust aware obligation-based system to adequately reduce this risk? Can we use *a posteriori* obligations as a way to determine how trusted a user is and what would be an appropriate methodology to do so? Are *a posteriori* obligations a good way to determine whether a user is about to become an insider attacker? How can we use the logs of the system to identify policy misconfigurations and misbehaving users?

[1]For example, violations of the Health Insurance Portability and Accountability Act, which requires the fulfillment of multiple obligations from part of the personnel of health care entities, may carry a maximum penalty of $50,000 per violation, with an annual maximum of $1.5 million [2]

This framework can be used independently with non-RBAC or RBAC based systems. It can also be integrated with the *Adaptive Risk Management RBAC Framework* proposed in this dissertation to achieve a more comprehensive insider threat prevention solution.

3. *A Geo-Social Insider Threat Resilient Access Control Framework (G-SIR)*

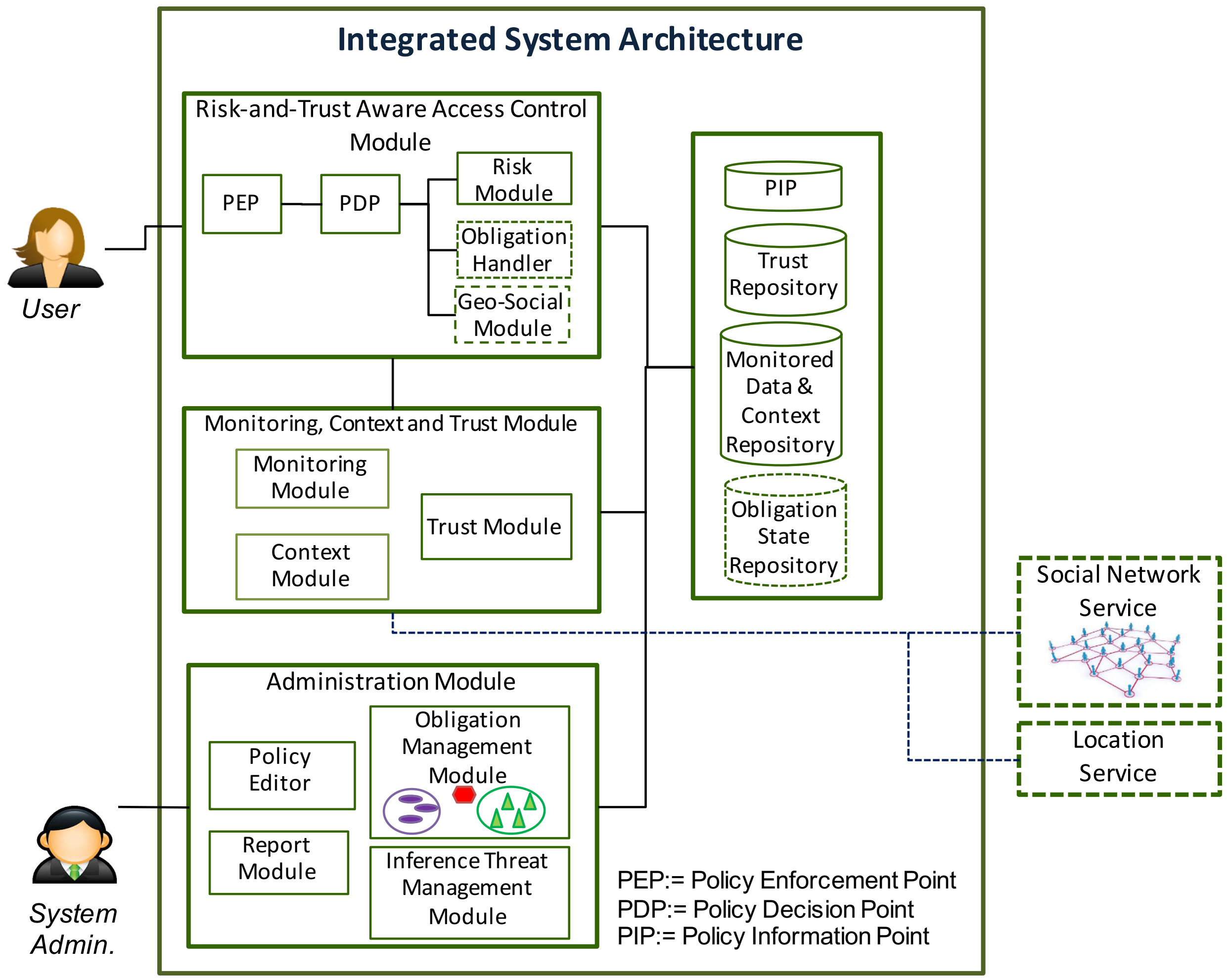
   We propose G-SIR to take advantage of the increasing availability of geo-social information to deter insider threats. This framework is tailored towards organizations that want to adapt to users' geo-social contexts during the access control process. Examples of organizations where geo-social controls may be useful to prevent insider threats are hospitals, research laboratories, critical infrastructure, cloud providers and any other organizations where users may not wander to certain places and where social and location information can provide indications of potential insider threats.

   The *research questions* that we address to develop this framework are the following: How can we use geo-social information to regulate access to critical privileges? What policy constraints are useful when geo-social information is available? Does using geo-social information to regulate accesses introduce new insider threats? Can we manage the risk of colludig communities and proximity threats using geo-social information? Can we use geo-social behavior to determine how trusted users are? If so, what mitigation techniques are appropriate?

The first two proposed frameworks, the adaptive RBAC framework and the obligation-based framework, provide orthogonal solutions that can be integrated or used separately. The first framework assumes the existence of an RBAC system with features that may include separation of duty, cardinality constraints and hybrid hierarchy. Our focus in designing this framework is to provide an approach that allows the automatic reduction of risk exposure for each access request received. Our obligation framework is primarily focused on mitigating the risk introduced by the presence of *a posteriori* obligations. The proposed approach provides a way to identify suspicious users based on their patterns of violation and fulfillment of obligations. Another differentiating factor between these frameworks is that the obligation-based framework does not require the use of RBAC. Hence, any access control model that includes *a posteriori* obligations can adopt the methodologies proposed as part of the obligation-based framework. The two frameworks can be integrated whenever an organization needs to perform insider threat mitigation by using both the RBAC mechanism and obligations.

Our third framework, G-SIR, differs from the previous two in that it focuses on the new opportunities and challenges that using geo-social information in the access control system brings to the picture. We define and design several geo-social policy constraints that may be useful to a variety of organizations with the capability of collecting geo-social information. Neither of the first two frameworks incorporate geo-social information to make access control decisions. In Figure 1, the proposed G-SIR is depicted on top of the adaptive and obligation-based framework because it makes use of some of their methodologies and constructs. In particular, G-SIR makes partial use of the enforcement approaches designed for the adaptive RBAC framework to minimize the risk of each access request. Thus, achieving a full integration between these two frameworks is relatively simple. One of the six types of policy constraints that we define as part of G-SIR are geo-social obligations. This kind of policy constraint is different from *a posteriori* obligations; however, they can be managed through the same trust methodology presented in the obligation-based framework. Moreover, the trust methodology proposed as part of the obligation-based framework can be used to monitor the behavior of insiders concerning the compliance of the G-SIR policy. The obligation-based framework and G-SIR can be integrated to control the risk of each access request considering relevant geo-social aspects as well as the assignment of possible *a posteriori* obligations. Next, we present the system architecture that integrates the three proposed frameworks.

Integrated System Architecture: Figure 2 presents the system architecture that integrates all three proposed frameworks. It consists of a *Risk-and-Trust Aware Access Control Module*, a *Monitoring, Context and Trust Module*, and an *Administration Module*. The *Risk-and-Trust Aware Access Control Module* is responsible for making access control decisions that manage the risk exposure. To perform this task effectively, this module considers the current context of users in the system as well as how trusted they are. This information comes from the *Monitoring, Context and Trust Module*. The latter consists of a *Monitoring Module* that collects data about the user's activities in the system, which are later analyzed



Integrated System Architecture
Risk-and-Trust Aware Access Control Module
PEP
PDP
Risk Module
Obligation Handler
Geo-Social Module
User
PIP
Trust Repository
Monitored Data & Context Repository
Obligation State Repository
Monitoring, Context and Trust Module
Monitoring Module
Context Module
Trust Module
Social Network Service
Location Service
Administration Module
Policy Editor
Report Module
Obligation Management Module
Inference Threat Management Module
System Admin.
PEP:= Policy Enforcement Point
PDP:= Policy Decision Point
PIP:= Policy Information Point

Figure 2: Overview of the integrated frameworks.

and correlated by the *Context Module* to identify the context of a user at a particular time. In this dissertation, we assume that these two modules are in place and work adequately. The *Trust Module* uses the monitored information to compute a value that reflects how trusted a user is given his behavior. All data produced by the *Monitoring, Context and Trust Module* is stored in the *trust* and *monitored data and context* repositories, respectively.

The *Risk-and-Trust Aware Access Control Module* works as follows. The policy enforcement point (PEP) intercepts all access requests, all requests are evaluated at the policy decision point (PDP) which grants or denies accesses according to the policy stored in the policy information point (PIP). For this purpose, all the information related to how trusted a user is, the current context and the risk exposure is used to make an authorization decision. The latter value is computed by *Risk Module*, which determines how risky an access request is. With all the relevant information, the access control system determines whether an authorization request should be granted. The decision is made to ensure that the risk exposure is under control. Thus, the Risk-and-Trust-aware module is in charge of minimizing the risk exposure. The last component of the architecture is the *Administration Module*. This component is designed to specify access policies as well as help administrators find policy problems and misconfigurations.

1.3 SCOPE OF THE DISSERTATION

In this dissertation, we make the following assumptions. We assume that all accesses are mediated by a reference monitor, the PEP in Figure 2. Attacks that bypass the PEP or that manage to subvert it, for example, by taking advantage of software vulnerabilities, are out of the scope of this dissertation. We focus on the mitigation of threats where the behavior of insiders change before committing an attack. With respect to the collection of geo-social information, we assume that it is possible to obtain accurate geo-social information from users when they are in their working spaces.

1.4 CONTRIBUTIONS

As part of this dissertation, we propose three frameworks. Their *detailed contributions* are presented in Chapter 3. In the following, we overview the *key contributions* of our work:

- We develop a risk-and-trust RBAC framework that utilizes a novel risk aggregation methodology, which includes the risk exposure caused by inference of unauthorized information. We also define an optimization problem to automatically and optimally minimize the risk exposure of an organization every time the system receives an access request, and propose an algorithm to solve it. Additionally, we provide a methodology to identify user-to-role assignments in the policy that lead to undesirable inference risks.
- We propose an obligation-based access control model that manages risk exposure caused by unfulfilled obligations. To the best of our knowledge, such an obligation-based approach is the first of its kind. We propose to use obligations as a way to determine the mood of employees –a psychological indicator– without introducing subjective information or violating users' privacy. We further develop a trust methodology to determine when it is too risky to allow an access that has obligations associated with it. We also provide a methodology that helps identify obligation-based policy misconfiguration and misbehaving users.
- Little work exists in geo-social access control and existing ones do not consider the intricacies of incorporating geo-social information as part of the access control system for insider threat prevention. We uncover threats that are enabled by existing geo-social access control systems. Then, we propose insider threat mitigation techniques using geo-social access control policies. To the best of our knowledge, this is the first research effort to analyze geo-social access control systems with the objective of protecting a system against insider threats. Based on this analysis, we propose our Geo-Social Insider Threat Resilient Access Control Framework (G-SIR) which is the first to capture acceptable and unacceptable geo-social behavior and incorporate it into the access control decision process. G-SIR includes new types of geo-social constraints and protects against proximity attacks, collusion attacks and some attacks that can be launched by the access requester.

1.5 DOCUMENT ORGANIZATION

The remainder of the dissertation is organized as follows. In Chapter 2, we present the background and related work. In Chapter 3, we present the detailed motivation and requirements that have led to the design of the three frameworks that are part of this dissertation. We also highlight the detailed contributions of each of the three proposed frameworks. In Chapter 4, we present the proposed *adaptive risk management RBAC framework*. Then, in Chapter 5 we present the proposed *obligation-based framework*. In Chapter 6 we present our *geo-social insider threat resilient access control framework*. Finally, in Chapter 7, we present the conclusions and future work.

The work presented in Chapter 4 has been previously published in [11, 12]. The material in Chapter 5 was previously published in [13]. Some of the content used in our geo-social framework was previously published in [14].

2.0 BACKGROUND AND RELATED WORK

In this Chapter, we present the background information and the related work of our research. We begin by presenting background on *insider attacks* (section 2.1). Then, we present *risk* and *trust* definitions and related work in these areas (sections 2.2 and 2.3). After that, we present RBAC basics and related work that include risk and trust (section 2.4). Then, we present the state of the art of obligation-based access control systems (section 2.5). Finally, we introduce some background information on geo-social access control approaches (section 2.6).

2.1 INSIDER ATTACKS

Insider threats have been broadly classified as *intentional* and *unintentional threats* [112] which are defined as follows.

1. Intentional insider threats: We define intentional insider attackers as individuals who have legitimate access to the resources of an organization and decide to attack the organization by disrupting its availability and compromising confidentiality or integrity of assets owned by the organization. The motives of insider attackers vary from highly calculating individuals motivated by personal gain (e.g., steal intellectual property) to disgruntled employees aiming to hurt the organization (e.g., sabotage the operations). In [56, 58], several incidents have been documented.

Modeling insider attacks successfully requires the identification of important indicators that may be used to determine if a user is misbehaving. If an employee is accused unfairly, he may worsen his performance, increase his level of disgruntlement, reduce his trust towards the organization and in the worst case, become an actual attacker. Technical and psychological precursors discussed in [81] are often used to identify misbehaving users. The set of technical precursors utilized is highly dependent on the type of system that is modeled. In particular, depending on which access control model is used, different indicators may be available. For instance, in a role based system, it is possible to profile users based on their behavior with respect to the behavior of all other users that share the same role [21, 37]. Several methodologies have been proposed to monitor different aspects of users' behavior that include monitoring their search behavior for files [98], the content of emails [111], among others, e.g., ([47, 115, 25]). A survey on insider threat detection can be found in [97]. To integrate all available anomaly detection data, *situation-aware systems* such as the one proposed in [27, 73] can be used.

Situation-aware systems should also include psychological precursors to boost their prediction value. However, including these indicators is a challenging task. Among the *psychological precursors*, insider attackers generally show the following symptoms: disgruntlement, bad attitude towards feedback, anger management issues, disengagement, disregard for authority, performance decrease, stress, confrontational behavior, personal issues, self-contentedness, lack of dependability and absenteeism [58]. Although psychological indicators may allow early detection of insider threats, monitoring users can be challenging because of privacy and legal concerns. Several of the indicators proposed in [58] are related to psychological and physical characteristics that are usually seen as private information. For instance, monitoring the health of an individual is not well regarded; indeed, the Health Insurance Portability and Accountability Act (HIPPA) protects individuals' right to medical privacy [86]. Hence, including psychological and physical monitoring of a user would breach the HIPPA legislation and other privacy related legislation. Therefore, new techniques to measure these indicators without violating the privacy of the employees are needed.

Greitzer *et. al* [61] propose to find these indicators using human input, which is subjective in nature and may be biased due to interpersonal relationships. For instance, asking an insider to evaluate these indicators will inevitably lead to a subjective evaluation based on how he sees his co-workers. Another approach is to ask employees to report suspicious behavior. This is not usually a successful practice as people tend not to report such information for fear of incriminating a co-worker who is possibly innocent or because they think someone else is going to report the suspicious behavior [92]. Existing practices to measure the employee's psychosocial state usually result in outdated information. For example, 360-performance evaluation methodology is typically completed once a year [58].

Ideally, a system should identify the psychological state of users without including subjective opinions or compromising their privacy and at the same time, it should include this information as quickly as possible. By doing so, it is possible to take advantage of the high predictability factor of psychological indicators [9, 60].

2. Unintentional insider threats: Recently, unintentional insider threats have been defined as threats that occur *"through action or inaction without malicious intent that causes harm or substantially increases the probability of future serious harm to the confidentiality, integrity, or availability of the organization's information or information systems"* [112]. Social engineering attacks, phishing, fatigue-related incidents, among others are classified as part of unintentional insider threats.

In this dissertation, our primary focus is to mitigate intentional insider threats. However, for our obligation-based and geo-social framework, we also include some design elements that aim at preventing unintentional insider attacks.

2.2 RISK

Risk is the cornerstone of the proposed research. Risk is defined by the likelihood of a hazardous situation and its consequences if it occurs [83]. To identify the impact of an event we use a probabilistic risk analysis as defined by Kaplan *et. al* in [74], where risk can be calculated using the expected value formula. First, all possible outcomes are found and quantified and then each outcome is weighted by its probability.

In information security, risk assessment methodologies such as the NIST risk management methodology [108], Octave [7], [4], among others, allow organizations to identify threats and evaluate their risks to determine an appropriate course of actions. The ultimate objective is to determine if it is appropriate to reduce the likelihood of occurrence of a particular threat through the implementation of policies, controls and mechanisms in the system. When controls are implemented the risk is said to be mitigated. Otherwise, it is said to be accepted. The risk exposure after all the controls and mechanisms are in place is called *residual risk*, and ideally, it is the risk that the organization is willing to accept. These methodologies usually focus on high level assets and are often performed once a year. Complementary risk mitigation techniques are needed to allow a more frequent and automatic prevention of insider attacks.

2.3 TRUST

Trust is another key concept in our research. Several definitions of trust have been provided in the literature [71, 82, 57, 87]. We adopt the following trust definition: *"Trust is a subjective expectation an agent has about another's future behavior based on the history of their encounters"* [82]. Trust may depend on the *context* in which the interaction between entities takes place. For instance, the type of service and the network connection used by the user may define a context. Our framework requires the use of methodologies to find trust values for users given their current and historic behavior. Several approaches for calculating trust in different domains have been proposed. A comprehensive survey of trust methodologies can be found in [34]. In Chapter 5.3 we show that existing trust methodologies are not directly applicable for obligation-based risk management.

2.4 ADAPTIVE ROLE-BASED ACCESS CONTROL APPROACHES

Because all access control models studied as part of this research corpus extend the Role-Based Access Control (RBAC) model, we begin by presenting an overview of RBAC model. Then, we present the related work on RBAC, risk and trust and we point out the limitations of existing work that we propose to address as part of this dissertation.

2.4.1 Background on RBAC, Constraints and Hybrid Hierarchy

Role Based Access Control (RBAC) model [50] has multiple benefits. It encompasses discretionary and mandatory access control models and supports organization or user-specific requirements. In addition, RBAC uses roles, which are a natural abstraction for most organizations, and it provides organizations with economic benefits due to the reduction of the administration cost [95].

In RBAC, *permissions* are assigned to *roles*, and roles are assigned to *users*. In order to obtain the permissions authorized for a role, users need to *activate* the role in a *session*. Sets U, R, and P represent the set of users, roles and permissions in the system, respectively. Separation of duty constraints (SoD) are used to avoid fraudulent activities within an organization by preventing a unique user from assuming two or more conflicting roles. There are two types of SoD constraints: Static (SSoD) and the Dynamic (DSoD). SSoD restricts the authorization of users to conflicting roles [6]. Each constraint is denoted as $ssod(RS, k) \in SSoD$, where $RS \subseteq R$ with $2 \leq k \leq n$. This constraint states that a user can be *authorized* to at most $k-1$ roles in RS. Similarly, a DSoD constraint $dsod(RS, k) \in DSoD$ states that a user can *activate* at most $k-1$ roles in RS simultaneously.

There are two types of cardinality constraints. An *activation cardinality constraint* restricts the number of users that can *activate* a particular role in a system simultaneously. To denote that a role r can be activated at the same time by at most $k-1$ users, we use the notation $card(r, k)$. An *assignment cardinality constraint* restricts the number of users that can be assigned to a role. This is denoted as $card_A(r, k)$.

Roles can be hierarchically organized using *hybrid hierarchy* [101]. Roles r_1 and r_2 can be hierarchically related in one of the following ways. (1) **I-hierarchy** ($r_1 \geq_I r_2$) where r_1 inherits the permissions of r_2. (2) **A-hierarchy** ($r_1 \geq_A r_2$) where users assigned to r_1 can activate r_2. (3) **IA-hierarchy** ($r_1 \geq_{IA} r_2$), in this case, r_1 is I-senior and A-senior of r_2. The hybrid hierarchy allows the enforcement of different types of policies such as DSoD when roles are hierarchically related [101].

2.4.2 Related Work on RBAC Extended with Risk and Trust

Although RBAC has several benefits, it cannot automatically revoke access to users that are not behaving properly. For this reason, several approaches have incorporated *trust* [30, 49, 46]. However, existing approaches do not present a comprehensive analysis of the way in which trust thresholds should be assigned, do not include separation of duty constraints nor specify how to enforce such policies or reduce the risk exposure automatically. In [30], roles are associated with trust intervals, and trust intervals are assigned to users. Users are assigned to roles according to their trust levels. This model does not capture the intuitive nature of RBAC systems in which users are assigned to roles according to their organization's functions, not trust levels. In [49], users are assigned to roles based on trustworthiness and context information. A similar approach was proposed in [46], where role thresholds are a function of the risk of the operations. If the trust of the user offsets the risk of the action, the access is granted. However, none of these works provide a clear framework to compute trust thresholds and do not reduce the risk the organization faces at runtime by selecting roles with minimum risk exposure.

In [80], each role is assigned a minimum level of confidence and each user a clearance level. Based on these values, the risk associated with a user activating a role is calculated. Objects and actions are assigned a value according to their importance and criticality. However, this work does not mitigate insider threats as the trust levels of users is a static value that does not depend on users' behavior. In addition, Ma *et. al* [80] do not consider role hierarchy in their work and do not present experimental results.

In [84, 8, 32], the main focus is also to reduce the risk exposure. In [84], a risk based analysis is proposed to ensure that system administrators assign permissions to the roles considering the risk inherent to those permissions. Each permission is assigned a risk value, and the role hierarchy is organized based on these risk values. This may not be appropriate, as it is more intuitive to organize the role hierarchy according to the employee's structure. We argue that maintaining a role hierarchy that matches the organization's hierarchy is more intuitive for security administrators. Additionally, this work does not reduce the risk exposure of the organization during the role activation process. In [8], a model that modifies the policy to minimize the risk exposure as systems evolve is proposed. This model results in a difficult to manage policy in which the administrator does not know the current status of the policy; making it cumbersome to modify it and prone to errors. Chen *et. al* [32] propose a model in which the risk associated with a role is calculated using the trustworthiness of the user, the degree of competence he has to activate a role, and the degree of appropriateness of the permission-role assignments. Each permission is assigned a mitigation strategy, which is a list of risk thresholds and an associated obligation pair. When a user wants to obtain a set of permissions, the role with minimum risk is selected. Then, the system consults the mitigation strategy to see which action is more appropriate: to deny the access or to allow the access imposing an obligation. Chen *et. al* do not consider SoD constraints, which is crucial for addressing insider attacks. Additionally, they do not account for the context as an important component to define the risk threshold that should be enforced. Chen *et. al* use the appropriateness of permission to role assignment as part of the risk computation. We believe this makes the semantics of permission to role assignment complex, as the appropriateness value becomes a functional input for such assignments. This may result in too many inappropriate assignments -although they will likely be captured through risk computation. Hence, this causes unnecessary complexity in the administration of the policy. Additionally, Chen *et. al* do not provide an algorithm to enforce the policy to reduce the risk exposure during the role activation process.

Salim *et. al* propose to assign costs of access to permissions depending on the risk of their operations, and to assign to each user a budget in [99]. Users are assigned to roles, but being assigned or not does not necessarily determine whether or not a user should be allowed to activate a role. If the user accesses permissions that he can obtain through an authorized role, the cost is reduced. In case a user is not authorized to a role, the cost of activating the role is *taxed*. Nonetheless, if the user has enough budget to make the operation, he can access the permissions. Salim *et. al* [99] claim that this mechanism incentivizes users to spend their budget cautiously, activating low cost (low risk) roles. However, this scheme exacerbates the risk of insider threats. Users can use their budget to perform unauthorized accesses without being detected; e.g., if a disgruntled employee wants to quit the organization, he would not mind expending all his budget performing a malicious action.

Many commercial products also incorporate risk in their solutions; e.g., SAP [102], Oracle [88], IBM [65] and Beta Systems [109]. These products mitigate risk by closely monitoring and auditing the usage of risky permissions. The risk values, however, are not used to make access control decisions, missing the opportunity to incorporate the overall known behavior of the users to prevent insider threats.

The threat of inference of unauthorized information is particularly relevant in the insider threat context. This threat occurs when through what seems to be innocuous information, a user is capable of inferring information that he should not have access to. In existing approaches to deal with inference threat [26, 45, 23], when a user is about to infer some unauthorized information, the system prevents it by either denying access or providing scrambled data. This is not adequate for all types of organizations. We believe that real organizations may need to provide access to multiple pieces of information to a single employee even if they result in undesirable inference. Existing RBAC extensions do not consider the risk of inferred information. New ways to mitigate the inference risk in RBAC-based systems are needed.

In summary, to the best of our knowledge, none of the related work has provided an analysis of the way the roles should be activated to mitigate risk of insider threats. For this purpose, there is need to have a comprehensive methodology to identify how risky an access is. In addition, current literature often does not include hybrid hierarchy, SoD and cardinality constraints or enforce least privilege. These constructs are crucial to provide flexibility during policy specification as well as reducing the risk exposure caused by insider threats. Another limitation of existing approaches is that they do not mitigate the risk of inference of unauthorized information. Finally, none of these works provides tools for administrators to validate policy correctness. This research avenue has been explored in non-adaptive access control systems e.g., [107, 69, 54], however it has been neglected in risk and trust-aware access control systems despite their increased policy specification complexity.

2.5 OBLIGATION-BASED ACCESS CONTROL

Many application domains, including healthcare information systems, require the inclusion of obligations as part of their access control policies [86, 91, 66]. An obligation is an action that needs to be performed before a deadline passes [67]. The Health Insurance Portability and Accountability Act (HIPAA) provides some examples of obligations, e.g., "when a patient sends a request to access her protected health information, the doctor must respond to that request within 30 days". Here the deadline is 30 days and the action is to respond to the patient's request. When an obligation is completed before the stipulated deadline, it is said to be *fulfilled*. Otherwise it falls into a *violated* state. We distinguish between *user-based* and *system-based* obligations. In *system-based* obligations, the system is in charge of performing the obligations while in *user-based* obligations the user is in charge of fulfilling the obligation. We further classify obligations in three categories based on when they need to be performed, these are: *a priori* obligations and *a posteri* obligations. *A priori* obligations need to be performed before an access to a resource takes places while *a posteriori* obligations need to be performed after an access takes place.

In this dissertation, we focus on user-based *a posteriori* obligations because they are very challenging to enforce as there is no guarantee that users will fulfill them. Additionally, they are particularly prone to sabotage threats. In what follows, we present the related work to obligations.

2.5.1 Related Work on Obligations

To the best of our knowledge, none of the existing work has recognized that not fulfilling an obligation has an inherent risk for organizations. Most existing work related to obligations focus on providing accountability in the system [36, 67]. The idea is to assign *a posteriori* obligations to the users in such a way that the only reason for the obligation to fall into a *violated* state is user's incompetence. Li *et. al* [79] propose an XACML extension to specify obligations as state machines. In [120], an RBAC policy augmented with obligations is presented. None of these works include risk management as part of the decision making process to assign an *a posteriori* obligation to a user.

Bettini *et. al* propose calculating a reliability value based on the history of fulfillment of obligations in [22]. However, the work limits itself to providing a syntax to include this value into the obligation policy and does not provide a methodology to calculate it. Their approach assumes a trust methodology is available to identify users' intentions. Additionally, no methodology to find policy misconfigurations, colluding and suspicious users is provided. Other approaches have tried to reduce the risk exposure through the use of *system* obligations (e.g., [32]), which are obligations performed by the system itself. These obligations are meant to mitigate the risk, e.g., an obligation may consist of having the system close a file after a low trusted user has accessed it. System obligation are a valuable approach to deter insider threats, but they do not mitigate the risk associated with *a posteriori* obligations.

Although several approaches combine access control with risk and trust [72, 103, 11, 100], to the best of our knowledge none of them considers risk when assigning *a posteriori* obligations to users nor provides a trust based methodology to do so. In [72], an abstract model for incorporating the concept of risk in Usage Control (UCON) [90] is presented. They consider risk coming from components such as the user, object, operations, connection used as well as the provenance of attribute certificates. However, they do not include the obligations as part of the risk components. We believe that it is relevant to incorporate *a posteriori* obligations in the risk assessment as obligations are inherently risky.

2.6 GEO-SOCIAL ACCESS CONTROL

Several works have extended RBAC to include the context of the user such as the location and temporal constraints as part of the access control decision [20, 31, 114, 41, 94]. We broadly classify the existing RBAC literature into two categories namely RBAC extensions that support location-based decisions [20, 31, 114, 94] such as Geo-RBAC [20] and LoT-RBAC [31] and models that extend RBAC with proximity constraints that include other user's proximity as part of the access control policies such as Prox-RBAC [76, 63]. In Table 1, we compare existing approaches with our previously proposed Geo-Social RBAC [14] based on the following types of constraints:

1. **Pure location constraints:** these constraints only take the location of users into account, e.g., to access a confidential file, a user may need to be in a particular room.
2. **Geo-social constraints:** these constraints consider both the location and the social dimensions of the users in the policies. These are further classified as follows. *(i) Geo-social graph-based constraints* which are based on the social graph structure, e.g., to enter into a room a person needs to be in company of at least two friends that work there and are present. *(ii) Geo-social tag-based constraints* which capture the types of relationships between the users in the social graphs in addition to the location and social constraints. For example, a child can only access a pay-per-view movie if he is in the presence of his parent or a nanny.
3. **Trace-based constraints:** These constraints are based on user's trajectory and whether the user has been physically co-located with a particular set of individuals. These include:
   - *Location trace-based constraints:* which capture the past location traces of a user as part of the access control policies. For instance, consider a silicon chip manufacturer company where even a minimum amount of dust may ruin an entire production batch. If an operator has been in known dusty rooms of the factory, he cannot enter the sterile chip production room unless he has previously passed through the cleaning room. This is a location trace policy as the previous whereabouts of the user determine whether or not he would be able to obtain the requested access.
   - *Geo-social trace-based constraints:* which capture both the location history and the social dimensions of the users. For example, in a company, if a visitor has entered into the rooms used for induction of new employees accompanied by an administrator, he can also access the welcome package files and the internal directory web pages.

As shown in Table 1, existing models do not support many geo-social constraints that our previously proposed Geo-Social-RBAC incorporates. For this reason, in the remaining of this dissertation, we focus on this model. In the remainder of this subsection, we examine more closely approaches that have included the geo-social context as part of access control systems [76, 63]. Prox-RBAC model [76] extends the Geo-RBAC model to include proximity

Table 1: Comparison of types of geo-social policies supported by existing RBAC based models.

Policy	RBAC extended with location [20, 31, 114, 94]	RBAC extended with proximity [76, 63]	*Our Approach:* Geo-Social-RBAC [14]
Pure location constraints	Yes	Yes	Yes
Geo-social graph-based constraints	No	Yes	Yes
Geo-social tag-based constraints	No	No	Yes
Location-trace-based constraints	No	No	Yes
Geo-social-trace-based constraints	No	No	Yes

of other individuals as part of the policy in indoor environments. In Prox-RBAC valid proximity constraints are based on the role of the access requester and the roles of other individuals in proximity of the requester. This model does not allow the specification of geo-social constraints based on social graphs. Gupta *et. al* [63] extended Prox-RBAC by providing formal definitions to determine the proximity between locations, users, attributes and time, each of which is referred to as a realm. The access control model does not include hybrid realm policies. Additionally, their work does not allow the specification of some of the policies presented in Table 1.

Other non-RBAC based models have been proposed in the literature [113, 53, 28, 110]. Besides not being RBAC-based, none of them are designed to protect against insider threats. TMAC [113] is a model to establish policies that require team cooperation. Fong present ReRAC [53] where decisions are based on the relationship between the resource owner and the access requester. Carminati *et al.* [28] propose an access control model where policies are expressed based on user-user and user-resource relationships. In [110], access control decisions are made based on the location of the resource owner, the resource requester and possibly other co-located individuals. Their model assumes that individuals own the resources and it is not based on RBAC, making it less suitable for company settings. Also, it does not consider trace-based constraints.

To the best of our knowledge, none of the existing approaches have been designed to capture the intricacies of creating a geo-social access control model capable of mitigating insider threats. In particular, they do not include risk and trust as part of the access control decision making process. As part of this dissertation, we present some insider threats that arise when existing geo-social models are used. We also provide policy constraints and enforcement mechanisms to mitigate these threats.

3.0 REQUIREMENTS AND CONTRIBUTIONS

In this chapter, we present the requirements and contributions of each of the three frameworks proposed as part of this dissertation. We begin by our RBAC-based framework in Section 3.1, followed by our obligation-based framework in Section 3.2 and conclude the chapter with our G-SIR framework in Section 3.3.

3.1 AN ADAPTIVE RISK MANAGEMENT RBAC FRAMEWORK

The proposed framework aims to reduce the attacks and misuses performed by insiders when using an RBAC-based access control system. We envision a model simple enough to abstract the fact that information to assess how trusted a user is may be coming from multiple sources. Failing to hide this complexity or to avoid having multiple sources of information to determine how trusted a user is would render the model unusable. No previous assumptions on trust levels of the users should be made based on their rank or on fixed security clearances such as top secret, secret, non-confidential, etc., to allow flexibility and applicability to multiple types of organizations.

This framework needs a risk a methodology to assess the risk exposure of each access request. Recall that in RBAC, to acquire a permission, it is necessary to activate a role that has the permission assigned to it in a session. Each permission may have a different risk value associated with it; hence, we need to design a suitable *risk aggregation methodology* to determine the risk exposure of an organization if a particular access request that requires the activation of a set of roles is to be granted. The risk aggregation methodology should be designed to account for different ways in which acquiring multiple privileges simultaneously may be used to attack the system. We consider two factors. The first factor is the risks associated with the permissions acquired through the roles. The second factor is the risk due to inference of unauthorized objects, which we call *inference risk* and is caused by multiple accesses. The inference risk arises when providing a set of permissions to a user allows him to infer information that, in principle, he should not have access to. Although ideally, this should never be the case, given the limited number of users in organizations, security administrators may specify access control policies that often enable undesirable inference of information [15, 43]. For this reason, the aggregation methodology should account for inference risk to manage it appropriately.

3.1.1 Requirements

We identify the following requirements.

1. The access control model should allow the specification and enforcement of separation of duty, cardinality constraints as well as the hybrid hierarchy to support fine-grained access control policies. The importance of these components for insider threat prevention was outlined in Chapter 2.
2. The system should detect suspicious activities. This process should be automatic and should be able to establish to which level each user is to be trusted by the system.
3. It should be possible to associate different trust values for a user depending on the user's and system's context. We stress the importance of including the *context* in which the access is taking place. For example, the risk of a user accessing a confidential file from a machine without connection to the Internet is less compared to the risk associated with the same access using a personal device from a remote location.
4. Since different permissions may have different risks associated with them, the system should be able to react to suspicious changes in the behavior of users by removing access to riskier permissions quickly, and if the misuse continues, to other permissions as well.
5. The framework should include a *risk aggregation methodology* to determine the risk associated with the activation of a set of roles by a particular user. The risk associated should include the imminent risk associated with the permissions acquired through the

roles, and the risk due to inference of unauthorized objects. The risk exposure should be *automatically* reduced, minimizing the impact of possible attacks.

6. The system should provide the security administrator the ability to identify the active inference threats associated with a particular policy, so he can decide whether the policy needs to be modified to reduce the risk exposure due to inference of unauthorized information.

In this dissertation, we focus on requirements 1, 3, 4 and 5. For requirement 2, anomaly detection solutions, such as those presented in Chapter 2, can be used to monitor the behavior of users and calculate how trusted they are.

3.1.2 Contributions

The details of our proposed risk-and-trust aware RBAC framework are presented in Chapter 4. The following are the **contributions** associated with the proposed Adaptive Risk Management RBAC Framework.

- We propose a model that includes risk and trust in RBAC systems that adapts to anomalous and suspicious changes in users' behavior.
- We propose a comprehensive approach to calculate the risk values associated with permissions and roles. In particular, we introduce the notion of inference of unauthorized permissions when calculating the risk of activation of a set of roles. For this purpose, we present a formulation of a Coloured Petri-net (CP-net) [70] to identify when a particular user may infer unauthorized permissions, and subsequently adjust the trust threshold required to activate needed roles.
- We propose a refinement methodology to reduce the amount of information stored and the performance of the CP-net used to identify the risk exposure due to inference of unauthorized information.
- We formulate an optimization problem to enforce the policy and reduce the risk exposure. To the best of our knowledge, this is the first work that attempts to reduce the risk exposure in this way.

- We present a role activation algorithm to solve the optimization problem, and evaluate its performance using well-formed policies and prove its correctness.
- In order to improve the risk management process related to inference threats, we propose a simulation strategy that allows administrators to identify active inference threats before a policy is deployed. In addition, an administrator can determine the effects of adding a user-to-role assignment before he modifies the access control policy in the production system. This methodology helps reduce undesirable inference threats.

3.2 OBLIGATION-BASED FRAMEWORK TO REDUCE RISK EXPOSURE AND DETER INSIDER ATTACKS

At the core of this framework is a methodology to find users' trust values based on patterns of violation and fulfillment of their assigned obligations. Such a methodology should withstand powerful adversaries. We propose the following **threat model** with two types of adversaries that are representative of possible insider attackers:

1. *Naïve users:* These are insiders who know the system is monitoring if they have fulfilled or violated a particular obligation. However, they do not know the details of how their trust values and trust thresholds to access resources are computed by the system.
2. *Strategic users:* These are insiders who have knowledge about the system's mechanism to compute trust values. This information gives them the power to try to maintain their trust levels within the expected thresholds to avoid being flagged as suspicious by controlling their behavior in a smart way.

In chapter 5.3, we show that the existing approaches do not withstand this adversarial model. Having defined our adversarial model, we now present the requirements of this framework.

3.2.1 Requirements

The proposed obligation-based framework should address the following requirements for detecting and mitigating the risk exposure of unfulfilled *a posteriori* obligations.

1. The associated access control model should capture the criticality of obligations. A criticality value represents the severity of the impact of not fulfilling an obligation for the organization.
2. Reduce risk of users not fulfilling obligations by considering their trust values and the criticality of *a posteriori* obligations associated with the permissions being requested. The system should deny access requests to users whose trust values are below a pre-specified threshold associated with *a posteriori* obligations that would be triggered by the requested accesses.
3. Develop a methodology to compute the obligation related trust value of a user based on the history of fulfilling or defaulting on *a posteriori* obligations as well as his performance with respect to his peers. The trust value should detect when a user is an outlier; e.g., when the user is the only one defaulting on a particular obligation. The proposed methodology should be reliable against both *strategic* and *naïve* adversaries.
4. Provide a methodology that allows an administrator to detect policy misconfigurations related to *a posteriori* obligations by identifying patterns of violation of *a posteriori* obligations. The patterns can serve to identify when a particular obligation is not being fulfilled by a large number of users. This may be due to different factors. It is possible that the policy is not updated, but there is a verbal or implicit agreement to ignore it or the users that are assigned those obligations are too busy or lazy. The system should also detect when a user is the only one continuously violating an obligation, which may imply he is sabotaging the operation. The knowledge of these patterns can be used to reduce the risk and identify policy misconfigurations.
5. Identify when a user is misbehaving, which in turn indicates that he poses a high risk of becoming an insider attacker, without invading users' privacy.

3.2.2 Contributions

The proposed *Obligation-based Framework to Reduce Risk Exposure and Deter Insider Attacks* is presented in Chapter 5. The **contributions** of this framework are as follows:

- We emphasize and show that *a posteriori* obligations have an inherent criticality level and propose a comprehensive framework to reduce the risk exposure faced by organizations every time a user is assigned critical obligations. To the best of our knowledge, this is the first work that has integrated the inherent criticality of *a posteriori* obligations and the obligation-based trust values in the authorization decision-making process.

- We propose and evaluate a methodology to calculate the obligation-based trust values for each user. The methodology is resilient against users who know how the system computes the trust values and try to exploit this knowledge. Our methodology is also able to discern among users who accidentally do not fulfill an obligation, maliciously avoid the fulfillment of obligations and those who strategically oscillate their behavior to maintain their trust values within an acceptable threshold to later launch an attack.
- We propose the use of a clustering-based methodology to identify policy misconfigurations, users colluding to avoid performing particular obligations and users whose behavior is worse than their peers (e.g., users that systematically avoid fulfilling an *a posteriori* obligation). This information can be used by the system administrator to take necessary actions, such as updating the policy or monitoring more closely certain users.
- Finally, the proposed framework provides a technique to detect insider threats by monitoring users without invading their privacy (e.g., other methodologies used for this purpose scan users' personal emails) or including subjective measures.

3.3 GEO-SOCIAL INSIDER THREAT RESILIENT ACCESS CONTROL FRAMEWORK

Geo-social information can significantly help to deter insider threats. When an organization establishes a geo-social access control system, it creates a unique opportunity to use the information collected by the infrastructure to account for users' behavior and make adequate access control decisions. These types of controls help prevent some insider attacks. For example, a user who is often at places that he is not supposed to frequent should be flagged as suspicious and actions to restrict his access to highly critical information should be automatically performed. This behavioral information should be considered at the time access control decisions are evaluated.

However, designing a system that uses this information without increasing the risk exposure is a challenging task. Before outlining the concrete challenges and showing where existing techniques fall short, we introduce the relevant actors and components of the proposed system.

3.3.1 System Actors

A geo-social access control system has a social network graph, where nodes represent users and edges represent relationships among them. These relationships are annotated with labels that represent the types of social relationships, e.g., boss. Additionally, a geo-social system has access to the location where users are at any particular time. Users may issue access requests and a policy can be defined to determine if an access request should be granted or denied. Geo-social access control systems also consider where the requester is located and who the users in the vicinity are. This information is very useful because it helps determine when the access request context is not adequate to grant a requested access.

We classify users in the vicinity in three classes: *enablers*, *inhibitors* or *neutral* users according to the way in which they impact the risk exposure associated with granting an access request. *Enablers* are users that may actually bootstrap and/or enhance the trust of an access request by vouching for the requester due to their social relationship. *Inhibitors* on the other hand, are users whose presence increases the risk of granting an access and *neutral* users are those whose presence does not increase or reduce the risk of a request. For example, consider a policy that requires a parent or a nanny to be in the same room with a child requesting an access to a pay-per-view movie. Here, the parent or nanny are enablers and the child is the requester. In contrast, inhibitors increase the risk of granting access to a request. An example of an inhibitor is a consultant trying to access sensitive information in presence of another consultant working for a competing company.

3.3.2 Insider Threats

A geo-social adaptive system to deter insider attacks should be able to determine the risks associated with these actors whenever an access request is evaluated. The risk exposure increases with respect to adaptive access control systems because enablers can influence the access control decisions as indicated by the following threats.

1. Collusion: The requester and enablers may decide to collude and probe the system to try to access information that they would not ordinarily have access to. Ways to collude to probe the system include changing the current location or trying to modify the social graph to gain more accesses.

These types of collusion attacks are new and have not been considered by existing adaptive access control models. Although existing geo-social access control models make use of statically defined and enforced *geo-social cardinality constraints* to reduce the risk of collusion, these constraints are not enough. A geo-social cardinality constraint is a rule that helps establish how many people need to be at a particular location for a user to be able to exercise a privilege [76, 63, 14]. Even if there is evidence that suggests a group of people is colluding, existing geo-social cardinality constraints disregard this information. As a consequence, colluding users may gain access to critical information despite availability of evidence of their malicious efforts.

2. Social engineering attacks: Social engineering attackers convince other users to perform an action that they should not perform under normal circumstances. For instance, an enabler may be tricked by a malicious requester through a social engineering attack to move to a location to allow his request to be granted. Similarly, the requester may be tricked to enter into a particular place and access some information.

3. Proximity threats: Users in the vicinity create multiple risks based on the groups to which they belong (e.g., conflicting projects, or being part of social communities that are undesirable for a particular access). When a user in the vicinity poses too much risk, she is classified as an inhibitor. A framework for insider mitigation needs to be able to specify that whenever there are one or more inhibitors, the access control system should deny the access.

4. Inadequate policy enforcement: Although existing geo-social access control systems specify policies that control access to some privileges based on the geo-social context of a user, they do not account for negative geo-social behavior. Undesirable behavior may not be prevented by an access control policy for reasons that include high costs of enforcement, inconvenience, and people working around enforcement mechanisms in place, as the following example illustrates. A user may enter a restricted area (e.g., by door piggybacking), where he should not be; however, he does not request any access while in the forbidden place. In this scenario, current geo-social access control systems are blind to the fact that the user entered into a forbidden place. Although it is understood that the user's behavior is inappropriate, no enforced access control policy is impacted by her behavior. Thus, current geo-social access control policies are not enough to detect negative geo-social behavior when it is *not* linked to an access request. As a result, dangerous behavior may not be captured.

Given this inability to enforce desired policies, often users are informed of the geo-social behavior they are expected to fulfill and are blindly trusted to do so. Such desirable behavior can be enforced through *social contracts* [24], which are a tacit or verbal understanding between interested parties about each other's expected behavior. We are interested in social contracts that specify the whereabouts and relationships that are appropriate or inappropriate for the role that users play within an organization.

Preventing inappropriate geo-social behavior is a daunting task, and new techniques need to be devised to capture violations of social contracts. Although it is difficult to enforce, through proper monitoring, it is possible to identify inappropriate behavior and raise an alert.

5. Privilege misuse threats: These threats occur when a requester decides to abuse his privileges. Our framework should also mitigate them by using historical behavior.

3.3.3 Requirements

Towards addressing these insider threats, we now discuss the requirements for the proposed *Geo-Social Insider Threat Resilient Access Control Framework*:

1. Provide policy constructions to classify users in the vicinity according to the risk they

impose given an access request into enablers, inhibitor or neutral users.

2. Define policy constraints to capture geo-social behavior relevant to access control decisions. In particular, the system should allow the specification of the following types of policy constraints.

   i) *geo-social contracts*, which specify places and people that a user cannot visit by virtue of being assigned to a role in an organization,

   ii) *geo-social obligations*, which are geo-social actions that a user needs to perform after an access has been granted. Geo-social actions include visiting or refraining from visiting a particular place or person, and

   iii) *trace-based constraints*, which reflect expected paths that users need to complete before being granted an access.

3. Restrict accesses where the requester or any of the enablers are violating any of his social contracts.

4. Monitor and analyze the behavior of users with respect to the fulfillment of geo-social policy constraints. Users violating policy constraints more often than their peers are suspected of disregard of authority and, hence, should be trusted less. Therefore, the estimated probability of the requester being an attacker should include geo-social policy violations.

5. Mitigate the risk of colluding users by identifying communities of colluding users and restricting accesses where there is a strong indication that the enablers and the requester are colluding.

6. Ensure that the access control system can adapt to negative changes in behavior of users by restricting critical privileges to users who do not behave properly. The decision to grant or deny an access should consider the risk exposure. G-SIR should minimize the risk exposure caused by the requester, users in the vicinity and potential collusion among enablers and the requester.

3.3.4 Contributions

We propose a Geo-Social Insider Threat Resilient Access Control Framework (G-SIR). G-SIR is capable of deterring insider attacks by considering users' geo-social context, their behavior and the risks associated with granting access to a set of permissions. In Chapter 6 we present in detail G-SIR. The **contributions** related to this framework are summarized as follows:

- To the best of our knowledge, this is the first research effort to analyze geo-social access control systems with the objective of protecting a system against insider threats. We present threats that are enabled by current geo-social access control systems.
- To mitigate these threats, we propose an access control model that includes a set of geo-social constraints to capture acceptable and unacceptable geo-social behavior. The proposed constraints include geo-social contracts, geo-social obligations, traces and vicinity constraints.
- We propose a risk management framework that incorporates geo-social behavior of the users and adaptably tunes the access control decision to minimize the risk. As part of this process, G-SIR monitors users who violate geo-social constraints to improve accountability and determine how trustworthy users are. The risk management procedure considers: *i)* how trustworthy the system considers the user is with respect to his geo-social behavior, *ii)* the user's current geo-social context, *iii)* the context of relevant social relationships, *iv)* existing indications of collusion among individuals in the vicinity, and *v)* other users in the vicinity who may compromise the security of accessed information.
- Finally, we evaluate G-SIR through simulations to demonstrate its effectiveness and feasibility.

In the following chapter, we present first and most general framework to deter insider threats.

4.0 AN ADAPTIVE RISK MANAGEMENT RBAC FRAMEWORK

In this chapter, we present the proposed Adaptive Risk Management RBAC Framework. We begin by presenting some preliminaries in Section 4.1. In Section 4.2, we present the requirements of the system and an overview of the proposed framework. The details of the risk calculations are presented in Section 4.3. The formal definition of the role activation problem and the proposed algorithm is presented in Sections 4.4. In section 4.5, we present a CP-net based technique to find and manage the inference risk. Finally, we show the experimental results in Section 4.6.

4.1 PRELIMINARIES

In this section, we present the notation and concepts used through this chapter. In our risk-and-trust aware RBAC model, we incorporate hybrid hierarchy introduced in Chapter 2. We use function $P_{au}(r \in R)$ to denote the set of permissions that can be acquired through r; this includes the permissions directly assigned to r and those inherited through I and IA hierarchical relations. Similarly, $P_{au}(R_c \subseteq R)$ returns the authorized permissions of all the roles in R_c. Function $authorized(u \in U)$ returns the roles in R that are authorized for u (if u is authorized for role r, it means he can activate r). Function $activated(r \in R)$ returns the number of sessions that contain role r. Finally, we denote the set of contexts as C.

4.1.1 Coloured Petri-net (CP-net)

We model the history of accesses as a *Coloured Petri-net* (CP-net) [70]. Here, we provide the basic concepts of CP-nets, and in Section 4.3.2.1, we present the proposed CP-net. A CP-net is a bipartite graph that contains two types of nodes: *places* (W) and *transitions* (T). Places and transitions are connected through *arcs* ($F \subseteq (W \times T) \cup (T \times W)$). No arc can exist between two nodes of the same type. *Tokens* (V) live in places, and move around in the CP-net when transitions fire. Usually, tokens represent objects and their attributes, which are called colors or types. Not all types of tokens are accepted in all the places. $\Upsilon(w \in W)$ denotes the type of accepted tokens in place w. Each transition $t \in T$ has a boolean guard that evaluates a condition based on tokens $V' \subseteq V$ located in the input place; a guard is represented by $G(t \in T,\ V' \subseteq V)$. If the guard evaluates to true, the transition fires. Otherwise, the transition does not fire. If a transition fires, it consumes the tokens that made the guard evaluate to true and collocates a new token in the output place(s). This is represented by the function $\lambda:\ t \in T \times V' \subseteq V \to v_o \in V$, where $G(t, V') = true$ and v_o is the token produced by the transition. We use m_o to denote the initial placing of tokens in the CP-net. Finally, a CP-net in its initial state is defined by tuple $\langle W, T, F, V, \Upsilon, \lambda, m_o \rangle$.

4.2 THE PROPOSED FRAMEWORK

4.2.1 Overview of the Framework

We consider RBAC systems with hybrid hierarchy, cardinality and SoD constraints. We extend this model by adding the following components (a detailed explanation of each of them is provided in Section 4.3).

- Each user is associated with a *trust value* that is a function of his behavior under a particular context. We denote this as $trust(u \in U, c \in C)$.
- Each permission is assigned a *risk value* within a particular context. We denote this as $rs(p \in P, c \in C)$.

- The policy contains a set of inference tuples $\mathcal{I}$, which allows the calculation of the risk exposure due to inference of unauthorized information.
- When a set of roles RS is to be activated, first its combined risk value is computed ($rs(RS \subseteq R, c \in C, u \in U)$) based on *(i)* the permissions it is authorized for, *(ii)* the inference risk associated with those permissions, *(iii)* the context, and *(iv)* the trust value of the user trying to activate the roles.
- Similarly, when a set of roles RS is to be activated simultaneously by a user, a *trust threshold* is computed based on the risk of RS. This threshold is denoted as $\tau(RS \subseteq R, c \in C, u \in U)$.

A user can activate a set of roles in a session if and only if *(i)* he is assigned to all the roles in the set, *(ii)* their activation does not violate any constraint, and *(iii)* he possesses a trust value greater than or equal to the trust threshold required for those roles. We formalize this notion in Section 4.4. In RBAC, only the first two conditions need to be fulfilled for a user to be able to activate a role. In our model, we also consider the trust value of the user, which allows the system to adapt to misbehaving users.

The proposed system architecture is shown in Figure 3. The *Monitoring Module* monitors the users in the system. The *Trust and Context Module* (TCM) uses the monitored information to identify the context, and calculate the trust value of each of the users accordingly. These trust values are stored in the *Trust Repository*.

The *Access Control Module* is composed of the *Enforcement Module* the *Administration Module* and the *Policy Information Point* (PIP). The policy of the system is stored in the PIP. The *Enforcement Module* is in charge of evaluating access requests and has several components, the *Policy Enforcement Point* (PEP), the *Policy Decision Point* (PDP), the *Risk Module* and the *Inference Module*. An access request consists of the set of permissions a user wants to acquire. The PEP intercepts all these requests, and ensures that the resources of the system can be accessed only if the policy authorizes it. The access requests are intercepted by the PEP, which sends them to the PDP. The PDP evaluates the policy according to the trust the system has on the user, the context, and the inference risk. The inference risk is computed by the *Inference Module* and the computations related to trust thresholds are performed in the *Risk Module*. In case the trust value of a user decreases

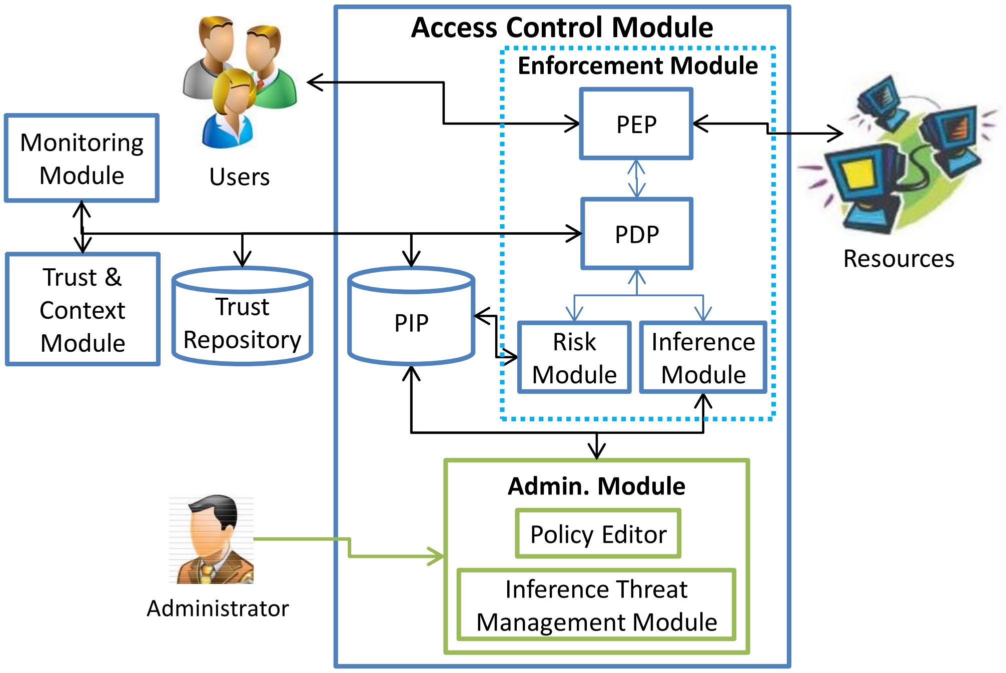


Figure 3: Risk-and-trust RBAC architecture.

while a user has a session open, the TCM sends a notification to the PDP, which re-evaluates whether the privileges the user is exercising should be revoked. In this way, the system is able to deny access to misbehaving users before they can perform extensive damage to the system.

The *Administration Module* enables the administrator to define, refine and analyze the policy. It is composed of two modules. The *Policy Editor* allows the specification of the policy and the *Inference Threat Risk Management Module* produces informs that identify the active inference threats of a particular policy configuration. Using this information, an administrator may iteratively modify the policy to reach the desired risk exposure. In this way, the administrator can identify the ideal policy, with respect to inference risk, before he realizes the policy in the production system. We discuss in detail this procedure in Section 4.5.

4.3 RISK AND TRUST THRESHOLDS

In this section, we present the proposed methodology to calculate the risk exposure of an organization. We show how the risk is calculated for different roles that a user wants to activate based on the risk of the permissions they can acquire. Finally, we show how to compute the trust threshold.

4.3.1 Risk Associated with Permissions

A permission is a tuple $\langle obj, act \rangle$ where the *obj* is an asset in the organization such as a file or other resource, and the *act* corresponds to the action that a user can perform on the object. Objects are susceptible to different threats. Among these are object's *loss of integrity, loss of confidentiality*, and *loss of availability*. Intuitively, different objects have different security requirements that depend on the business functions of a particular organization. For instance, some objects require that their integrity be well guarded, other objects are sensitive (their leakage would result in a lot of damage to the business), while others may be critical and sensitive simultaneously. Hence, the risk exposure of the organization depends on the action that is performed on the object and the relevance of the object.

The risk value of a permission p is the likelihood that p is misused multiplied by the corresponding damage cost. We are interested in the residual risk which means that the likelihood of a particular misuse depends on the mitigation mechanisms and controls that the organization has in place to reduce the vulnerabilities that can lead to the misuse.

Definition 1. *The risk of permission* $p = \langle obj, act \rangle \in P$ *in context* $c \in C$*, written as* $rs(p, c)$*, is defined as follows:*

$$rs(p, c) = \sum_{x_p \in MaliciousUsage_c} Pr[x_p|\ c\ ] * cost(x_p)$$

Where $MaliciousUsage_c$ *is a set of possible events in context* c *that can lead to a misuse of object* obj *through the action* act*,* $Pr[x_p|\ c\ ]$ *is the probability of occurrence of a particular malicious usage of object* obj *through action* act *given* c*, and* $cost(x_p)$ *is its associated cost.*

Example 1. *For simplicity, in this example, we only consider one context: users are accessing the system through the intranet. Consider an organization that produces soaps. To calculate the risk associated with permission* $p_1 = \langle listOfProviders, read \rangle$*, the organization performs the following analysis. The provider's list is considered to be sensitive, as its information provides the organization a competitive advantage. The organization calculates that its leakage would cost around $30,000. According to their system's configuration, the probability of occurrence of this event is 0.1. This results in a total risk of $3,000. Permission* p_2*, corresponds to writing the number of orders to be placed. The concern related to this object is its integrity. In case this number is overwritten maliciously, the organization would face problems. They may either run out of materials before planned or they would be paying for a large unnecessary inventory. The company estimates that having a large inventory would cost them around $500 and an insufficient inventory $2,000, for a total cost of $2,500. The probability of those events is 0.1. Therefore, the total risk of* p_2 *is $250. Permission* p_3 *allows halting the machines that produce soaps. If this permission is maliciously used, the entire factory would be stopped and serious consequences may occur. The organization may not be able to fulfill its contracts, may lose money, and in the worst case, clients. The cost of this event is estimated to be $20,000. However, in order to use it, three administrators need to authorize the operation. Hence, the probability of this misuse is estimated to be very low: 0.005, for a total risk of $100.*

4.3.2 Risk Associated with Role Sets

Intuitively, the risk associated with a set of roles is a function of the risk of the permissions that can be accessed through those roles. When calculating such risk values, we include the risk of the permissions that can be explicitly acquired through those roles, as well as those that can be inferred from them. We first show how we model the inference problem, and then we present how to calculate the risk of activating a set of roles.

4.3.2.1 Inference Threat and Activation History An *Inference threat* exists when a user is able to infer unauthorized sensitive information through what seems to be innocuous data he is authorized for.

Definition 2. *An Inference Tuple $\langle PS, p_x \rangle$ consists of a set of permissions $PS \subset P$, and an inferred permission $p_x = \langle x, read \rangle \in P$, for which the following conditions hold:*

1. *PS does not contain the inferred permission: $p_x \notin PS$.*
2. *Once a user has acquired all the permissions in PS, he has all the information required to infer object x.*
3. *The set PS is a minimal set of permissions that allows the inference of object x.*

We denote the set of all inference tuples by $\mathcal{I}$.

Note that in the above definition, an object may be inferred through more than one set of permissions. For instance, it may be possible to infer object x through two different sets of permissions PS_1 and PS_2; which results in two inference tuples: $\langle PS_1, p_x \rangle$ and $\langle PS_2, p_x \rangle$. It is also possible that the same set of permissions can be used to infer different objects, e.g,. $\langle PS_3, p_i \rangle$ and $\langle PS_3, p_j \rangle$. These inference tuples can be found automatically using techniques such as those described in [33, 118].

To determine when a user has acquired all the permissions necessary to be able to infer unauthorized information, we need to keep track of his access history because the user may accumulate over time information to perform the inference. We denote the history of access of user u as $\mathbb{H}_u$ and use two functions. Function $permInferred(\mathbb{H}_u)$ returns the set of permissions the user would be able to infer given his previous accesses, and function $permExercised(\mathbb{H}_u)$ returns the set of permissions the user has exercised.

Definition 3. *Given a user u, his history of access $\mathbb{H}_u$ and a set of roles R' that he is authorized for, function* inferred$(\mathbb{H}_u, u, R')$ *returns the set of unauthorized permissions that u can infer uniquely after activating R':*

$$\begin{aligned} inferred(\mathbb{H}_u, u, R') = \{p_x \quad | \quad & \langle PS, p_x \rangle \in \mathcal{I} \ \wedge \ p_x \notin authorized(u) \\ & \wedge \ permExercised(\mathbb{H}_u) \cup P_{au}(R') \supseteq PS \\ & \wedge \ p_x \notin permInferred(\mathbb{H}_u) \ \} \end{aligned}$$

Note that in the above definition, because of the last condition, only permissions that the user would not be able to infer without activating R' are included. In section 4.5, we present in detail a methodology to find $inferred(\mathbb{H}_u, u, R')$ using a CP-net.

4.3.2.2 Calculating The Role Set Risk

The risk exposure of providing access to a set of roles $R' \subseteq R$ to a user u depends on the state of the CP-net. The following formula provides the risk exposure.

Definition 4. *The risk exposure of the system if user u activates a set of roles $R' \subseteq R$ in context c is given by*

$$rs(R', c, u) = \sum_{p \in \wp} rs(p, c)$$

where $\wp = P_{au}(R') \cup inferred(\mathbb{H}_u, u, R')$.

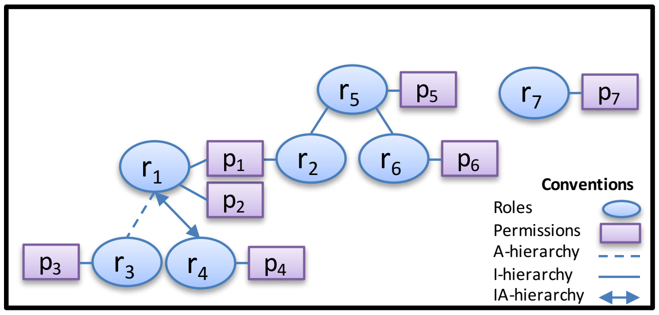
When no inference occurs due to the activation of R', $inferred(\mathbb{H}_u, u, R') = \emptyset$; and the risk exposure is given by the risk of the authorized permissions $P_{au}(R')$. On the contrary, when one or more roles in R' allow the user to infer unauthorized information, the risk includes the risk of directly acquired permissions and the risk of the inferred permissions in $inferred(\mathbb{H}_u, u, R')$. In Example 3, Section 4.5.1, we show how $inferred(\mathbb{H}_u, u, R')$ is computed and used.

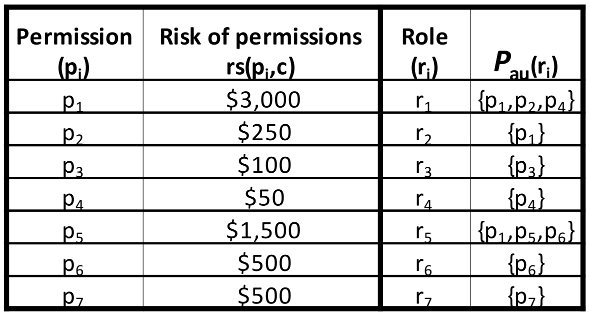
4.3.3 Trust Thresholds Associated with Role Sets

The trust threshold associated with a set of roles represents how trusted a user needs to be in order to use those roles. Intuitively, this threshold needs to reflect the risk exposure of the organization when the roles are activated by a user. We define the trust threshold as follows.

Definition 5. *The trust threshold of the set of roles $R' \subseteq R$, in context c for user u is defined as follows:*

$$\tau(R', c, u) = \frac{rs(R', c, u)}{\sum_{p \in P} (rs(p, c))}$$





Permission (p_i)	Risk of permissions $rs(p_i,c)$	Role (r_i)	$P_{au}(r_i)$
p_1	\$3,000	r_1	$\{p_1,p_2,p_4\}$
p_2	\$250	r_2	$\{p_1\}$
p_3	\$100	r_3	$\{p_3\}$
p_4	\$50	r_4	$\{p_4\}$
p_5	\$1,500	r_5	$\{p_1,p_5,p_6\}$
p_6	\$500	r_6	$\{p_6\}$
p_7	\$500	r_7	$\{p_7\}$

Figure 4: Policy for example 2.

Where $0 \leq \tau(R', c, u) \leq 1$. When $\tau(R', c, u) = 0$, it means that user u does not need to be trusted to activate R' in context c; when $\tau(R', c, u) = 1$, it means that user u needs to be completely trusted in order to activate R' in context c.

Example 2. *Consider the policy presented in Figure 4 where roles are represented by circles and permissions by rectangles. Roles are organized hierarchically as shown in the figure. In the table, the risk of each permission and the set of authorized permissions per role are listed. Let* $inferred(\mathbb{H}_u, u, R') = \emptyset$ *and assume that user* u *is assigned to roles* r_1, r_5 *and* r_6. *(a) Suppose that* u *wants to activate role set* $\{r_1, r_6\}$ *under context* c. *Given that* $P_{au}(\{r_1, r_6\}) = \{p_1, p_2, p_4, p_6\}$ *and according to Definition 4, the risk exposure is equal to* $rs(\{r_1, r_6\}, c, u) = rs(p_1, c)+rs(p_2, c)+rs(p_4, c)+rs(p_6, c)$. *Hence,* $rs(\{r_1, r_6\}), c, u) = \$3,800$ *and* $\tau(\{r_1, r_6\}, c, u)$ *= \$3,800/\$5,900 = 0.64, where \$5,900 corresponds to the sum of the risk of all the permissions. (b) Now suppose that* u *requests the activation of* r_5 *under context* c. *Here* $P_{au}(r_5) = \{p_1, p_5, p_6\}$, $rs(\{r_5\}), c, u) = \{rs(p_1, c) + rs(p_5, c) + rs(p_6, c)\}$ *= \$5,000 and* $\tau(\{r_5\}, c, u)$*=0.85. Role* $\{r_5\}$ *grants access to permissions that are very critical, more than the ones granted by* $\{r_1, r_6\}$*; therefore its trust threshold is higher.*

4.3.4 Trust of Users

We assign each user in the system a trust level. The trust for a user u in context c is denoted by $trust(u, c)$ and is defined in the interval $[0, 1]$, where 1 means the user is fully trusted and

0 means the user is totally untrusted. The *Trust and Contexts Module* in Figure 3 considers the behavior of users over time and the context to calculate the trust value for each user; e.g., if the user is using an untrusted connection, the trust in the user may be reduced. The details of this process are out of the scope of this dissertation. Solutions such as [21, 37] can be used to construct profiles and latter calculate a trust value based on the behavior of a user.

4.4 MINIMIZING THE RISK EXPOSURE

To make sure that our system enforces a policy correctly, we provide the definition of a *well-formed policy* that establishes a baseline of the types of accepted policies.

Definition 6. *A well-formed policy is defined as follows:*

1. *No roles in a DSoD constraint are allowed to have any I or IA-seniors:*
   $\forall\ r \in R, dsod(RS, k) \in DSoD\ \nexists r' \in R :\ (r' \geq_I r)\ \vee (r' \geq_{IA} r)$
2. *User to role assignments should respect SSoD:*
   $\forall u \in U, ssod(RS, k) \in SSoD : |authorized(u) \cap RS| < k$
3. *User to role assignments should respect the cardinality constraints:*
   $\forall u \in U, card_A(r, k) \in CARD_A : |authorized(u) \cap RS| < k$

Condition 1 states that the roles involved in a DSoD constraint may only have A-senior roles. As explained in [101], this condition allows the system to enforce DSoD constraints. Condition 2 establishes that all SSoD constraints are enforced in presence of hybrid hierarchy. Finally, condition 3 ensures that the user assignment fulfills the cardinality constraints. When a policy fulfills all these conditions, it is possible to enforce it during runtime.

4.4.1 Trust-and-Risk Aware Role Activation

The role activation process is instrumental in our framework. It is in charge of identifying when a user should be denied to activate roles due to lack of trust or other policy constraints.

It also allows us to minimize the risk exposure by selecting the roles that have less risk in the system. First, we present the problem statement.

Problem Statement: A user u in context c with a trust value $trust(u, c)$ requests the system to activate a permissions set $PS \subseteq P$ in a single session. The system responds to the user's request by either accepting it and determining the proper roles to be activated or rejecting it. If the access is granted, the roles selected to be activated should *minimize the risk* exposure of the organization.

A request of a user $u \in U$ in context c for permissions $PS \subseteq P$ is granted if a set of roles $R_q \subseteq R$ can provide the permissions in PS, and the following conditions hold: (1) The user is authorized for all the roles in R_q. (2) The user's trust level ($trust(u, c)$) is greater or equal to the trust threshold of the set of roles R_q. (3) The DSoD and cardinality constraints are not violated when roles in R_q are activated simultaneously. The following optimization problem captures the Trust-and-Risk Role Activation problem.

Definition 7. *The Trust-and-Risk Aware Role Activation Optimization Problem for a query $q = \langle u, PS, c\rangle$, consists of finding a solution, R_q, such that:*

$$\min_{R_q \subseteq authorized(u)} \quad rs(R_q, c, u)$$

s.t.

$$\begin{aligned}
\forall\ dsod(RS_i, k_i) \in DSoD\ &: |R_q \cap RS_i| < k_i && (a)\\
\forall\ card(r_c, k) \in CARD \wedge r_c \in R_q &: activated(r_c) + 1 \leq k - 1 && (b)\\
trust(u, c) &\geq \tau(R_q, c, u) && (c)\\
P_{au}(R_q) &\supseteq PS && (d)
\end{aligned}$$

The system grants a request only if the entire set of requested permissions can be authorized to the user, as we assume that the permissions in PS need to be used simultaneously. In addition, we only require that $P_{au}(R_q) \supseteq PS$. This means that the selected roles may provide additional permissions than those requested by the user. We argue that selecting the roles that minimize the risk is better than providing the roles that minimize the number of extra permissions. To see why, let us consider two possible solutions. The first solution contains one role that provides one additional permission, with a risk of \$10, whereas the second solution contains a role that provides two extra permissions with a risk of \$1. In this case, the algorithm selects the second solution because, even though the number of additional permissions is higher, the total risk exposure is reduced.

4.4.2 Role Activation Algorithm

We propose Algorithm 1 to find a solution for the Trust-and-Risk Aware Role Activation Problem. Our algorithm assumes that the policy is well formed, as per Definition 6. The algorithm first removes from the search space the roles that cannot be activated due to trust issues (line 4). The current best solution is stored in the set R_q, which initially is empty. The function $selectRolesMinimumRisk(P_{rem},\ R_{avail},\ R_{sel},\ u, c)$ finds candidate solutions, and compares them to select the best one. This function is recursive and it starts by checking the base case. This occurs when a candidate solution provides all the permissions requested (line 16). If the candidate solution is less risky than the current best solution, it becomes the new best solution. If both solutions have the same risk, the algorithm selects the one that has lesser number of roles. Otherwise, it keeps the original solution. Before the algorithm reaches the base case, it prunes the search space by removing the roles that cannot be activated due to DSoD and cardinality constraints (lines 25 and 28). Roles that do not provide the missing permissions in the candidate solution are also removed (line 32). These pruning steps take place before any role is added to a candidate solution, ensuring that candidate solutions do not contain roles that violate the constraints of the policy. In case no candidate roles are left after the pruning (line 34), the algorithm backtracks as that search path did not lead to a valid solution. Otherwise, the next role to be added to the candidate set is chosen in line 36; this function only selects a role r if adding it to R_{sel} fulfills $\tau(R_{sel} \cup \{r\}, c, u) \leq trust(u, c)$ (line 44). We evaluate two heuristics to perform this step in Section 4.6. The selected role is denoted as r_{best}. The algorithm evaluates the two possible paths *i)* a candidate solution where r_{best} is added (line 40) and *ii)* a candidate solution that does not contain r_{best} (line 41).

Algorithm 1 Trust-and-Risk Aware Role Activation

Precondition: The policy is well-formed (Definition 6).
Postcondition: R_q contains the solution of the problem specified in Definition 7.

```
 1: findTrustAndRiskAwareActivationSet($u, PS, c$)
 2: $R_{avail} \leftarrow authorized(u)$ {Candidate roles}
 3: for all $r \in R_{avail}$ do
 4:   if $\tau(r, c, u) > trust(u, c)$ then
 5:     $R_{avail} \leftarrow R_{avail} \setminus r$ {Pruning based on user's trust}
 6: $R_{sel} \leftarrow \emptyset$ {Selected roles so far}
 7: $P_{rem} \leftarrow PS$ {Set of permissions that haven't been found}
 8: $R_q \leftarrow \emptyset$ {Global variable, stores the best found solution}
 9: $selectRolesMinimumRisk(P_{rem}, R_{avail}, R_{sel}, u, c)$
10: if $R_q \neq \emptyset$ then
11:   return $R_q$ {Request accepted, activate $R_q$}
12: else
13:   return $\emptyset$ {Request denied}
14: ------------------------------------------------------------

15: selectRolesMinimumRisk($P_{rem}, R_{avail}, R_{sel}, u, c$)
16: if $P_{rem} = \emptyset$ then
17:   if $R_q = \emptyset$ then
18:     $R_q \leftarrow R_{sel}$
19:   else
20:     if $rs(R_q, c, u) > rs(R_{sel,c,u})$ then
21:       $R_q \leftarrow R_{sel}$
22:     else if $(rs(R_q, c, u) = rs(R_{sel}, c, u)) \wedge \mid R_q \mid > \mid R_{sel} \mid$ then
23:       $R_q \leftarrow R_{sel}$
24:   return {Found candidate solution}
25: for all $dsod(RS, k) \in DSoD$ do
26:   if $\mid R_{sel} \cap RS \mid = (k - 1)$ then
27:     $R_{avail} \leftarrow R_{avail} \setminus [RS \setminus (R_{sel} \cap RS)]$
28: for all $card(r_c, k) \in CARD \wedge r_c \in R_{avail}$ do
29:   if $activated(r_c) + 1 = k - 1$ then
30:     $R_{avail} \leftarrow R_{avail} \setminus r_c$
31: for all $r_i \in R_{avail}$ do
32:   if $P_{rem} \cap P_{au}(r_i) = \emptyset$ then
33:     $R_{avail} \leftarrow R_{avail} \setminus r_i$
34: if $R_{avail} = \emptyset$ then
35:   return
36: $r_{best} \leftarrow nextRole(P_{rem}, R_{avail}, \mathcal{H}, u, R_{sel})$
37: if $r_{best} = \perp$ then
38:   return
39: $R_{avail} \leftarrow R_{avail} \setminus r_{best}$
40: selectRolesMinimumRisk($P_{rem} \setminus P_{au}(r_{best}),\ R_{avail},\ (R_{sel} \cup \{r_{best}\}), u, c$)
41: selectRolesMinimumRisk($P_{rem}, R_{avail}, R_{sel}, u, c$)
42: ------------------------------------------------------------
43: nextRole($P_{rem}, R_{avail}, \mathcal{H}, u, R_{sel}$)
44: select $r \in R_{avail}$ such that $\tau(R_{sel} \cup \{r\}, c, u) \leq trust(u, c)${We evaluate selection heuristics in
    Section 4.6}
45: return if found $r$ otherwise return $\perp$
```

4.4.2.1 Proof of Correctness of the Algorithm We now prove that Algorithm 1 is correct with respect to Definition 7. Algorithm 1 has the following pre and post conditions. *Precondition:* The policy is well-formed as per Definition 6. *Postcondition:* If $R_q = \emptyset$, no solution was found otherwise, R_q solves the problem specified in Definition 7.

Theorem 1. *(Correctness of Algorithm 1) Given an authorization query* $q = \langle u, PS, c\rangle$ *and a well-formed policy PL, as per Definition 6, Algorithm 1 finds* R_q *with the minimum risk value,* $rs(R_q, c, u)$ *that satisfies the problem specified in Definition 7.*

Proof. Note that the postcondition of the algorithm is fulfilled if the set of roles R_q returned by function *selectRolesMinimumRisk* constructs the set appropriately. Hence, we focus on that function. Let us begin by presenting the invariants of Algorithm 1.

- *Invariant 1:* At any time during the execution of the algorithm, R_q satisfies all the constraints (a), (b), (c) and (d) specified by Definition 7.
- *Invariant 2:* At any time during the execution of the algorithm, R_q contains the best solution explored so far (less risky), as specified by Definition 7.

We divide the proof into three parts for clarity. We first show that the solution found respects the constraints of the policy, that is, invariant 1 is fulfilled. Then, we prove invariant 2 and show that the algorithm terminates. Finally, we prove that the algorithm always finds the best solution.

1) Invariant 1 is fulfilled: R_q is only updated in lines 18, 21 and 23, and in each case, R_q is assigned roles that are in R_{sel}. Thus, for R_q to fulfill the invariant, R_{sel} also needs to fulfill the constraints of Definition 7. We prove that this is the case by induction.

Base case: The first time the function is called in line 9, $R_{sel} = \emptyset$. It is clear that constraints (a), (b) and (d) are respected. Since $\tau(\emptyset, c, u) = 0$, constraint (c) is also respected.

Induction case: Assume that R_{sel} received as parameter by the function respects invariant 1. Let R'_{sel} be the parameter used by the function when it invokes itself recursively in lines 40 and 41; we now show that R'_{sel} also fulfills the invariant. The invocation in line 41 trivially fulfills the invariant as $R'_{sel} = R_{sel}$. The invocation in line 40 contains $R'_{sel} = R_{sel} \cup r_{best}$. Constraint (c) is trivially fulfilled as in line 44, r_{best} is selected explicitly to fulfill this condition. Note that the roles from which r_{best} has been selected have been pruned to avoid violating constraints (a), (b) and (d). After line 27, $\forall\ dsod(RS, t) \in DSoDR_{sel}, r \in R_{avail}$: $\nexists\ R_{sel} : |R_{sel} \cup \{r\} \cap RS| = k - 1$. Therefore, $R_{sel} \cup r_{best}$ respects constraint (a). Similarly, in line 30 the roles of R_{avail} are pruned so that adding one to R_{sel} does not violate constraint (b). After this pruning $\forall card(r_c, k) \in CARD \wedge r_c \in R_{avail} : activated(r_c) + 1 \geq k - 1$. Finally, in line 33 the roles that do not contribute to the coverage of permissions are pruned so that $\forall r \in R_{avail} : P_{au}(r) \cap P_{rem} \neq \emptyset$. Hence, in line 36 no matter what role is selected to be added to R_{sel}, it is sure that it will not violate invariant 1.

2)Invariant 2 is fulfilled: R_q is only updated when: (i) no solution has been found up to that point (line 18), in this case, a solution is better than no solution and hence the invariant is fulfilled, (ii) the risk of the candidate solution is smaller (line 21) or (iii) a solution with the same risk, but less number of roles is found (line 23). It is clear that invariant 2 is always fulfilled in these three cases.

3) The algorithm always terminates:

Algorithm 1 terminates if the recursive function *selectRolesMinimumRisk* terminates. The function *selectRolesMinimumRisk* returns when (i) a solution has been found (line 24), (ii) the algorithm backtracks because there are no roles in R_{avail} (line 35) or because the inference risk is too high (line 38).

In the worst case, no solution is found and hence, the algorithm terminates when $R_{avail} = \emptyset$. The first time the function *selectRolesMinimumRisk* is invoked, R_{avail} contains the roles user u is authorized for (line 2) minus the ones he cannot activate because they require a larger trust value (line 4). In the worst case, R_{avail} contains all the roles authorized for u when the recursive function *selectRolesMinimumRisk* is called.

The algorithm terminates because:

- The set R_{avail} is bounded by the total amount of roles existing in the policy R.
- No element is ever added to R_{avail}.
- Every time the function is invoked recursively, at least one role is removed from R_{avail} with which it was invoked.

R_{avail} is reduced in function *selectRolesMinimumRisk* in lines 25, 32 and 39. In the worst case, no role is removed from R_{avail} due to policy constraints or already covered permissions (lines 25, 28 and 32). However, in line 39 there is always a *deterministic* reduction of the set R_{avail} in which a role is always removed from R_{avail} and added to the solution. For this reason, the next time the function is called in lines 40 and 41, R_{avail} always contains one less role. Therefore, the algorithm terminates.

4) The algorithm always finds the set of roles with minimum risk exposure: Since invariants 1 and 2 hold, it suffices to prove that the algorithm explores all the possible *valid* solutions. In line 40), r_{best} is selected. The algorithm follows a depth-first strategy: it first explores the solution where r_{best} is added and later where it is not (41). Thus, all valid solutions are explored, and hence, the algorithm always finds the best solution. □

The previous proof demonstrates the correctness of Algorithm 1. In Section 4.6, we experimentally evaluate its performance. Before doing so, in the following section, we present in detail the inference threat analysis and administration module.

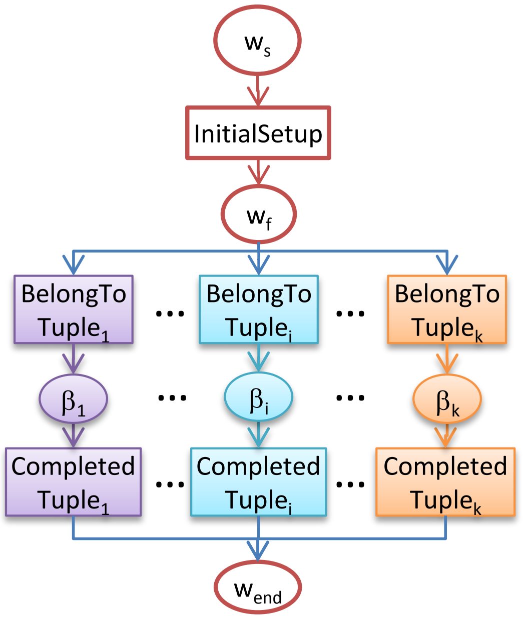
4.5 INFERENCE THREAT ANALYSIS AND ADMINISTRATION

In this section, we present a CP-net based methodology to find the information a user may infer after a particular access, and a technique to manage inference threats associated with a particular policy. We begin by presenting the CP-net we propose to find an access inferred permissions as specified in Definition 3. Then we present a simulation methodology to find *active inference threats*. We show how the simulation results can be used to refine the CP-net reducing its complexity. Finally, we present a methodology that allows an administrator to manage the risk of active inference threats.

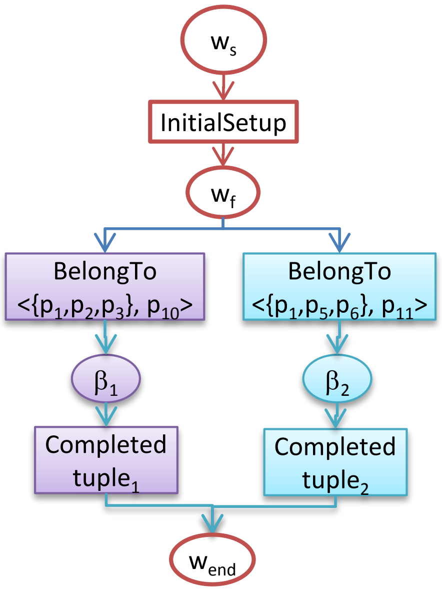
4.5.1 Finding Inferred Permissions

In Definition 2, the set of inference tuples, $\mathcal{I}$, that are applicable to an organization were defined. We begin by specifying the terminology we use to refer to the components of $\mathcal{I}$.

Definition 8. *Given a set of inference tuples $\mathcal{I}$, we define the set of* risk inference objects *$\mathcal{O}_{\mathcal{I}}$, and the set of* inference permissions *$\mathcal{P}_{\mathcal{I}}$ as follows:*



(a) General structure of the CP-net, as per Definition 9.



(b) Example 3.

Figure 5: CP-net graphical representation.

$$\mathcal{O}_{\mathcal{I}} = \{o \mid \langle PS, \langle o, r \rangle \rangle \in \mathcal{I}\} \quad \textit{and} \quad \mathcal{P}_{\mathcal{I}} = \bigcup_{\langle PS,p \rangle \in \mathcal{I}} PS$$

To identify if a user has obtained all information needed to infer a particular object, we model the role activation history using a CP-net. The inference tuples in $\mathcal{I}$ determine the specific structure of the CP-net. The general structure of the proposed CP-net is presented in Figure 5a.

In the following discussion, we assume that each inference tuple in $\mathcal{I}$ has been enumerated from 1 to k. That is, there are k inference tuples in the system, and $\langle PS_i, p_{xi} \rangle_i$ refers to the i^{th} inference tuple. For each inference tuple $\langle PS_i, p_{xi} \rangle_i$, two transitions $BelongToTuple_i$ and $CompletedTuple_i$, and a place β_i are created. Each user has tokens positioned in different places of the CP-net; the placement of tokens reflects the access history of each user. We use function $tokensAt(u \in U, w \in W)$ to retrieve the set of tokens of user u at place w. In what follows, we formally define the CP-net and then explain how it works and show an example.

Definition 9. *Given a policy a* $PL = \langle R, U, P, \mathcal{I}, C, DSoD, SSoD\rangle$, *we define an Inference CP-net as a tuple* $\mathcal{H} = \langle PL, W, T, F, V, \Upsilon, \lambda, m_o\rangle$ *where:*

1. ***Places (W):*** *For each* $\langle PS_i, x_i\rangle_i \in \mathcal{I}$, *a place* β_i *is created. Let* $B = \{\beta_1, ..., \beta_k\}$, *then:* $W = \{w_s, w_f, w_{end}\} \cup B$.
2. ***Transitions (T):*** *For each* $\langle PS_i, x_{pi}\rangle_i \in \mathcal{I}$, *a pair of transitions* $BelongToTuple_i$ *and* $CompletedTuple_i$ *are created. Let*
   $D_1 = \{BelongToTuple_1, ..., BelongToTuple_k\}$,
   $D_2 = \{CompletedTuple_1, ..., CompletedTuple_k\}$. *Then:*
   $T = \{InitialSetup\} \cup D_1, \cup D_2$
3. ***Arcs (F):*** *Let* $E = \{\langle BelongToTuple_i, \beta_i\rangle : \forall i\ 1 \leq i \leq k\} \cup \{\langle \beta_i,\ CompletedTuple_i\rangle : \forall i\ 1 \leq i \leq k\}$. *Then,*
   $F = \{< w_s,\ InitialSetup>\},\ <InitialSetup,\ w_f > \cup \{w_f\} \times D_1 \cup\ E \cup D_2 \times \{w_{end}\}$
4. ***Token Types (V):*** *Let* $u \in U$, $R' \subseteq R$, $P_{R',\mathcal{I}} \subseteq \mathcal{P_I}$, *and* $p_x \in \mathcal{P_I}$, *we have:*
   $V = \{\langle R', u\rangle, \langle P_{R',\mathcal{I}}, R', u\rangle, \langle u, p_x\rangle\}$.
5. ***Accepted Types of Tokens (Υ):*** $\Upsilon(w_s) = \langle R', u\rangle$,
   $\Upsilon(w_f) = \Upsilon(\beta_i) = \langle P_{R',\mathcal{I}}, R', u\rangle$, *for all* $1 \leq i \leq k$ *and*
   $\Upsilon(w_{end}) = \langle u, p_x\rangle$.
6. ***Firing rules*** *(G and λ):*
   InitialSetup: Given token $\langle R', u\rangle$ *placed at* w_s*:*
   $G(InitialSetup, \langle R', u\rangle) = true$
   $\lambda(InitialSetup, \langle R', u\rangle) = \langle P_{R',\mathcal{I}}, R', u\rangle$,
   where $P_{R',\mathcal{I}} = P_{au}(R') \cap \mathcal{P_I}$

   $BelongToTuple_i$ $(1 \leq i \leq k)$: Given token $\langle P_{R',\mathcal{I}}, R', u\rangle$ *placed at* w_f, *and tuple* $\langle PS_i, p_{xi}\rangle_i$*:*
   $G(BelongToTuple_i, \langle P_{R',\mathcal{I}}, R', u\rangle) = [(PS_i \cap P_{R',\mathcal{I}}) \neq \emptyset \wedge p_{xi} \notin P_{au}(authorized(u))]$
   $\lambda(BelongToTuple_i, \langle P_{R',\mathcal{I}}, R', u\rangle) = \langle P_{R'',\mathcal{I}}, R'', u\rangle$ *where* $P_{R'',\mathcal{I}} = P_{au}(R') \cap PS_i$ *and* $R'' = \{r | r \in R' \wedge P_{au}(r) \cap PS_i \neq \emptyset\}$

   $CompletedTuple_i$ $(1 \leq i \leq k)$: Given a set of tokens $V' = tokensAt(u, \beta_i)$ *of type* $\langle P_{R',\mathcal{I}}, R', u\rangle$*:*
   $$G(CompletedTuple_i, V') = [\bigcup_{\langle P_{R',\mathcal{I}}, R', u\rangle \in V'} P_{R',\mathcal{I}}] = PS_i$$
   $\lambda(CompletedTuple_i, V') = \langle u, p_{xi}\rangle$, *where* p_{xi} *is the inferred permission of tuple i.*
7. *Initial State* (m_0)*: Initially, no tokens have been placed.*

The CP-net works as follows. When a user u initially tries to activate a set of roles $R' \subseteq R$ for which he is authorized, a token $\langle R', u\rangle$ is placed in w_s. Then, transition *InitialSetup* fires, consuming the token in w_s and placing a token of a different color at w_f. Changing colors enables us to keep track of relevant attributes. In this case, it is important to know which of the permissions acquired through R' would allow an inference. We denote this set of permissions as $P_{R',\mathcal{I}} = P_{au}(R') \cap \mathcal{P_I}$. Hence, when *InitialSetup* fires, token $\langle P_{R',\mathcal{I}}, R', u\rangle$ is placed at w_f. Tokens placed at w_f are evaluated in parallel by the *BelongToTuple* transitions.

A transition $BelongToTuple_i$ fires when at least one of the permissions in $P_{R',\mathcal{I}}$ belongs to the corresponding set of inference PS_i, and when the user cannot legitimately acquire p_{xi}. If the transition fires, the token at w_f is consumed and a token is placed at β_i. Note that it is possible that a token placed at w_f fires several transitions of the type $BelongToTuple_i$. If at some point of time, a place β_i contains all the tokens that for the same user complete the entire set of permissions PS_i required to infer object x_i, transition $CompletedTuple_i$ fires. In other words, $CompletedTuple_i$ is triggered when a user has acquired all the permissions in PS_i of inference tuple $\langle PS_i, p_{xi}\rangle_i$. Transition $CompletedTuple_i$ consumes *all* the tokens that show user u has acquired enough information to infer p_{xi}, and places token $\langle u, p_{xi}\rangle$ at w_{end}.

Hence, the history of accesses is provided by the places where the tokens are stored in the Inference CP-net; as roles are activated by users, the tokens move around the CP-net. Tokens placed in w_{end} represent information that the user may be able to infer given his previous accesses.

In Algorithm 2, we show the procedure to find the possible new permissions P' a user u may infer after activating a set of roles R' as per Definition 3. First, in line 2, the current inferred permissions of the user are saved in M. Thus, M contains the set of tokens inferred by user u before activating R'. Then, we place one token $\langle R', u\rangle$ at w_s. After the transitions fire, and all tokens are in a place different than w_s, we check the state of the CP-net. We denote this new state as $\mathcal{H}'$. Then, in line 5, we store in N all the tokens placed at w_{end}. Since we only need to identify the permissions that u will be able to infer if he activated R', in line 6, Q is initialized to contain only newly inferred information. Finally, in line 7, we extract from Q the set P' of newly inferred permissions.

We first prove that P' contains the newly inferred permissions as per Definition 3, and then we show an example of how the Inference CP-net works.

Theorem 2. *Given a user u, an Inference CP-net $\mathcal{H}$ that contains the history of access of user u, and a set of roles R' that user u can activate, Algorithm 2 finds the set of inferred permissions P', such that $P' = Inferred(\mathbb{H}_u, u, R')$ as per Definition 3.*

Proof. To prove that the set of permission P' returned by Algorithm 2 follows Definition 3, we need to ensure that the four conditions of that definition are fulfilled. The conditions (1)

Algorithm 2 Given an *Inference CP-net* $\mathcal{H}$, a user $u \in U$ and a set of roles R' that he can activate, return the set of inferred permission P', as per Definition 3.

```
1: findInferredPermissions($\mathcal{H}, u, R'$)
2: $M \leftarrow \mathcal{H}.tokensAt(u, w_{end})$ {Save old inferences}
3: $\mathcal{H}.place(\langle R', u\rangle, w_s)$
4: Wait for $\mathcal{H}$ to distribute the tokens. We referred to this new state as $\mathcal{H}'$.
5: $N \leftarrow \mathcal{H}'.tokensAt(u, w_{end})$
6: $Q \leftarrow$ Tokens in $N$ that are not contained in $M$.
7: $P' = \{p_x \mid < u, p_x > \in Q\}$
8: return $P'$
```

and (2) are trivially fulfilled as they are explicitly checked by the guards of the firing rules of type $BelongToTuple_i$ for $(1 \leq i \leq k)$. Because transitions of type $CompleteTuple_i$, for $(1 \leq i \leq k)$, are only fired when the entire set of permissions PS is acquired, tokens placed at w_{end} correspond to inferences that the users would be able to perform, hence condition (3) is fulfilled. Finally, condition (4) is also fulfilled because of the processing performed in line 6, where only newly inferred permissions are assigned to Q. Therefore, the set of inferred permissions P' returned by Algorithm 2 are equal to $P' = Inferred(\mathbb{H}_u, u, R')$ in Definition 3. □

Example 3. *Suppose $\mathcal{I} = \{\langle\{p_1, p_2, p_3\}, p_{10}\rangle, \langle\{p_1, p_5, p_6\}, p_{11}\rangle\}$. The corresponding CP-net is shown in Figure 5b; initially there are no tokens. User u_1 activates roles $R_1 = \{r_1, r_2\}$ for which $P_{au}(R_1) = \{p_1, p_2, p_8, p_9, p_{15}\}$. A token $v_1 = \langle u_1, \{r_1, r_2\}\rangle$ is placed at w_s. After transition InitialSetup fires, v_1 is consumed and a token $\langle u_1, \{p_1, p_2\}, \{r_1, r_2\}\rangle$ is placed at w_f. Since R_1 acquires p_1 which is part of both inference tuples and u_1 is not authorized for p_{10} or p_{11}, the token at w_f is removed, and two tokens are placed at β_1, and β_2. The tokens placed contain this information: $\langle u_1, \{p_1, p_2\}, \{r_1, r_2\}\rangle$. Since none of the inference tuples is completed by R_1, there are no new tokens at w_{end}, and $findInferredPermissions(\mathcal{H}, u_1, R_1)$ returns $\emptyset$. Thus, in context c, $rs(R_1, c, u_1) = rs(p_1, c) + rs(p_2, c) + rs(p_8, c) + rs(p_9, c) + rs(p_{15}, c)$, which does not contain any inferred risk. We denote the new state of the CP-net by $\mathcal{H}'$. After a while, assume u_1 activates role r_3, where $P_{au}(r_3) = \{p_3, p_4\}$. Token $\langle u_1, \{r_3\}\rangle$ is placed at w_s. Transition $BelongTo\langle\{p_1, p_2, p_3\}, p_{10}\rangle$ fires and token $\langle u_1, \{p_3\}, \{r_3\}\rangle$ is placed at β_1. At that point, transition $CompletedTuple_1$ fires because two tokens that belong to u_1, and* *complete the inference tuple are at* β_1. *This time* $findInferredPermissions(\mathcal{H}', u_1, \{r_3\})$ *returns* $\{p_{10}\}$. *Hence, the risk in context c of* $rs(r_3, c, u_1) = rs(p_3, c) + rs(p_4, c) + rs(p_{10}, c)$, *which includes the risk of the inferred permission* p_{10}.

4.5.2 Finding Active Inference Threats

In this section, we propose a methodology to improve the performance of the *Inference CP-net* presented in Definition 9. We assume that $\mathcal{I}$ contains all the existing inference threats. Note that the set of inference tuples $\mathcal{I}$ does not depend on the user-to-role or the permission-to-role assignments. Inference tuples are uniquely dependent on the types of objects that exist on the organization (this is the case for existing methodologies to find automatically inference tuples [33, 118]). Although it would be possible to include in $\mathcal{I}$ uniquely the *active* inference threats for a particular user-to-role and permission-to-role assignments, we argue that this would be undesirable. The reason is that if $\mathcal{I}$ contains all the existing inference tuples, even if the user-to-role and the permission-to-role assignments change, the framework can still capture the risk exposure due to inference threats. In contrast, if $\mathcal{I}$ only contains the tuples for a particular policy configuration, for each possible policy modification, the administrator would need to verify and possibly include or remove new tuples in the set $\mathcal{I}$.

Since the tuples in $\mathcal{I}$ are independent of the user-to-role and the permission-to-role assignments, there is some room for refinement during the deployment of the *Inference CP-net.* This refinement consists in finding the *active inference* threats for a given policy and uniquely including the relevant inferences tuples in the deployed *Inference CP-net.*

4.5.2.1 Simulating users' behavior to identify active inference threats

To find the active inference threats, we take advantage of the existing properties of CP-nets. In particular, using a CP-net we can simulate the behavior of the system to determine its properties and to understand how the system will behave in the long term (stable state); e.g., whether a place in the CP-net is unreachable for a particular set of tokens. The simulations consist of placing a set of tokens in the starting place of the CP-net and allowing the CP-net to distribute those tokens according to its transition rules. Depending on the input that is used, the results may differ and different conclusions may be drawn.

To determine if in the long term there are any users that will be able to infer unauthorized information, we modify the *Inference CP-net* to verify specific properties of the policy in the system. We want to identify whether a policy configuration will provide any user enough information to perform an inference attack. Our objective is also to identify the user-to-role assignments that create an inference threat so that an administrator can decide if it is necessary to modify the policy to mitigate this risk. The Inference CP-net in Definition 9 was not designed to maintain this information. For this reason, for the simulation purposes, we create a similar CP-net that additionally stores the set of roles that led to the inference. The simulation CP-net is defined as follows.

Definition 10. *Given a policy* $PL = \langle R, U, P, \mathcal{I}, C, DSoD, SSoD \rangle$, *we define a* Simulation CP-net *as tuple* $\mathcal{H}_s = \langle PL, W, T, F, V, \Upsilon, \lambda, m_o \rangle$ *where:*

1. *Places* W*, transitions* T *and arcs* F *are defined as in the Inference CP-net in Definition 9.*
2. ***Token Types*** *(*V*): Let* $u \in U$, $R' \subseteq R$, $P_{R',\mathcal{I}} \subseteq \mathcal{P}_{\mathcal{I}}$, *and* $p_x \in \mathcal{P}_{\mathcal{I}}$, *we have:* $V = \{\langle R', u \rangle, \langle P_{R',\mathcal{I}}, R', u \rangle, \langle R', u, p_x \rangle\}$.
3. ***Accepted Types of Tokens*** *(*Υ*):* $\Upsilon(w_{end}) = \langle R', u, p_x \rangle$, *while* $\Upsilon(w_s)$, $\Upsilon(w_f)$ *and* $\Upsilon(\beta_i)$ *are defined as in Definition 9.*
4. ***Firing rules*** *(*G *and* λ*):*
   $\underline{InitialSetup}$ and $\underline{BelongToTuple_i}$ for $1 \le i \le k$ *are defined as per Definition 9.*

   $\underline{CompletedTuple_i\ (1 \le i \le k)}$*: Given a set of tokens* $V' = tokensAt(u, \beta_i)$ *of type* $\langle P_{R'',\mathcal{I}}, R'', u \rangle$*:*
   $$G(CompletedTuple_i, V') = [\bigcup_{\langle P_{R'',\mathcal{I}}, R'', u \rangle \in V'} P_{R'',\mathcal{I}}] = PS_i$$
   $\lambda(CompletedTuple_i, V') = \langle \mathcal{R}, u, p_{xi} \rangle$,
   where $\mathcal{R} = \{r \mid \langle P_{R'',\mathcal{I}}, R'', u \rangle \in V \wedge r \in R''\}$ *and* p_{xi} *is the inferred permission of tuple* i.

The main difference between the *Simulation CP-net* (Definition 10) and the *Inference CP-net* (Definition 9) is the amount of information they store. In Definition 10, the last place w_{end} stores the roles responsible for the inference of a permission. For this purpose, transition rules of type $CompletedTuple_i$ for $1 \le i \le k$ are also redefined to create a token of type $\langle \mathcal{R}, u, p_{xi} \rangle$. This token contains the set of roles $\mathcal{R}$ that is responsible for the possible inference of permission p_{xi} by user u.

The complete process to identify active inference threats is shown in Algorithm 3. Its input is the policy that is going to be tested, PL, and its output is the list of users that are able to infer unauthorized information and the user-to-role assignments that allow the inference. This list is stored in a variable $lstActive$ which contains tuples that show the inference $\langle \mathcal{R}, u, p_x \rangle$, where u is the user that infers permission p_x through role set $\mathcal{R}$. This list is initialized in line 2. The set of active inference tuples that we denote as $\mathcal{I}_y$ is initialized as an empty set in line 3. Then, a simulation CP-net $\mathcal{H}_s$ is generated according to Definition 10. Because several users may have exactly the same roles assigned to them, which we call having the same profile, we can perform the analysis only once for each profile. In line 5, we create a *representative user* for each profile. We assume that each representative user will activate at some point of time all the roles that he is authorized for. For this reason, in line 10 for each user $u \in U_p$, we generate a set of tokens that aim at simulating the behavior of the user throughout the life of the system. For each of the authorized roles a token is created in line 10 and added to the list of tokens of that user. Then, the simulation is run for several combinations of roles. In line 12 all the possible permutations of the way in which roles can be activated are found. Later, $\mathcal{H}_s$ is inspected to see whether the system allowed any inferences. For this purpose, in line 15, we verify if each user was able to infer information, and if so, we store the inference in the variable $lstActive$. Finally, to know which is the inference tuple that was activated, in line 18 we use the method *identifyInferencePath()* that identifies the path through which a token arrived to w_{end}. Then, the tuple identified is added to $\mathcal{I}_y$. Several CP-net simulations tools exist; we used CNP tools [42] to perform the simulations.

4.5.2.2 Refinement of the inference CP-net

We use the results of the simulation (Algorithm 3) to improve the performance of the system. Knowing which of the users may have in fact the power to infer information before the system is deployed, allows us to maintain uniquely the information of the Inference CP-net for the relevant users. This may reduce the amount of data that is actually stored by the system. Additionally, the simulation may reveal that not all of the tuples in $\mathcal{I}$ are in fact a feasible threat given a policy configuration. These may also be removed from the system to reduce the time required to calculate the risk exposure of activating a set of roles, and hence, minimize the time it takes to answer an access control request.

Algorithm 3 Find active inference threats given a policy configuration $PL = \langle R, U, P, \mathcal{I}, C, DSoD, SSoD \rangle$

```
1: findActiveInferenceThreats($PL$)
2: $lstActive \leftarrow \perp$ {List with active inferences, initially empty. Each element in this list is a tuple that represents an active inference $\langle \mathcal{R}, u, p_x \rangle$ where $u$ is the user that infers permission $p_x$ through role set $\mathcal{R}$}
3: $\mathcal{I}_y \leftarrow \emptyset$ {Initialize active inference tuples as an empty set.}
4: $\mathcal{H}_s$ = createCPNet($PL$) {According to Definition 10}
5: $U_P = sameProfile(U)$
6: for all $u \in U_p$ do
7:   $R_u = authorized(u)$
8:   $lstTokens_u \leftarrow \perp$
9:   for all $r \in R_u$ do
10:    $t$=createToken($\langle r, u \rangle$)
11:    $lstTokens_u$.add(t)
12:  for all $\Gamma \in$ nextPermutation($lstTokens$) do
13:    $\mathcal{H}_s$.placeAt($w_s, \Gamma$) {Place each token $t \in \Gamma$ at $w_s$ in $\mathcal{H}_s$}
14:    runSimulation()
15:    if $\mathcal{H}_s.tokensAt(u, w_{end}) \neq \emptyset$ then
16:      for all $t = \langle \mathcal{R}, u, p_x \rangle \in \mathcal{H}_s.tokensAt(u, w_{end})$ do
17:        $lstActive$.add($t$)
18:        $I = identifyInferencePath()$
19:        add $I$ to the set of active inference tuples $\mathcal{I}_y$
20: return $\langle lstActive, \mathcal{I}_y \rangle$
```

Let PL denote a policy that has $k = |\mathcal{I}|$ inference tuples. After Algorithm 3 is run for PL, no user was able to infer the permission associated with the inference tuples in $\mathcal{I}_n = \mathcal{I} \setminus \mathcal{I}_y$. As a consequence, it is not necessary to consider the inference tuples in $\mathcal{I}_n$, and the system uniquely has $q = |\mathcal{I}_y|$ active inference tuples for the current policy configuration PL. The CP-net constructed for the production system (Definition 9) needs only to contain the inference tuples $\mathcal{I}_y$. Additionally, we can reduce the storage required by not maintaining information of the users, U_n that cannot infer information under the policy PL. The set of users U_n is defined as follows.

Definition 11. *Given a policy PL and the list of tokens $lstActive$ created by Algorithm 3 for that policy, the set of users that cannot infer information under PL is computed as follows:*

$$U_n = U \setminus \{u \mid \langle \mathcal{R}, u, p_x \rangle \in lstActive\}$$

With this information, we can now define a *Refined Inference CP-net* as follows.

Definition 12. *Given a policy PL and its Inference CP-net $\mathcal{H} = \langle PL, W, T, F, V, \Upsilon, \lambda, m_o \rangle$, its correspondent* Refined Inference CP-net *is created according to Definition 9 for a modified policy PL' such that $PL' = \langle R, U \setminus U_n, P, \mathcal{I}_y, C, DSoD, SSoD \rangle$, where U_n is the set of users who are not able to infer unauthorized information as per Definition 11 and $\mathcal{I}_y$ contains the active inference threats found through Algorithm 3 for the original PL. A Refined Inference CP-net, does not stored information for any of the users in U_n.*

Theorem 3. *Given a policy with $k = |\mathcal{I}|$ inference tuples, $q = |\mathcal{I}_y|$ active inference tuples, and a set of users, U_n, that are not able to infer unauthorized information, a Refined Inference CP-net improves the performance with respect to the corresponding non-refined Inference CP-net as follows:*

- *It decreases the number of created places in q and created transitions $2q$.*
- *For each user U_n and s of his requests, there is a reduction of s verifications of the CP-net state.*
- *For each user in U_n, there is a reduction of the number of tokens placed in the interval $[s,\ s * k]$ where s is the number of access requests he issues to the system.*

Proof. We begin by proving the reduction on the components needed by the *Refined Inference CP-net*. Then, we prove the reduction on the storage required. *(i) Reduction of components of the Refined CP-net:* Let us first compute the number of components for Inference CP-net constructed based on $\mathcal{I}$. The total number of transitions is $|T| = 2k$ while the total number of places is $|W| = k$, as $k = |\mathcal{I}|$. In contrast, a Refined CP-net created using $\mathcal{I}_y$ has $|T| = 2(k - q)$ transitions and $|W| = k - q$ places. This is because for each of the non-active inference tuples, the CP-net does not contain its correspondent *BelongToTuple* and *CompletedTuple* transitions. Similarly, there is a total reduction of q created places, as a place of type β is no longer required for each of the inference tuples in $\mathcal{I}_n$. Hence, there is a decrease of q places and $2q$ transitions needed when the refined version is used. *(ii) Reduction of the information stored:* In addition, the results of the simulation can be used to avoid maintaining the state of the CP-net for users that are not able to infer information given the current policy PL. This means that the system does not have to store the tokens for these users and does not need to verify whether there is an inference risk associated with any request for those users. This saves space and computations. Suppose that user $u_n \in U_n$ makes s authorization requests to the system. If the system is not fine tuned to avoid maintaining the state of u_n in the CP-net, the system will be storing unnecessary tokens. For each of the s requests the system does not place 1 tokens in w_s. Hence, for s requests, the lower bound of the tokens placed is s. The upper bound of the stored tokens that would be maintained by the system for a user is $s * k$. This occurs when all of the s tokens placed at w_s manage to trigger all the transitions of type *BelongToTuple*, creating $s * k$ tokens where k is the number of inference tuples with which the CP-net was created. Therefore, for s queries, the system will be storing a number of tokens in the interval $[s,\ s * k]$. □

These reductions are especially important when the number of users that are not posing an inference threat and when the non-active inference tuples are large.

4.5.3 Managing Active Inference Threats

Having identified the *active inference threats* of a particular policy through Algorithm 3, we can provide additional information to administrators so they can manage the inference

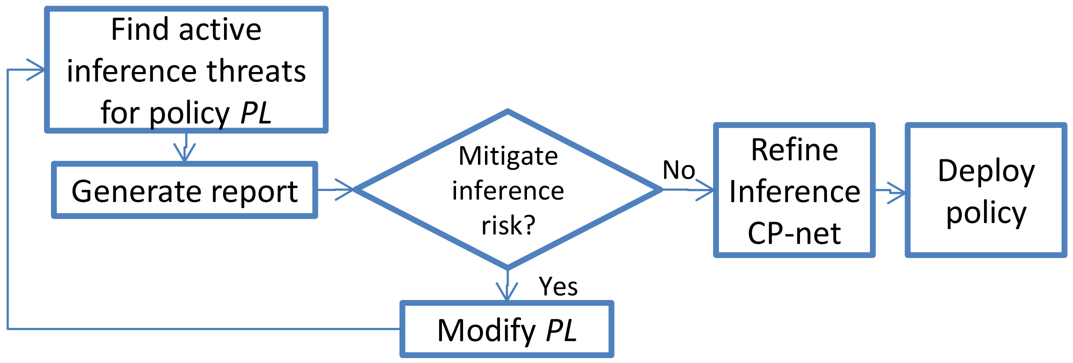


Figure 6: Managing active inference threats.

risk. Figure 6 shows the process. Once the active inference threats are identified, the system generates a report indicating the factors in the policy that are creating the existing active inference threats. With this information, the administrator can decide whether the current policy provides adequate protection against inference threats or whether it is necessary to modify it to reduce the risk exposure due to inference of information. This procedure can be run offline iteratively until the administrator finds an acceptable level of risk exposure. Once the desired level of risk exposure is achieved the administrator can release the particular policy configuration to the production system.

In an effort to help the administrator understand the risk exposure of the organization due to active inference threats, we present the results in categories. Each inference threat is categorized as *high, medium* or *low* according to the severity of the risk exposure depending on the risk associated with the inferred permissions. Recall that in the policy, each permission is assigned a quantitative risk ($rs(p, c)$), so providing a qualitative measure of the impact requires some internal processing. We transform the risk of the inferred permissions from the qualitative measure provided in the policy definition to a quantitative one. We perform this transformation to provide an easier way for the administrator to interpret the results and prioritize the possible actions. A simple threshold based categorization does not grantee a good transformation. Since different organizations may have different distributions of the risks, a different set of thresholds would need to be selected for each of them. Furthermore, establishing where to make the division between observations may be challenging and may require human input. For these reasons, we cluster the permissions based on their risk using k-means clustering technique with k equal to three which will group all the permissions in one of the three categories according to how risky they are. Before feeding the data to the clustering algorithm, some preprocessing of the data is required. First, we need to decide which of the risk associated with a permission p we are going to show to the administrator, e.g., the risk associated with p may be different in different contexts, we need to decide which risk should be considered. We use the maximum of all the risks associated with a permission, which is given by $\max\limits_{\forall c \in C} rs(p, c)$. It is also possible to use the average, the minimum or other statistical measure of all the risks associated with a permission. However, we decided to use the maximum because it does not underestimate the risk exposure, whereas an average may provide a false sense of security.

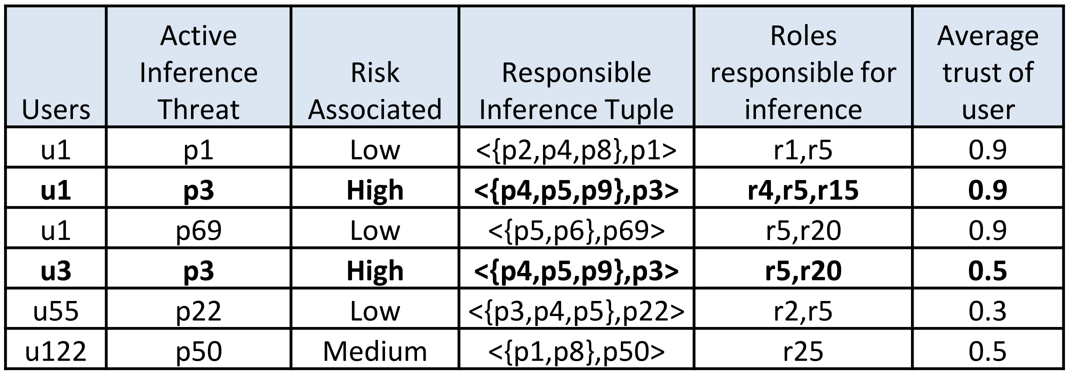
Definition 13. *The* standardized risk value of an inference threat *associated with permission p is defined as follows:*

$$rs_{p,s} = \frac{rs(p) - \max\limits_{\forall p_i \in P}(rs(p_i))}{\max\limits_{\forall p_i \in P} rs(p_i) - \min\limits_{\forall p_i \in P}(rs(p_i))}$$

where $rs(p) = \max\limits_{\forall c \in C} rs(p, c)$ *denotes the maximum risk associated with permission p.*

In the previous formula, each of the permissions in the system is considered when the standardized version of the permissions associated with active inference threats are computed. With this standardized information, the clustering algorithm is run and the permissions are classified. The following example shows an output presented to the administrator.

Example 4. *Figure 7 presents an example of an output shown to the administrator after the simulation process. The first and second columns show the users that may infer information and the permission ($\langle object, read \rangle$) that the user may be able to infer. The third column shows the severity of the inference threat. Then, the inference tuple that allows the inference and the roles that are activated for this purpose are presented. Finally, the average trust value of the user is presented if it is available (this value is the average trust in all context). The administrator can use this information to make an informed decision as follows.*



Users	Active Inference Threat	Risk Associated	Responsible Inference Tuple	Roles responsible for inference	Average trust of user
u1	p1	Low	<{p2,p4,p8},p1>	r1,r5	0.9
u1	**p3**	**High**	**<{p4,p5,p9},p3>**	**r4,r5,r15**	**0.9**
u1	p69	Low	<{p5,p6},p69>	r5,r20	0.9
u3	**p3**	**High**	**<{p4,p5,p9},p3>**	**r5,r20**	**0.5**
u55	p22	Low	<{p3,p4,p5},p22>	r2,r5	0.3
u122	p50	Medium	<{p1,p8},p50>	r25	0.5

Figure 7: Inference simulation results.

The report shows that u_1 is able to infer permissions p_1, p_3 and p_{69}. According to the risk associated with each threat, the inference of p_1 and p_{69} is not too critical. However, the possible inference of permission p_3 is important. This may lead the administrator to modify the policy. The simulation results show that the inference arises because u_1 is assigned to roles r_4, r_5, r_{15}.

The output also allows the administrator to focus on the inference threats that are more severe to prioritize his actions. In this case u_1 and u_3 may infer p_3 which has a high risk. The administrator may also notice that in average u_3 seems to maintain a lower trust value than u_1. Based on this information, he may prioritise to modify the policy for u_3. The report shows the set of roles responsible for the inference threat; the administrator may remove the authorization one or more of them to u_3 to eliminate the inference threat.

In this way, an administrator can have an overview of the existing inference threats and prioritize his actions. If the administrator judges appropriate, he may modify the user-to-role assignment to mitigate the risk of inference. In some cases, the reduced number of employees may lead to the acceptance of inference risks, whereas in others, the organization may take steps to avoid those inferences.

Table 2: Experiment parameters for the policy generation for the risk-and-trust RBAC framework.

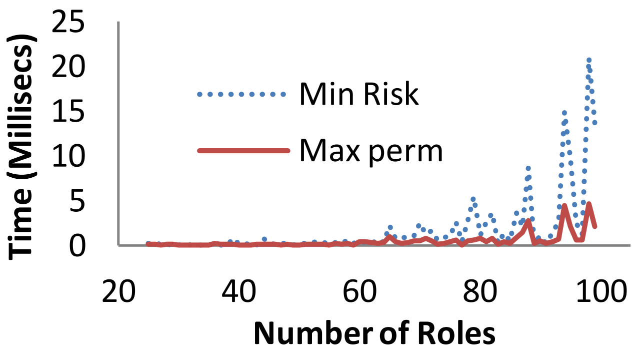
Ratio of number of roles to number of users	1:1
Ratio of number of roles to number of users assigned to roles	6:1
Ratio of number of roles to directly assigned permissions	1:5
Ratio of number of roles to constraints	5:1
Ratio of number of roles to *active* inference tuples	10:1
Ratio of number of requested permissions to maximum user assignment multiply by directly assigned permissions per role	1:2
Maximum number of junior roles	3
Number of contexts	1

4.6 IMPLEMENTATION RESULTS

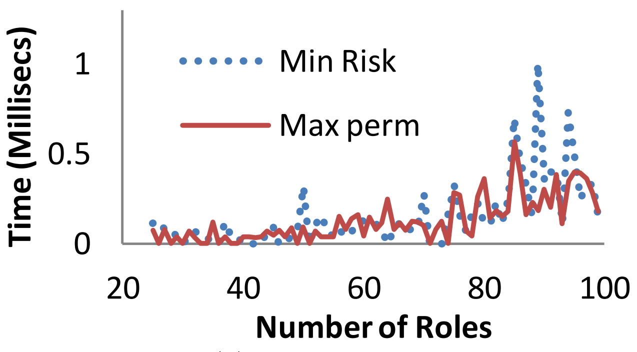
In order to evaluate the proposed algorithm, we generated synthetic *well-formed policies*, as per Definition 6. The policies were generated randomly according to the ratios shown in Table 2, which were chosen to match the ones used in [116][1]. The risk assigned to each permission in the policy was randomly assigned a value in the interval [0,100] using a uniform distribution. The users' trust thresholds were also randomly assigned using a uniform distribution. Each point in our figures represents the average time of running the algorithm for 30 different policies. Requests were randomly generated. The time required to process requests that could not be granted was very low. For this reason, we only present the results for granted requests.

Comparing Selection Heuristics: Our first experiment contrasts the performance of the algorithm under two different heuristics to select the next role in line 36. Heuristic 1 selects a role that provides the minimum risk (*min risk*), and heuristic 2 selects a role that provides the maximum number of permissions in the request set that have not been covered by previously selected roles (*max perm*). In order to compare these heuristics, the

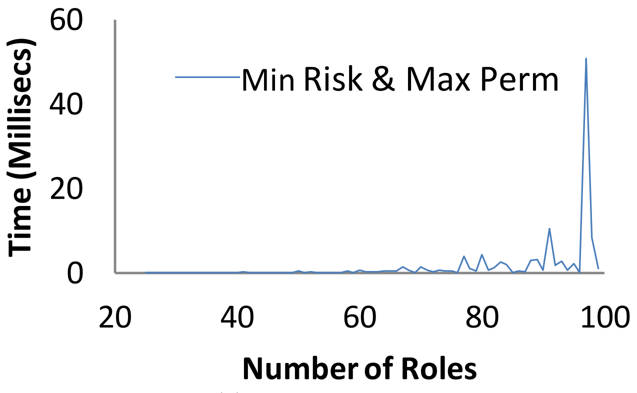
[1]Since their experiments did not include hybrid hierarchy and risk, we had to adapt slightly the parameters. For instance, the ratio of permissions *directly assigned* for a roles in [116] is the same ratio of *authorized* permissions for a role in our experiment to account for the effect of hierarchical relations.



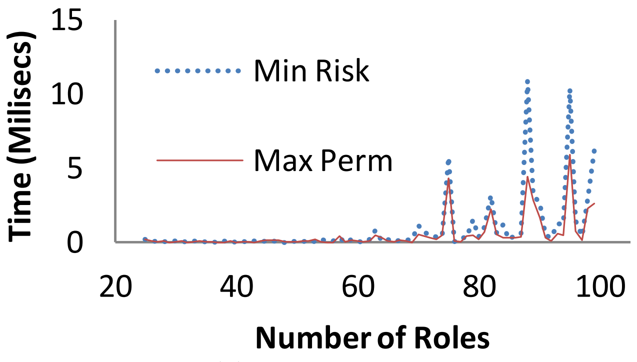
(a) $I{:}A{:}IA = 0{:}0{:}1$



(b) $I{:}A{:}IA = 1{:}0{:}0$



(c) $I{:}A{:}IA = 0{:}1{:}0$



(d) $I{:}A{:}IA = 1{:}1{:}1$

Figure 8: Comparison of selection heuristics for different types of hierarchy proportions.

same policy and requests were used for both algorithms. The performance of the heuristics depends on the proportion of *I, IA* and *A-hierarchy* relations. As shown in Figure 8a, when all the hierarchical relations were of type *IA*, the *max perm* heuristic is faster. Figure 8b presents the results of having all the relations of type *I*. There, 61% of the times the *max perm* heuristic behaved better than the other heuristic. In contrast, when all the relations in the hierarchy are of type *A*, as shown in Figure 8c, the results of the two heuristics are equivalent. We also compared the results for policies that contain the same number of *I, A* and *IA* relations. Figure 8d shows that *min risk* heuristic is slower for all the policy sizes. Our results suggest that when the amount of *A-hierarchy* relations is greater than the other two types of hierarchical relations, the two heuristics behave similarly. However, when the proportions are different, the *max perm* heuristic is consistently faster than the *min risk* heuristic. In all the experiments, the *min risk* heuristic never outperformed the *max perm* heuristic. For that reason, in the following experiments, we only present the results for the *max perm* heuristic.

The time required to find a solution for policies of the same size, but with different proportions of *I, A*, and *IA* hierarchical is very similar for policies with less than 75 roles (it took less than 0.2 milliseconds in all cases). For bigger policies the results change. The time required for finding a solution for policies where all relations are of type *I*, is smaller than for policies with all hierarchy relations of type *A*. This difference occurs because the number of roles users can activate increases when the hierarchy relations are of type *A* (e.g., the search space includes the roles the user is assigned to and all their junior roles). In contrast, when all the relations are of *I*-type, the number of roles users can activate are uniquely those directly assigned to them. When all the relations are of type *IA*, the algorithm takes more time than when they are all of type *I*, but it takes less time than when the relations are all of type *A*. The time required to find a solution when all the relations are of type *IA* is very similar to the cases where the proportion of the three hierarchy types is the same.

Comparing Percentage of Granting Requests for Different Proportions of Misbehaving Users: This experiment shows how the system behaves as the percentage of users that misbehave increases, and their trust thresholds are reduced. Initially, we randomly generated policies, and requests that were all granted. This is represented by the line of 0% users

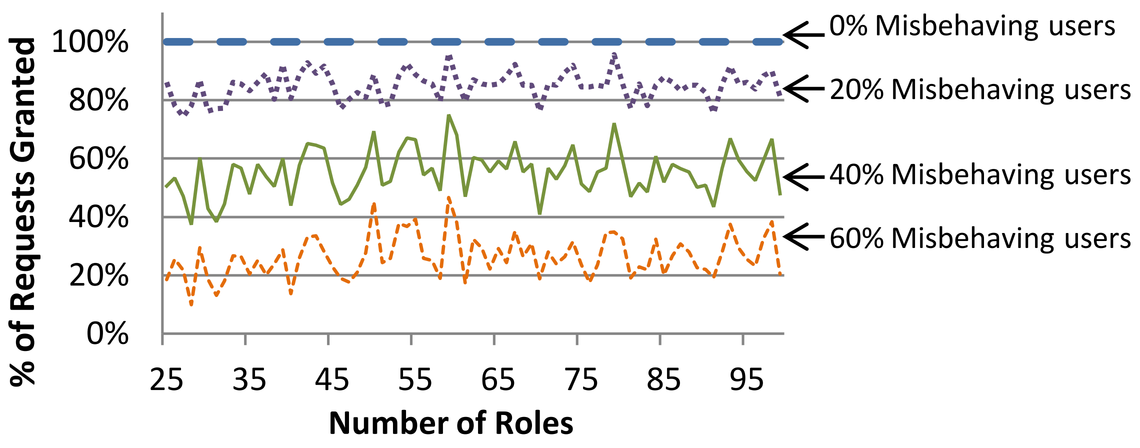


Figure 9: Comparison of granted requests for different percentage of misbehaving users.

misbehaving in Figure 9. For the same policies and requests, we randomly selected some users, and reduced - randomly again - their trust thresholds; then we ran the experiments again to see how many requests were denied because of decreases in trust of some users. The results for 20%, 40% and 60% of users misbehaving are shown in Figure 9. As the number of misbehaving users increases, the number of requests granted decreases. The lines are not flat, as the number of requests denied depends on the trust threshold of the selected roles, as well as the random reduction of the user's trust value. The results of this experiment show that our framework is able to deny access to misbehaving users, thus adapting to prevent possible insider attacks.

Comparing Objective Functions: We compared the risk exposure when two different objective functions are used to select the roles to be activated. The objective functions compared are as follows. (1) Our proposed objective function: minimize the risk which is presented in lines 20 to 23; we refer to it as *objective min risk*, and (2) The traditional objective function used in RBAC systems which consists on minimizing the number of roles to be activated; we refer to it as *objective min num roles*. When all the hierarchical relations are of type *IA*, the *objective min num roles* always found riskier solutions, as shown in Figure 10. Similarly, when the ratios of *I, A* and *IA* hierarchies were proportional, 95% of the time the *objective min num roles* provided riskier solutions. The *objective min num roles* was also

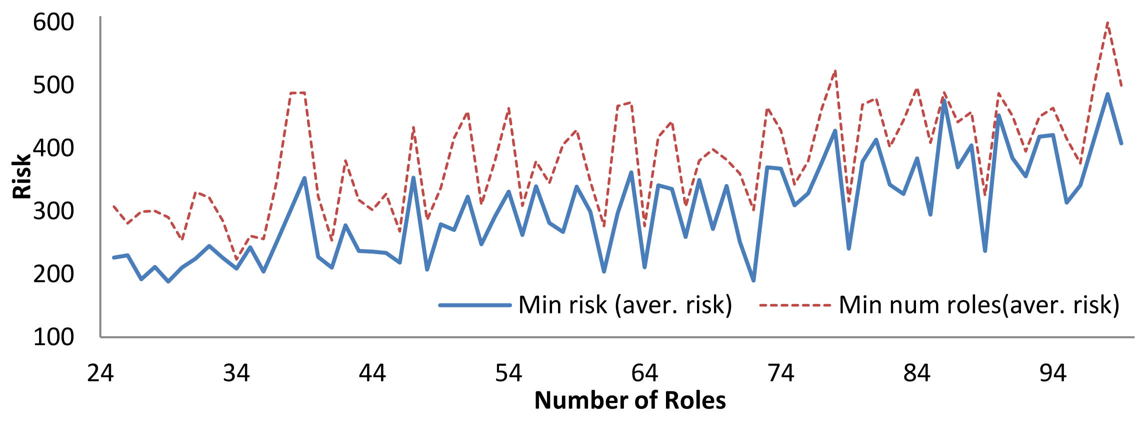


Figure 10: Risk exposure using our algorithm (min. risk) compared to the risk of traditional role activation algorithm (min. num. of roles) when all relations are of type IA.

riskier in 51% of the cases when all the relations were of type I. Interestingly, for policies in which all the hierarchical relations were of type A, the risk of the solutions found by both objective functions were the same. This is because there are no permissions directly acquired by any role that can reduce the number of roles to be selected. The time required to find a solution for both objective functions were very similar, and the number of roles selected by both approaches were always the same for all hierarchy ratios. Hence, we believe our objective function and the proposed algorithm is appropriate, as it reduces the risk and does not augment the number of selected roles. Finally, our results suggest that the time required to answer an authorization query is acceptable.

4.7 CHAPTER SUMMARY

In this chapter, we presented our most general framework for insider threat prevention. As part of this framework, we provided a methodology to determine the risk exposure of activating a set of roles. This methodology considers the risk of inference of unauthorized permissions. To manage this risk, we provided a CP-net based-approach. Additionally, we formulated an optimization problem to minimize the risk exposure of each access request. To solve this problem, we proposed an algorithm and evaluated two different heuristics. We showed that the risk exposure is reduced with respect to standard RBAC-based enforcement algorithms. Finally, we provided a simulation methodology to identify active inference risks that arise given a particular access control policy. The resulting simulation results can be used by administrators to mitigate inference risks.

In the next chapter, we present our second framework, which aims at mitigating the risk of *a posteriori* obligations.

5.0 OBLIGATION-BASED FRAMEWORK TO REDUCE RISK EXPOSURE AND DETER INSIDER ATTACKS

In this chapter, we present our research on risk management for obligation-based access control systems. We begin by explaining the reasons that support the use of obligations as a valid indicator to identify if a user is about to launch an attack in Section 5.1. Then, we present our framework in Section 5.2. The detailed methodology to compute users' trust values is presented in Section 5.3. The methodology to identify policy misconfigurations and outliers is presented in Section 5.4. Finally, we evaluate our approach in Section 5.5.

5.1 WHY USING A POSTERIORI OBLIGATIONS AS AN INDICATOR?

Although psychological precursors may allow early detection of insider threats, monitoring users can be challenging because of privacy and legal concerns. To tackle this problem, we propose to monitor and evaluate the users' behavior towards *a posteriori* obligations as a way to determine two highly relevant psychological indicators: these are *disregard of authority* and *lack of dependability* [61]. As part of the *disregard of authority*, the employee ignores rules, authority or policies, and feels above the rules or that they only apply to others. Greitzer *et. al* [60] classified this indicator as the second most important psychological precursor. As part of *lack of dependability* an employee is unable to keep commitments and is unworthy of trust. When users stop fulfilling their *a posteriori* obligations, they are disregarding authority and they may be less dependable. This may be due to lack of interest and the fact that they may be occupying their time with other activities such as preparing an attack.

Monitoring and evaluating users' patterns of fulfilling and violating obligations has several advantages with respect to existing approaches to collect psychological precursors, which were outlined as part of the challenges in Chapter 1. The rate of fulfillment of obligations can be used as one of the metrics to assess the employees' performance that does not introduce any subjective information in the system. Since employees are being paid to perform their jobs, using performance metrics is a well-accepted practice [58]. An additional advantage of this methodology is its ability to include up-to-date information of a user's behavior. Recall that traditional ways to measure the employee's psychosocial state usually are incorporated slowly into the system. Therefore, we argue that the obligation-based trust values are an objective measure of the actual, up-to-date performance of the users and hence capture their real behavior.

5.2 PROPOSED FRAMEWORK

In this section, we present our proposed framework. First, we present the policy definition model that we call Trust-and-oBligation based Core RBAC model (Core TB-RBAC Model). This model extends the standard RBAC model [51]. We note that the methodology we propose to evaluate the trust can be used for any access control system that includes *a posteriori* obligations. Then, we present the architecture of the system.

5.2.1 The Core TB-RBAC Model

We extend the standard core RBAC model with obligations, risk and trust; it includes the following components.

- U is the set of users, R is the set of roles, P is the set of permissions defined as $P = OPS \times OBJ$, where OPS is the set of operations and OBJ is the set of objects in the system.
- $\mathcal{B}$ is the set of *a posteriori* obligations as defined in Definition 14 below and S is the set of sessions.

- $UA \subseteq U \times R$ is the user to role assignment, as in standard RBAC.
- $\mathcal{BP} \subseteq P \times 2^{\mathcal{B}}$ is *obligation-aware permission* set, where, $bp \in \mathcal{BP}$ is a tuple $\langle p \in P, BS \subseteq \mathcal{B}\rangle$ that indicates that once p has been exercised all the obligations in BS need to be fulfilled.
- $\mathcal{PB}A \subseteq R \times \mathcal{BP}$ is the assignment of obligation-aware permissions to roles. Permissions associated with different roles can have different obligations associated with them. This function replaces permission to role assignment (PA) of traditional RBAC.
- Function $session\_user : S \rightarrow U$ maps a session onto the corresponding user and function $session\_roles : S \rightarrow 2^R$ maps a session onto a set of roles.
- Each user $u \in U$ is assigned an obligation-based trust value, $trust(u,t) \in [0,1]$ at time t. If $trust(u,t) = 0$, the user is not trusted. When $trust(u,t) = 1$ the user is completely trusted to perform *a posteriori* obligations. This value is automatically updated by the framework every time the user fulfills or violates an obligation.
- $P_{au} : (r \in R) \rightarrow PS \subseteq P$ is a function that returns the permissions PS assigned to role r. Formally, $P_{au}(r \in R) = \{p \mid (r, \langle p, BS\rangle) \in \mathcal{PB}A \wedge \langle p, BS\rangle \in \mathcal{BP}\}$.
- $B_{au} : (r \in R) \rightarrow B \subseteq \mathcal{B}$ is a function that returns the set of obligations that would be assigned to the user that activates role r. Formally, $B_{au}(r \in R) = \{b \mid (r, \langle p, BS\rangle) \in \mathcal{PB}A \wedge \langle p, BS\rangle \in \mathcal{BP} \wedge b \in BS\}$.

We define an *a posteriori* obligation as follows:

Definition 14. *An* a posteriori obligation b *is defined as a tuple* $b = \langle \mathcal{A} \subseteq OPS \times OBJ,\ D,\ \varphi\rangle$ *where*

1. $\mathcal{A}$ *is a set of actions that need to be performed to fulfill the obligation. The user assigned to* b *needs to perform all* $a \in \mathcal{A}$ *in order to fulfill the obligation.*
2. D *specifies how much time a user has to fulfill the obligation after the obligation is assigned to him.*
3. $0 \leq \varphi \leq 1$ *indicates how critical it is for the organization that the obligation is performed in time; where* $\varphi = 1$ *means that it is very critical and* $\varphi = 0$ *means that the obligation is not critical at all.*

In order to refer to a particular component of obligation $b \in \mathcal{B}$, we use the dot notation.

For instance, $b.\varphi$ returns the criticality value of the obligation.

Obligation Instantiation: Obligations are assigned to users when they activate associated roles, as follows.

Definition 15. *When user u activates role r in a session, for each a posteriori obligation b associated with r, the system instantiates the obligation creating the tuple: $\langle u,\ b,\ \tau,\ \mathcal{S}\rangle$ where:*

1. *u is the user that needs to fulfill the obligation.*
2. *b is the obligation that needs to be fulfilled by u.*
3. *τ is the time when the obligation is acquired by user u.*
4. *$\mathcal{S}$ is the state of the obligation which is initially set to pending.*

Once an *a posteri* obligation has been triggered, it can be in one of the following states: *pending, fulfilled* or *violated.* The interval within which the obligation needs to be fulfilled is $[\tau,\ \tau + b.D]$. The obligation is *pending* when the user has not performed the actions required by the obligation and the deadline to perform it has not passed. The obligation is *fulfilled* when the user performed the required actions within the stipulated time interval. Conversely, the obligation is *violated* when the user does not perform the required actions during the valid interval of time.

Access control decision process: To obtain the permissions authorized for a role, users need to *activate* the role in a *session.* Hence, a user u requesting a permission set $PS \subseteq P$ is granted access to PS if the following conditions hold:

1. $\exists\ RS \subseteq R \wedge\ PS \subseteq \bigcup_{r \in RS} P_{au}(r) \wedge (\forall r \in RS : (u, r) \in UA)$, which means that there is a set of roles RS that can provide all the permissions in PS and all of the roles in RS are assigned to user u.
2. The system trusts the user enough to perform all the *a posteriori* obligations that would be acquired by activating the set of roles RS:
   $\forall\ r \in RS, BS \subseteq B_{au}(r), b \in BS : trust(u, t) \geq b.\varphi$

In this dissertation, we do not specify how to select RS so that they respect the least privileged principle; however, this can be easily done using one of the algorithms presented in [119].

In the following example, we illustrate how different obligations have different criticality values associated with them.

Example 5. *Consider a manufacturing organization. When a new supply container arrives, the employee in charge needs to access the system and register it; this in turn triggers obligation b_1. This obligation corresponds to updating the inventory state after reviewing an order of a component to produce their most sold product. If the user fails to fulfill this obligation on time and the ordered supplies have defects, the entire operation of the organization would be negatively impacted. In case the defect is difficult to notice and nobody recognizes the lack of quality of the supplies, the organization would manufacture defective products. This may lead to a decrease on the goodwill of the organization and may also result in fines and additional product repairing costs for the enterprise. In a second scenario, the defect of the supply is noticed by a different employee during production and the operation is stopped due to the lack of available materials. In this case, the production line is stopped and orders may not be fulfilled on time causing delays, fines and loss of goodwill. Since the entire operation of the organization may be severely affected due to the lack of fulfillment of b_1, its criticality for the organization is* high*, so we assign a value of $b_1.\varphi = 0.9$.*

Obligation b_2 requires the obliged user to review a report of expenditures by the end of the week. This obligation aims at identifying discrepancies every week. However, an accountant reviews the report at the end of each month, so the discrepancy would be found eventually. The impact of violating this obligation is medium *because not performing the obligation does not have long-term repercussions for the organization. Only in the short term the discrepancy would exist. Hence, we assign $b_2.\varphi = 0.5$. Finally, when a user registers a new sale, obligation b_3 is triggered requiring the user to update the internal review file with comments regarding the interaction with the client. This obligation has* low *impact because not updating the file does not affect the operations of the organization. Thus, we can assign $b_3.\varphi = 0.3$.*

The above example shows that the criticality of an obligation depends on the impact of its violation. The risk exposure an organization faces when an *a posteriori* obligation is assigned to a user is a function of the criticality of that obligation and the likelihood that the user will default on it. The larger the trust value of the user, the less likely he would default on an obligation. We use the criticality value of an obligation as a threshold that indicates how trusted a user needs to be in order to be assigned to a particular *a posteriori* obligation. Note that the criticality of the obligations can be expressed qualitatively and later mapped to a quantitative measure. Hence, the policy specifier can use any of the existing risk assessment methodologies (e.g., [108, 7]) to assign these values.

5.2.2 Risk-and-Trust Obligation Framework

Figure 11 presents the architecture of our framework. First, we describe the functionality of each of the modules in the system and then we show the steps followed when a user's access request arrives at the system. The *Obligation-based Trust Module* monitors the users of the system and is in charge of determining the trust values associated with all of them. The trust values are stored in the Trust Repository.

The *Administration Module* generates alerts of possible policy misconfigurations related to *a posteriori* obligations and suspicious users. The *Clustering Module* finds the patterns of misbehavior and the *Report Module* generates the corresponding alert reports for the administrator. This process is explained in Section 5.4.

In addition, the framework contains the *Enforcement Module* which consists of the Policy Enforcement Point (PEP), the Policy Decision Point (PDP) and the Obligation Handler. The *Obligation Handler* is responsible for maintaining the state (*pending, fulfilled* or *violated*) of the instantiated obligations in the system up-to-date. This information is stored in the *Obligation State Repository.* Every time an obligation changes its state to *fulfilled* or *violated*, the system informs the *Obligation-based Trust Module* of the new information, which in turn updates the trust value of the corresponding user. The PEP is in charge of intercepting all the access requests of the users in the system and it passes them to the PDP, which evaluates the request according to the policy stored at the Policy Information Point (PIP). The PDP returns the grant or deny decision to the PEP, which enforces the decision.

Figure 12 presents the process that is followed by the system to determine whether an access request is granted or denied. When a request is received by the PEP, it forwards the request to the PDP, which retrieves the set of roles that need to be activated to grant

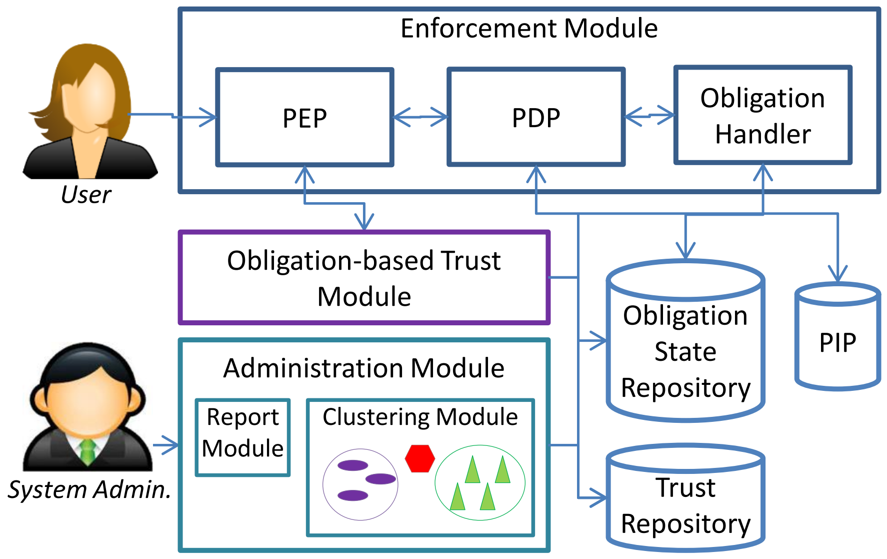


Figure 11: Architecture of the risk-and-trust obligation framework.

the access. If the system cannot find such a set, the request is denied as the user is not authorized for roles that provide the requested permissions. If the roles that provide the privileges are found, the system evaluates whether activating them would create any *a posteriori* obligations for the user. If so, the PDP retrieves the trust value of the user and determines whether the value offsets the criticality of the obligations that would be assigned to the user. When the user is trusted enough to complete successfully such obligations, the access is granted and the *Obligation Handler* instantiates them according to Definition 15. In case the user is not trustworthy enough to fulfill one or more of the obligations that would be assigned to him, the system denies the access request.

5.3 TRUST COMPUTATION

In this section, we present the methodology to compute the trust of a user. As noted by Greitzer *et. al* [59], one of the limitations of threshold-based approaches is the fact that smart attackers would try to stay within the threshold to avoid being detected. Hence, the

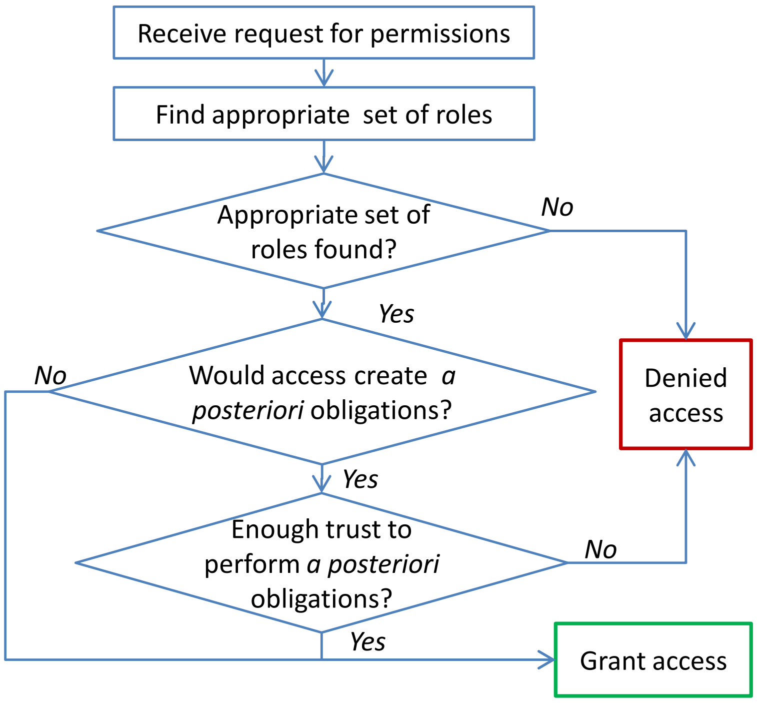

Receive request for permissions
Find appropriate set of roles
Appropriate set of roles found?
No
Yes
Denied access
No
Would access create *a posteriori* obligations?
Yes
Enough trust to perform *a posteriori* obligations?
No
Yes
Grant access

Figure 12: Processing flow of an access request.

trust computation mechanism needs to account for strategically controlled variations on the user's behavior. Strategic changes in behavior occur when a user first constructs a good level of trust and then starts misbehaving. In addition, the trust value should provide a way to discern when the user accidentally does not perform an obligation. We want to reduce the trust value to account to the bad behavior, but give the opportunity to users to redeem themselves if they have defaulted obligations by mistake. In addition, the trust value should include a group factor to determine whether the evaluated user is the only one among the users assigned to a particular obligation, who is repetitively violating the obligation.

Our trust model considers the following aspects to find the obligation-based trust value ($trust(u, t)$):

1. His recent behavior.
2. His historical behavior, which shows how many times he has fulfilled or defaulted on assigned obligations.

3. His sudden changes in the behavior, which allows the system to penalize the user for negative changes in behavior.
4. His performance with respect to other users.

Our trust model is inspired by that of Srivatsa *et. al* presented in [106], where the first three components are included; however, we compute the trust values differently. In addition, to capture the overall group behavior and its relation to that of an individual user, we include the drift from the group. In Section 2.5.1, we discuss in further detail the differences between our approach and the one presented in [106].

5.3.1 Trust Methodology

An observation o of a user's behavior consists of a *fulfilled* or *violated* obligation ($o = \langle u \in U, b \in \mathcal{B}, final\_status \rangle$). We assume a user's observations are ordered based on their generation timestamps and that they are grouped in what we call *observation groups*. Each observation group contains a fixed maximum number of observations x. If at a particular time instant there are m logged observations, there would be $n = \lceil m/x \rceil$ groups. We denote observation groups as $T_1, ..., T_i, ...T_n$, where, group T_n contains the most recently logged observations and T_1 contains the oldest observations. Each group T_i for $2 \leq i \leq n$ is guaranteed to contain x observations while T_1 may contain less than x observations. The groups are recalculated every time a new observation is logged to the system. For instance, suppose that the fixed maximum number of observations per group is set to three ($x = 3$), and that, at time t_{19}, the system has logged six observations $o_1, ..., o_6$, where o_1 is the first and o_6 the last observation logged, respectively. At t_{19} there are two groups T_1 and T_2, where T_1 contains [o_1, o_2, o_3] and T_2 contains observations [o_4, o_5, o_6]. Suppose that at t_{22} another observation o_7 is generated; it causes a re-grouping of observations as follows: a new group T_3 is created containing the most recent observations [o_5, o_6, o_7], T_2 contains [o_2, o_3, o_4] and T_1 contains the oldest observation o_1. Hence, T_3 contains the most recent behavior of the user. In this way, at time instant t the observation groups are created according to the observations available and each group represents the behavior of a user in a period of time.

Table 3 contains the notation that we use in the rest of the dissertation. We use multisets

Table 3: Notation for Chapter 5 (obligations).

$m(M, b)$	Function that returns the multiplicity (number of elements of type b) contained in multiset M.
$\mathcal{B}$	Set of obligations in the system
GB_u^T	Multiset that contains the obligations fulfilled by user u in observation group T
BB_u^T	Multiset that contains the obligations violated by user u in observation group T
TGB^T	Multiset that contains the obligations fulfilled by all users in observation group T
TBB^T	Multiset that contains the obligations violated by all users in observation group T
$totalRisk(u, T)$	Function that returns the total risk of the obligations fulfilled and violated in observation group T by user u

to refer to the observations in each group. A *multiset* is a collection in which each element may appear more than once. For instance, a multiset of obligations $M = \{b_1, b_2, b_3, b_1\}$ contains obligation b_1 twice. The *multiplicity* is a function that returns the number of times an element appears in a multiset and is defined as $m : Multiset \times element \rightarrow int$. In the previous example $m(M, b_1) = 2$ as obligation b_1 appears two times.

Definition 16. *The* raw trust $RT_u[T]$ *of user* u *in observation group* T *is calculated using the following expression:*

$$RT_u[T] = \frac{\sum_{b \in \mathcal{B}} b.\varphi * m(GB_u^T, b)}{\sum_{b \in \mathcal{B}} b.\varphi * m(GB_u^T, b) + \sum_{b \in \mathcal{B}} b.\varphi * m(BB_u^T, b)}$$

The raw trust captures the behavior of user u in period defined by the observation group T and it is a weighted average of the number of obligations fulfilled over the total number of obligations assumed by the user. The weights are determined by the importance of the obligations themselves (φ). In this fashion, an obligation that is very critical to the organization has a heavier impact on the raw trust, than one that is not so critical. A user that has violated all his acquired obligations has a raw trust equal to zero. In contrast, when the user has promptly fulfilled all his assigned obligations, his raw trust is equal to one.

Definition 17. *The* historical trust *of user* u *for observation group* T_n*,* $H_u[T_n]$*, is computed*

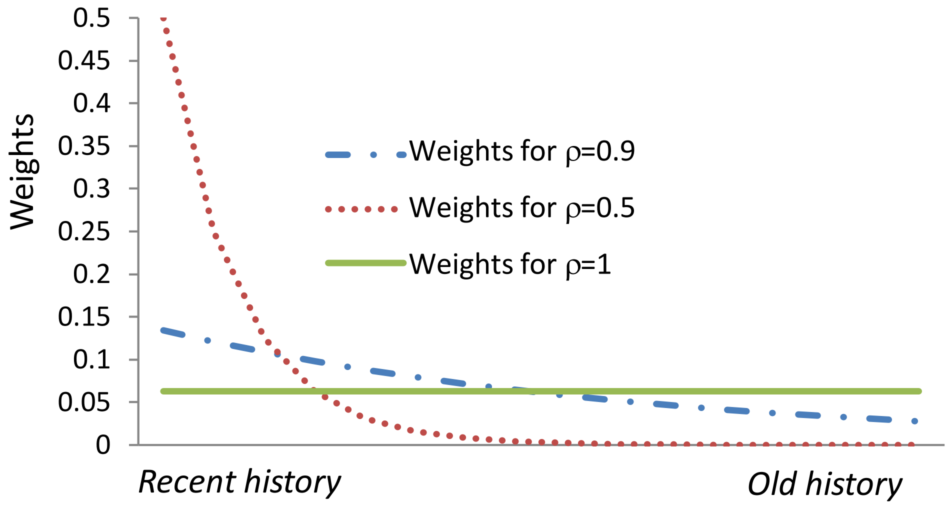


Figure 13: Effect of ρ on the historic trust (Definition 17) considering that all the obligations have the same criticality.

as follows:

$$H_u[T_n] = \sum_{k=1}^{n-1} RT_u[T_{n-k}] * w_k$$

where w_k is the weight of observation group T_{n-k} which is calculated as follows:

$$w_k = \frac{\rho^{k-1} + totalRisk(u, T_{n-k})}{\sum_{i=1}^{n-1}(\rho^{i-1} + totalRisk(u, T_{n-i}))}$$

where $0 \leq \rho \leq 1$.

When the weight of recent events is much higher than those of previous observations, the system allows the users to improve their trust values rather quickly because it prioritizes the most recent behavior. The weight w_k has two components, decay of historical information and the criticality of the observation groups. The first one is provided by ρ^{k-1} and allows the system administrator to change the importance of each historical observation group. Figure 13 depicts the effect of ρ on w_k. When $\rho = 1$, all the observation groups have the same weight; hence all the periods that contribute to the historical trust have the same importance. In this case, the historical trust is equivalent to the average of the raw trust. When $\rho = 0.5$, some of the older observations do not have much weight and we would be losing some information. In contrast, when $\rho = 0.9$ there is a desirable effect in which all the historical observations are considered, but the more recent ones have more weight than the older ones. We prefer to have ρ nearby 0.9 to maintain freshness of the observations while considering all the historical information available.

The second component of the weight w_k corresponds to the total criticality of the obligations that are included in observation group T_{n-k}. This component allows us to provide a higher weights to observation groups that contain obligations with higher criticality values and inhibits *strategic users* from improving their trust values by fulfilling only low criticality obligations.

Definition 18. *The* trust fluctuation $D_u[T]$ *of user* u *in observation group* T *is defined as follows:*

$$D_u[T] = RT_u[T] - H_u[T]$$

which represents the variation of the current trust with respect to the historical trust.

When $D_u[T] \geq 0$, the user has improved or maintained his behavior with respect to his historical trust. In contrast, when $D_u[T] < 0$, the user behavior has worsened.

It is also desirable to discover when a user does not fulfill a particular obligation more frequently than his peers, which may represent attempts to sabotage the operation. We capture it using the notion of *group drift.*

Definition 19. *The* group drift, $G_u^b[T]$, *of obligation* $b \in \mathcal{B}$ *for user* u *in observation group* T *is defined as follows:*

If $m(BB_u^T, b) = 0$, *then* $G_u^b[T] = 0$. *Otherwise:*

$$G_u^b[T] = \frac{m(BB_u^T, b)}{m(TBB^T, b)} - \frac{m(BB_u^T, b) + m(GB_u^T, b)}{m(TBB^T, b) + m(TGB^T, b)}$$

Here, $0 \leq G_u^b[T] \leq 1$. If the user has not violated any obligation of type b, his group drift is zero. In addition, the group drift is zero if user u is the only one that has been assigned to obligation b, as there is no evidence that shows his behavior is drifting from the group (in fact, there is no group). When the number of users assigned to b increases, there is more evidence as to how far apart from the group the user is. When $G_u^b[T] = 0.5$, it means that half of the total assigned obligations (fulfilled and violated in observation group T) were violated by u. A $G^b_u[T]$ close to one implies that user u is the only person in a large group that has violated the obligation.

A big drift from the average may actually predict attempts to **sabotage** the operation. This is especially relevant when an obligation that has a large criticality value (φ) is consistently violated. This behavior is suspicious and is penalized as follows.

Definition 20. *The* benchmark penalization *of user u in observation group T, $PG_u[T]$, is calculated as follows:*

$$PG_u[T] = \sum_{\forall b : G^b_u[T] > \chi_b} \delta_b$$

where $0 \leq \chi_b \leq 1$ is a threshold for obligation $b \in \mathcal{B}$ that specifies how far apart from the group a user needs to be in order to be penalized and δ_b is the penalization received for drifting from the group substantially.

In the previous definition, when $G^b_u[T] > \chi_b$, user u is an outlier that does not fulfill obligation b, and should be penalized by an amount of δ_b. Note that the penalization and the threshold of each violated obligation $b \in \mathcal{B}$ (δ_b and χ_b) may have different values in the system depending on the importance of the obligation ($b.\varphi$). Finally, we compute the total obligation-based trust values for user u at time t, which is equivalent to finding the trust value for observation group T_n (remember that the most recent observation group is denoted by T_n).

Definition 21. *The* individual obligation-based trust
$trust(u, T_n)$ of user u in observation group T_n is calculated as follows:

$$trust(u, T_n) = \begin{cases} trust(u, T_{n-1}) & \text{if } \gamma(D_u[T_n]) = 0 \\ 0 & \text{if } \mathcal{T} \leq 0 \\ \mathcal{T} & \text{otherwise} \end{cases}$$

where

$\mathcal{T} = \alpha \times RT_u[T_n] + \beta \times H_u[T_n] + \gamma \times (D_u[T_n]) - PG_u[T_n]$ *and* $\alpha + \beta + \gamma = 1$.

Here, α represents the weight of the current behavior, β represents the weight of the historical information and γ the weight of sudden changes of behavior. We use two possible values for this latter weight, γ_1 and γ_2, to be able to penalize heavily negative changes in behavior while allowing users to regain trust slowly for positive changes. Letting $\gamma_1 < \gamma_2$, when $D_u[T_n] \geq 0$, we use γ_1 and when $D_u[T_n] < 0$ (the user behavior has worsened), we use γ_2. In this way, the user takes longer to regain trust than to lose it. We show the effect of these weights in Section 5.5.

In Definition 21, if the user does not change his behavior his trust value remains unchanged with respect to the previous interval of time. When the $\gamma * D_u[T_n] - PG_u[T_n]$ is too small making $\mathcal{T}$ negative (recall that $\gamma * D_u[T_n]$ is negative when a negative change of behavior occurs), the new trust value is zero, which is the minimum possible. Finally, when none of these two cases happen, the trust is updated according to the current and historical behavior, the behavior fluctuations and the benchmark penalization.

Contrasting our trust computation approach: Our proposed trust computation methodology is an extension of the trust computation methodology presented by Srivatsa *et. al* in [106]. We decided to base our methodology on their approach because of its strengths [34]. Srivatsa *et. al* approach was designed for decentralized overlay networks. In their work, the final trust value of a node is based on its current and historical behavior and sudden changes of behavior. This methodology is not adequate for measuring how trusted a user is to fulfill a particular obligation because adversaries that know how their trust values are computed may try to manipulate the system in the following ways. *(i)* In [106], all the failures or good behaviors have the same weights. In the case of obligations, this assumption is not valid as each obligation has its own criticality value. An adversary that wants to avoid detection would maintain an adequate trust value by fulfilling only non-critical obligations while violating critical obligations. *(ii)* Similarly, their historical value does not include the criticality of the obligations to prevent strategic users from manipulating the trust computation by fulfilling only low-criticality obligations. *(iii)* Their approach does not include group behaviors as part of the metrics to determine how trusted a user is. However, including group drift would allow us to identify users trying to sabotage particular operations by avoiding the fulfillment of one or more obligations. For these reasons, current methodologies are not adequate to measure how trusted a user is to fulfill a particular obligation.

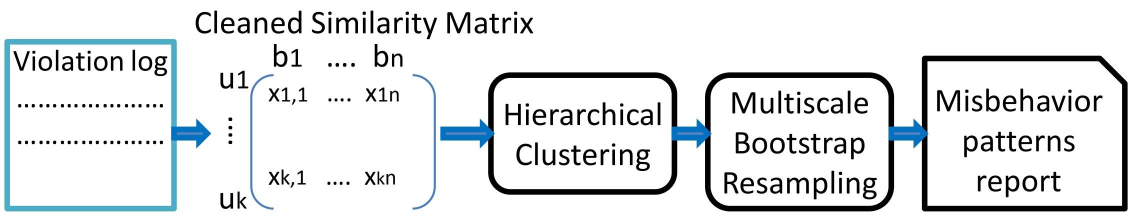
We evaluate our trust methodology in Section 5.5. In the following section, we present the *Administration Module* which is in charge of detecting policy misconfigurations.

5.4 ADMINISTRATION MODULE

An important consideration for monitoring systems is the fact that some of the suspicious behaviors may, in fact, be due to factors other than insider attacks and incompetence. For instance, if it is informally agreed that an obligation is no longer required, but the policy is not up to date, users may be ignoring that particular obligation in accordance with the informal agreement. To find the patterns of misbehavior, we incorporate clustering techniques within the administrative module. These patterns can be used by the policy administrator to review whether a particular obligation should cease to exist or to see why those employees are not performing them (e.g., the reasons could include: the obligations may no longer be necessary for the business process, users are too busy, the obligation should be assigned to other roles, etc.). In addition, during this process, users that are not fulfilling a particular obligation more often than their peers are also identified. In what follows we explain the process followed by this module, but first we provide some background on clustering algorithms.

5.4.1 Clustering Algorithms

Clustering is an unsupervised machine learning technique that aims to discover similar groups and outliers in datasets with unknown characteristics. We refer to [68] for a comprehensive review. Each observation being compared is represented by a vector that contains information about different characteristics. Clustering algorithms use a distance measure to identify how far apart the observations being clustered are. Different distance metrics exist in the literature, e.g., Euclidian, Manhattan distances. There are two types of clustering algorithms: *hierarchical* and *partitional*. *Partitional* methods require the specification of the number of clusters to be found; given this number, they output a solution with that number of clusters. In contrast, *hierarchical* algorithms do not need as input the number of clusters to be found



Cleaned Similarity Matrix
Violation log
b1 .... bn
u1
uk
x1,1 .... x1n
xk,1 .... xkn
Hierarchical Clustering
Multiscale Bootstrap Resampling
Misbehavior patterns report

Figure 14: Procedure to find the patterns of misbehavior.

and output several possible clusters. The hierarchical clustering algorithms begin by placing each observation into a separate cluster. Then, they verify the distances between all the observations and put together the two most similar ones in a new cluster. Existing methods to perform hierarchical clustering mainly vary on the way they compute the similarity between clusters; among them are *Ward, single-link* and *mean/average* methods. A detailed discussion on the differences among them can be found in [68].

The output of hierarchical clustering algorithms is a set of possible clusters, however, they do not assess the strength of the relation between the grouped observations. *Multiscale bootstrapping resampling* [105] is a methodology that allows us to overcome this downside by computing p-values for each of the clusters found by the hierarchical clustering algorithm. The methodology indicates the clusters that have high cohesion, which allows the data analyst to focus his attention in those relevant patterns.

5.4.2 Process to Find Patterns of Misbehavior

The process to find patterns of misbehavior is illustrated in Figure 14 and should be performed periodically. We use a clustering technique to detect patterns of misbehavior and outliers. We utilize hierarchical clustering, as it does not require the specification of the number of clusters to be found. This is appropriate since administrators do not know whether the users in the system have similar misbehaviors, whether they can be grouped or how many groups would result. The only parameter that needs to be specified is the distance metric to compare individuals and clusters. We use Ward hierarchical algorithm with Manhattan distance, as it finds better clusters for our purpose. We evaluate different algorithms in Section 5.5.

In order to use the algorithm, the logged information is set up in a *similarity matrix* $M_{|U| \times |\mathcal{B}|}$, which has one row for each of the users and one column for each obligation of the system. Each cell $x_{i,j}$ in the matrix contains the total number of obligations of type b_j that user u_i has violated. The information included can have as much historical information as the administrator desires. Then, the matrix is cleaned by removing users that have not misbehaved, as there is no point in trying to find patterns of misbehavior for them.

The cleaned similarity matrix is used as input for the clustering algorithm, which outputs a set of possible clusters. Then, the system performs a multiscale bootstrapping resampling that establishes which of the clusters are cohesive. *Cohesive clusters* may represent policy misconfiguration or users colluding not to perform an obligation. This information can be used by the administrator to take corrective measures. For instance, he may decide to investigate why a cluster of users is not fulfilling an obligation and if appropriate, he may remove the obligation from the policy. On the other hand, *outliers* with a high number of obligation(s) violated may represent lazy, absent users or employees that may have higher risk of becoming insider attackers. This information can be used to further monitor their performance. Figure 18a presents an example. The dendogram was generated by the clustering algorithm and it shows all the possible clusters. The multiscale bootstraping resampling method created the rectangles that show the cohesive clusters that represent different patterns of misbehavior. Note that u_{10} is an outlier; if the number of obligations violated is high, he is flagged as suspicious.

5.5 EVALUATION

In this section, we begin by evaluating our proposed trust methodology presented in Section 5.3. Then, we present the assessment of the procedure to find patterns of misbehavior presented in Section 5.4.

Evaluation of the trust methodology: We evaluated our system under different

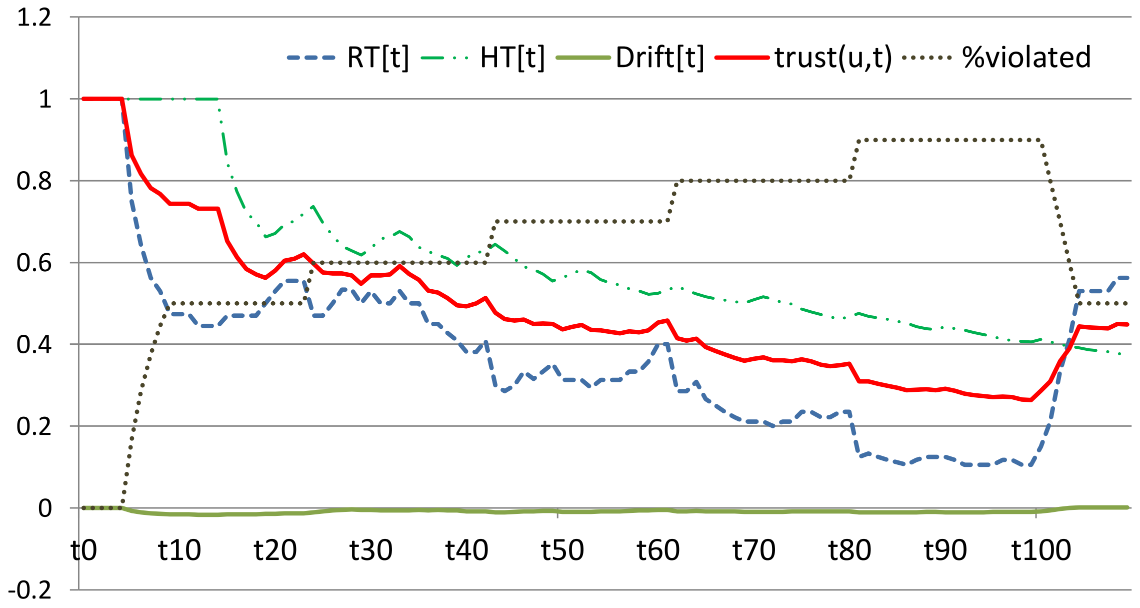


Figure 15: Evolution of trust values when the percentage of violated obligations increases, with $\alpha = 0.4$, $\gamma_1 = 0.01$, $\gamma_2 = 0.03$ and $\rho = 0.9$.

users' behaviors. We generated synthetic data to test our approach. In each iteration, a user could fulfill or violate one of 15 *a posteriori* obligations. The criticality values of the obligations were assigned using the following distribution: 10% of obligations were set to high (0.9), 60% were set to medium (0.6) and the remaining were set to low (0.3) criticality. The number of observations in each period was set to 10. Each of the points in the following experiments was found every time a new observation was generated. In our experiments, we used $\rho = 0.9$ to compute the historical trust (Definition 17), for the reasons explained in Section 5.3. Our implementation was done in java.

Misbehaving users: To verify that our methodology is able to identify when a user is misbehaving, we examined three different cases. Figure 15 presents the results for a user that initially was completely trusted $trust(u, t_0) = 1$, but later starts misbehaving, as it is shown by the percentage of violated obligations per period. As the number of violated obligations increases, the obligation trust value, $trust(u, t)$, of the user is reduced. In addition, the historical and raw trust values also decrease as the misbehavior continuous. Consider an

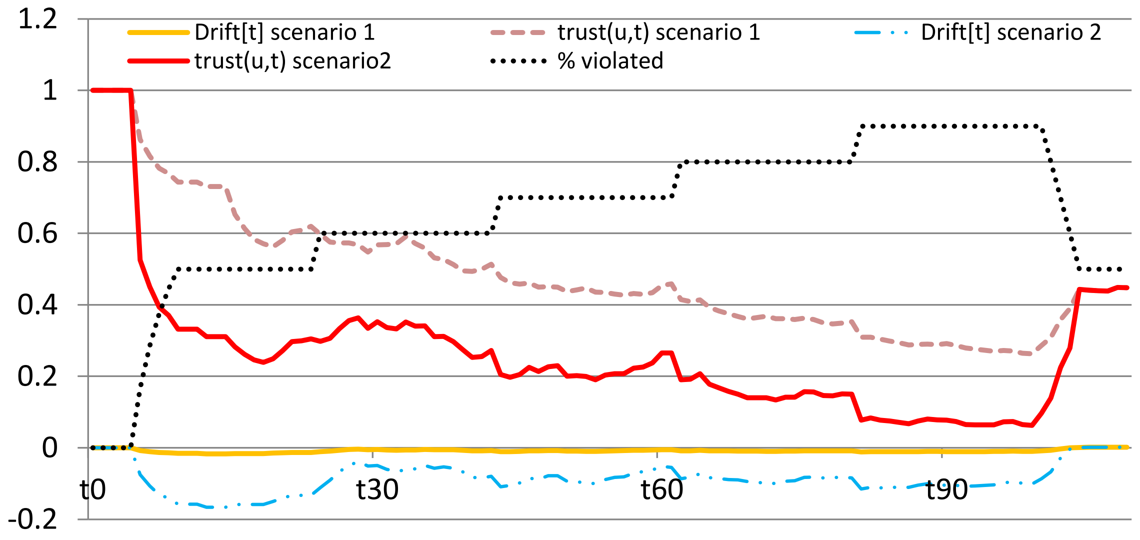


Figure 16: Trust values comparison for: *scenario 1*: $\alpha = 0.4$, $\gamma_1 = 0.01$ and $\gamma_2 = 0.03$ and *scenario 2*: $\alpha = 0.4$, $\gamma_1 = 0.01$ and $\gamma_2 = 0.3$.

obligation with a high criticality of 0.9. If the user attempts to access a permission that would require the fulfillment of that particular obligation, he would not be able to obtain the privilege after t_5. Around t_{17}, he would lose accesses that require a trust value higher than 0.6.

If the administrator desires the system to adapt faster to unfavorable changes in behavior, the weight γ_2 can be set up higher to increase the punishment for negative drifts on user's behavior. For the same user we presented in the previous experiment, Figure 16 shows how the system increases its sensitivity to negative behaviors. Scenario 1 presents a conservative γ_2 weight while scenario 2 shows the results for a bigger γ_2 value. The obligation trust value of the user decreases faster for scenario 2 than for scenario 1 resulting in a faster revocation of highly critical privileges. This is due to the amplified effect of a negative drift, which is also shown in the figure. For scenario 1, the drift is almost zero, while for scenario 2, the negative effect is substantially smaller, which according to Definition 18 results in a smaller obligation-based trust value. Hence, the larger γ_2, the faster the system adapts to negative behaviors.

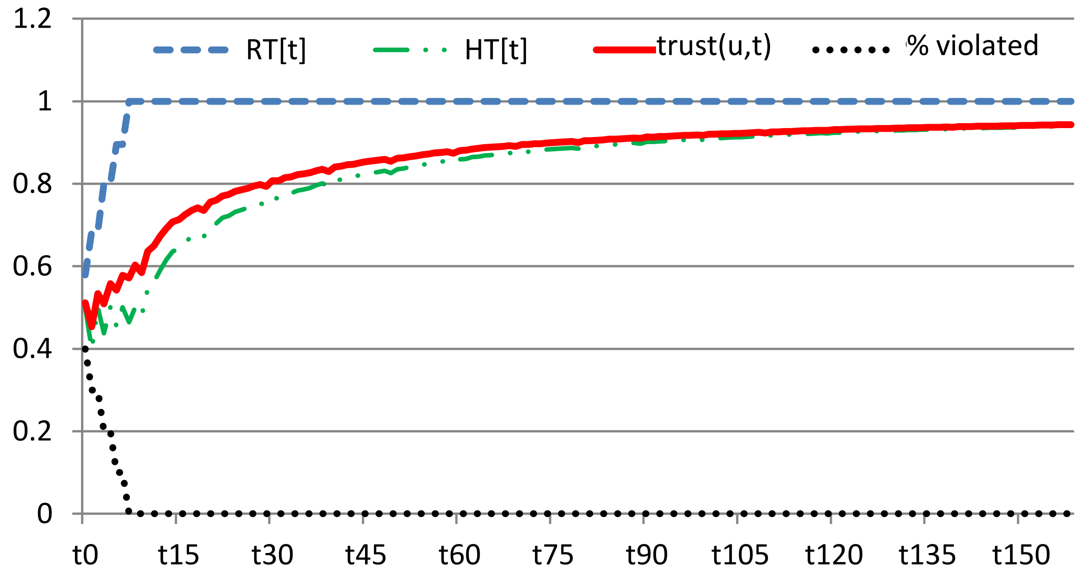
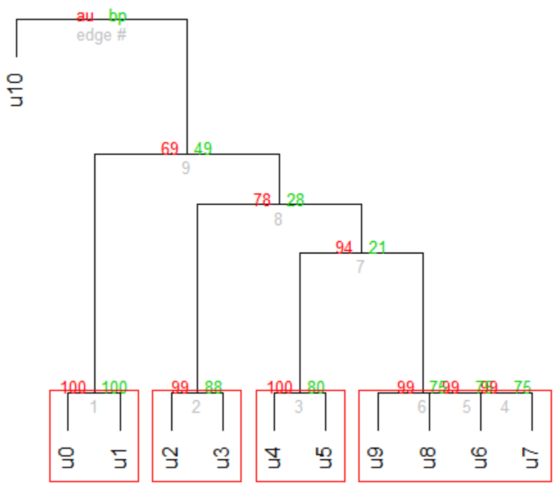


Figure 17: User redemption after having a trust value of 0.5. Parameters used: $\alpha = 0.4$, $\gamma_1 = 0.01$, $\gamma_2 = 0.03$ and $\rho = 0.9$.

Redemption: We also evaluated the results of the system when a user improves his behavior. This is relevant, as it is possible that the user was not able to fulfill his obligations due to legitimate reasons (e.g., absence caused by sickness), hence, the system should allow the user to improve his trust value based on his new behavior. At the same time, it is important that the trust increases slowly, otherwise attackers would be able to increase their trust value too fast. Figure 17 shows the results of a scenario in which the user's initial trust values are set to 0.5, but after t_7 he starts fulfilling all the assigned obligations. Since the user fulfills all the obligations (from t_7 onwards), $RT[T]$ is always equal to one and the drift is always zero. The user requires twenty periods of spotless behavior before he improves his trust to 0.8 (with the parameters of scenario 1). Because the good behavior continues, the user's trust value also continues the improvement trend.

Evaluation of the methodology to find patterns of misbehavior: We created several logs with different patterns of misbehavior, outliers and noise. A *misbehaving pattern* consists of several users not fulfilling a particular obligation, as if there was a legitimate informal agreement not to perform that obligation. *Outliers* are users who did not fulfill


(a) Ward with Manhattan

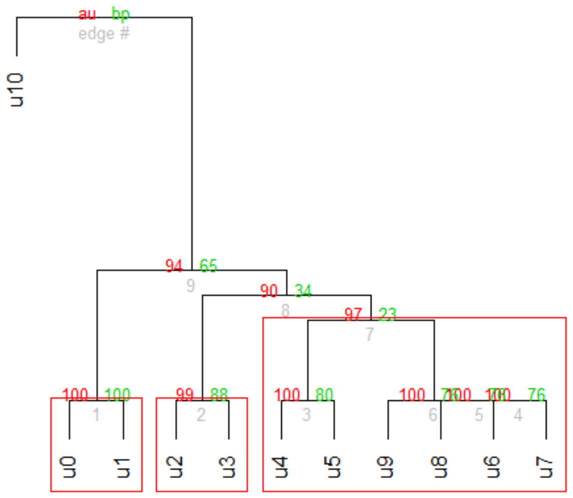

(b) Single-link with Euclidean

Figure 18: Example. The boxes in the dendongram represent cohesive clusters.

continuously a particular obligation and hence had a larger number of violations for that obligation than the average of the users. In addition, we included random *accidents* which represented obligations not fulfilled, unintentionally e.g., once someone missed a deadline. These observations can be considered as noise. The maximum number of obligations in the system was set to 15, the maximum number of users to 30 and we generated a total of 10 logs. We used *R* [93] to run cluster algorithms and the bootstrapping sampling method (with a significance level of 0.95) on the data and verified how many of the expected observations were classified correctly. We compared three hierarchical clustering algorithms: *Ward, single-link* and *mean* with two distances *Euclidean* and *Manhattan.* Since we know the existing patterns in the data tested, we can compare the solutions of the algorithms. The expected patterns in the data, are referred to as *classes*; they represent the ground truth. For example, $class_1 = \{u_0, u_1\}$ represents the users that violated obligation b_4, $class_2 = \{u_2, u_3\}$ represents the users who violated obligation b_6, $class_3 = \{u_4, u_5\}$ represents the users violating b_8 and $class_4 = \{u_6, u_7, u_8, u_9\}$ represents those violating obligation b_{12}. Figure 18 presents two solutions; one found by *Ward with Manhattan* and the other by *Single-link with Euclidean.* The rectangles around the users represent cohesive clusters. For the Ward output, the three

Table 4: Comparison between clustering algorithms

Method	Average Entropy	Average Purity
Ward Euclidean	0.28	0.68
Ward Manhattan	0.19	0.70
Average Euclidean	0.32	0.64
Average Manhattan	0.28	0.67
Single Euclidean	0.33	0.62
Single Manhattan	0.26	0.67

expected classes were found. In contrast, the Single-link algorithm created one cluster for all the elements in classes 3 and 4, failing to identify the existing misbehaving pattern. In this case, Ward with Manhattan outperformed Single-link with Euclidean.

To compare clustering algorithms *purity* and *entropy* are typically used [121]. *Entropy* is a function of the distribution of classes in the resulting clusters and *purity* is a function of the relative size of the largest class in the resulting clusters. The details of how to calculate these metrics are provided in Appendix . Both entropy and purity are in the interval [0,1]. Solutions with higher purity are preferred, while solutions with small entropies are preferred.

Table 4 presents the comparisons among the algorithms. Ward in combination with Manhattan distance provides the most reliable results according to both of the metrics used, as clusters found most of the time represented the existing classes. The worst results were found for single-link in combination with Euclidean distance. In addition, when the algorithms used Euclidean distance to measure difference among users, the results were consistently worse than when they used Manhattan distance. The empirical validation shows that the best option is to use Ward with Manhattan.

5.6 CHAPTER SUMMARY

In this chapter, we presented an access control framework to mitigate the risk exposure caused by *a posteriori* obligations. We incorporated the criticality value of violating an obligation into its definition to monitor users' violation patterns, prevent sabotage threats and unintentional damage. Additionally, we presented an obligation-based trust methodology that is resilient to naïve and strategic users. This methodology can be integrated into any access control model with *a posteriori* obligations. Through experimental evaluations, we showed that our methodology can effectively identify suspicious behavior as well as allow users to recover their trust values slowly. We also showed that based on previous work on psychological precursors *a posteriori* obligations can be used to identify suspicious users. Finally, we provided an approach to identify patterns of misbehavior, suspicious users and policy misconfigurations.

In the next chapter, we present our third framework, G-SIR, which incorporates geo-social information into the access control decision making process to deter insider threats.

6.0 AN INSIDER ATTACK RESILIENT GEO-SOCIAL ACCESS CONTROL SYSTEM

In this chapter we present our proposed *Geo-Social Insider Threat Resilient Access Control Framework* (G-SIR). First, we present the notions of spatial scopes and social predicates in Section 6.1. Then, in Section 6.2, we present an overview of the proposed framework. The detailed specification of our proposed G-SIR model is presented in Section 6.3. The risk management procedure is presented in 6.4. The proposed enforcement algorithm is presented in Section 6.5. Finally, we conclude the chapter in Section 6.6 where we present the system evaluation.

6.1 SOCIAL PREDICATES AND SPATIAL SCOPES

G-SIR makes use of the notions of *social predicates* and *spatial scopes* introduced in Geo-Social RBAC model [14]. They are defined as follows.

Social Predicates: A social graph can be represented as $\mathcal{G} = \langle V, E \rangle$ where V is a set of vertices that represent users and E is a set of edges that represent the existence of a social relation between users[1]. These edges may be also labeled to refine further the types of relationships between users. Let W be a set of social relation labels (e.g., nanny, spouse, etc.) that may be organized in a hierarchy. $W_{(i,j)}$ represents the set of labels of edge (i, j), for example, $W_{(i,j)}$=\{*nanny*, *schoolmate*\} shows that user i is the nanny and schoolmate of user j.

[1]Our model may also be used with graphs built from available information such as tweets, retweets, among other data.

Additionally, there is a set of social functions to evaluate the social relations between users. Examples of these functions include $areFriends(v_i, v_j)$, $haveSocialRelation(label, v_i, v_j)$, $socialDistanceLessEqualTo(v_i,\ v_j,\ k)$, $isSuperior(v_i,\ v_j)$, $haveCommonNeighbor(v_i,\ v_j)$, $areIn$-$Clique(v_i,\ v_j)$, $formAClique(v_i,\ V' \subseteq V)$, among others. Functions $belongsToCommunity(u,$ $comm)$, and $assignedToRole(u,\ r)$ are useful for our framework and are defined in Table 5. Let F be the set of social functions such as those mentioned above. We define a *Social Predicate* $\mathcal{S}$ as[2] $\mathcal{S} ::= \mathcal{S} \wedge \mathcal{S} | \mathcal{S} \vee \mathcal{S} | f | \neg f$, where $f \in F$.

To specify *social predicates* we use u_r to denote the requesting user and $u?$ to denote a user in the vicinity that is instantiated at the time of evaluation of the policy.

Spatial Scopes: A *Spatial Scope*, SC, defines a place of interest. It is defined as $SC = \langle h, \ell \rangle$, where h is a feature and ℓ is a location function. A feature is a place of interest in the space, e.g., room 410, x-y coordinate or hallway. The geometry of these features are defined according to the Open GeoSpatial consortium geometric model [1]. Function ℓ evaluates where with respect to $feature$ the user needs to be located. For example, $SC = \langle room420, in \rangle$, defines as spatial scope being inside *room420* and $SC = \langle radiusAround(u, 5feet), in \rangle$, defines a circle with a radius 5 feet around the current position of user u. Function ℓ can also be *overlap*, *touch*, *cross*, *in*, *contains*, *equal*, and *disjoint* [1], and may also be defined using more refined proximity functions as the ones defined by Gupta *et. al* in [63].

6.2 OVERVIEW OF THE PROPOSED G-SIR

At the core of the proposed G-SIR framework there is an access control policy specification and enforcement mechanism designed to leverage users' geo-social behavior. The access control component captures current and historic geo-social interactions to determine whether an access should be granted or denied. Our framework extends RBAC; hence, users need to fulfill constraints that are assigned to roles they play in an organization. A role may be subject to the following constraints.

- *Spatial scope:* A role may have a spatial scope that defines a set of locations where it

[2]For simplicity, parenthesis are omitted to avoid distracting readers from the main issues

can be activated by users assigned to it.

- *Geo-Social Contracts:* These constraints indicate places that users assigned to the constrained role cannot visit and people they cannot frequently meet.
- *Vicinity constraints:* These constraints impose restrictions on people that may or may not be at a certain distance from the requester at the time of an access. There are two types of vicinity constraints: *inhibiting* and *enabling constraints*.

  Inhibiting constraints specify that a requested permission needs to be denied when certain inhibiting users are in the vicinity. They are designed to avoid potential proximity attacks, such as shoulder surfing attacks. For this, a spatial scope where inhibitors cannot be located is defined.

  Enabling constraints are designed to verify the validity of an access request by leveraging the trust on other users in the vicinity. These constraints specify who and how many people should be in a spatial scope of interest. To enforce them, it is important to ensure that the enablers and the requester are not colluding to prevent insider attacks. We refer to this as *collusion-free* enforcement.
- *Geo-social trace based constraints:* These constraints require a user to follow a particular geo-social path before he can be authorized to access a particular resource. They are often useful to ensure that users do not access a resource without proper previous interactions.
- *Geo-social obligations:* These are geo-social actions that users need to fulfill after they have been granted an access.

The proposed constraints are useful in two ways. First, they help capture inappropriate geo-social context and subsequently deny accesses that violate the access control policy. Secondly, monitoring the fulfillment of these constraints provides a way to identify users' whose geo-social behavior is frequently questionable and outside of the expected patterns.

When users violate their geo-social contracts, do not fulfill their obligations or traces, G-SIR flags them as suspicious. Because some of the constraints may be more important than others, their violation has a criticality value. The observations of suspicious geo-social behavior are used to obtain the likelihood of insider attacks and, ultimately, to determine the risk exposure of granting an access.

Figure 19 presents the architecture of G-SIR. All monitoring and likelihood computations described take place in the *Monitoring, Context and Inference Module*. The *Context Module* determines the context of a user, which includes information such as the current device used by the user, type of connection used, etc.

The *Access Control Module* is in charge of making the access control decisions. To determine if an access request should be granted, all applicable geo-social constraints are verified. Additionally, this module verifies if the risk exposure of granting access to a set of requested permissions is tolerable to allow the access. To manage the risk exposure, at the time of policy specification, the system administrator should perform a utility elicitation process. During this process, described in detail in Section 6.4, the possible costs of misuse of granting a malicious access, the cost of denying a non-malicious access and gain of allowing a non-malicious access are analyzed. Through this analysis, a threshold that determines the maximum tolerable probability of attack is found. If the probability of attack is too high according to the risk management procedure, the access is denied. Otherwise it is granted.

The steps performed by the *Access Control Module* are as follows. Each access request, $\mathbf{Q}_u = \langle u, P' \rangle$, where u denotes the user requesting permission set P', is received by the *Policy Enforcement Point* (PEP). Then, it forwards them to the *Policy Decision Point* (PDP) which evaluates the policy stored in *Policy Enforcement Point* (PIP). An access request is granted by the PDP if all of the following conditions are satisfied; otherwise it is denied:

- User u is assigned to a role set, R', required to obtain the requested permissions P'. Additionally, the current location of u allows the activation of R' and does not violate any of u's geo-social contracts.
- All trace-based constraints associated with R' are fulfilled by u.
- No inhibitors are located in the vicinity.
- There are enough enablers, not suspected of colluding, to fulfill all enabling constraints associated with R' and the current locations of the possible enablers do not violate any of their geo-social contracts.
- Finally, the risk management procedure that considers the historic geo-social behavior of the user permits the access.

In the next section, we present the proposed access control model.

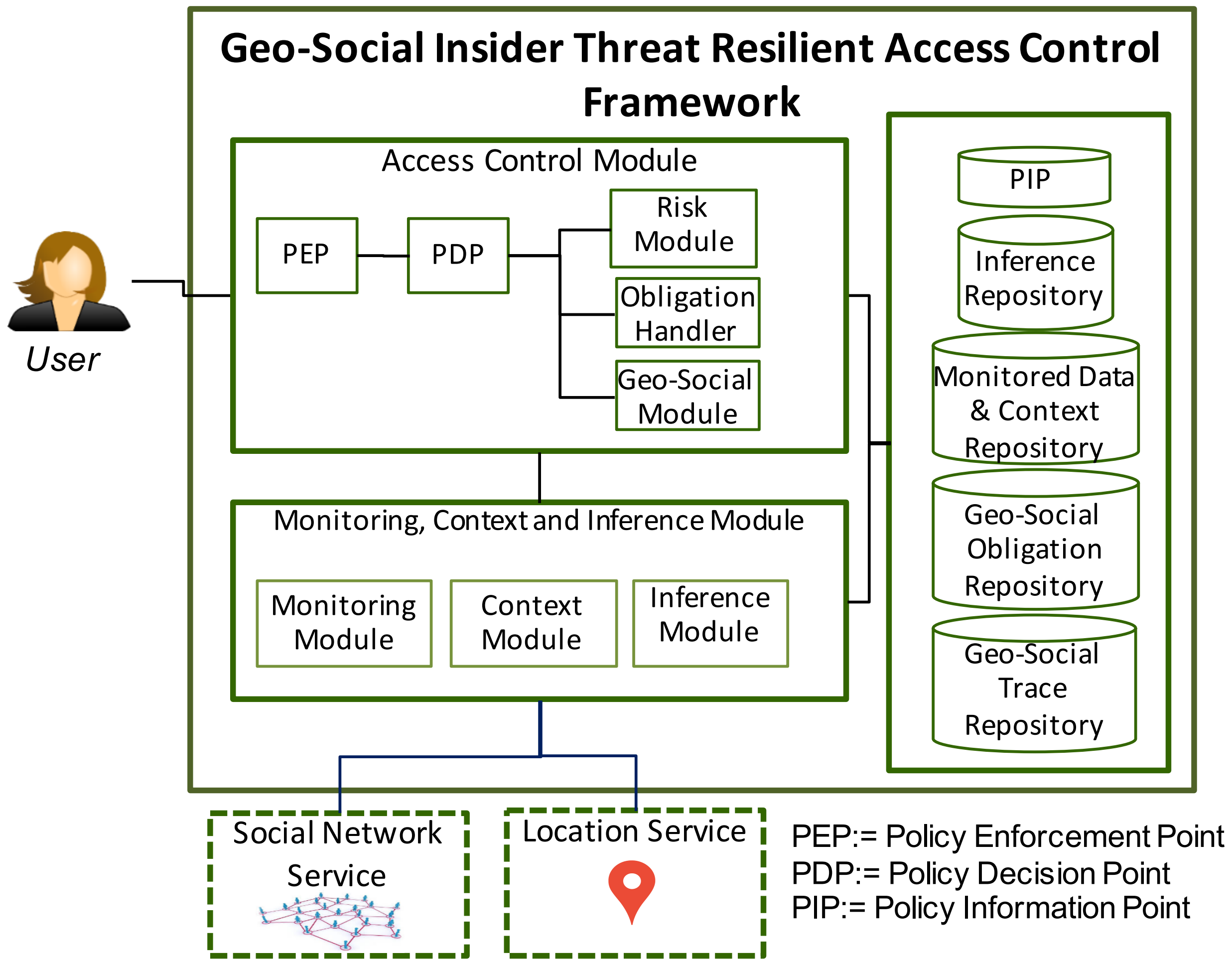

Geo-Social Insider Threat Resilient Access Control Framework
Access Control Module
PEP
PDP
Risk Module
Obligation Handler
Geo-Social Module
Monitoring, Context and Inference Module
Monitoring Module
Context Module
Inference Module
PIP
Inference Repository
Monitored Data & Context Repository
Geo-Social Obligation Repository
Geo-Social Trace Repository
User
Social Network Service
Location Service
PEP:= Policy Enforcement Point
PDP:= Policy Decision Point
PIP:= Policy Information Point

Figure 19: Overview of the proposed G-SIR framework

6.3 G-SIR ACCESS CONTROL MODEL

The G-SIR access control model consists of sets of roles R, users U, actions A, objects O and sessions S. Permissions are defined as $P = A \times O$. Users are assigned to roles, and roles are assigned permissions.

Table 5: Function specifications for G-SIR.

Function	**Meaning**
$assigned(u \in U)$	Returns the set of roles that u is assigned to.
$P_{au}(r \in R)$	Returns the set of permissions assigned to r.
$P_{au}(R' \subseteq R)$	Returns the set of permissions assigned to all roles in R'.
$validLocation(u, r)$	Returns true if the current location of u satisfies the spatial scope of $r.SC$.
$vicinity(SC)$	Returns a set of users located in the place specified by spatial scope SC.
$PrCollusion(U_c \subseteq U)$	Function that determines the probability that users in U_c are colluding.
$belongsToCommunity(u \in U,\ comm)$	Given a user u and a community name $comm$, returns true if the user is part of $comm$.
$assignedToRole(u \in U,\ r \in R)$	Given a user u and a role r, returns true if u is assigned to r.
$fulfillSocialPredicate(u_r, u_c, \mathcal{S})$	Returns true if users u_c and u_r fulfill social predicate $\mathcal{S}$.
$fulfillContracts(u \in U)$	Returns true if user $u \subseteq U$ currently satisfies *all* his contracts. It evaluates the union of all contracts assigned to roles in $assigned(u)$.
$inhibitors(u_r \in U, r \in R)$	Given a requester u_r and a role r, returns a set of users that are classified as inhibitors according to the inhibiting constraints $r.\mathcal{I}$.
$enablers(u_r \in U, r \in R)$	Given a requester u_r and a role r, returns a set $U_e \subseteq U$, if it exists, that satisfies all enabling constraints $r.\mathcal{E}$ according to Definition 26. Otherwise it returns $\emptyset$. If $r.\mathcal{E} = \emptyset$, it returns $\emptyset$.
$completeTraces(r \in R, u \in U)$	Returns true if user u has completed traces $r.\mathcal{W}$.
$traceContains(w \in \mathcal{W}, node)$	Returns true if trace w contains $node$ as part of its spatial scope $w.SC$.
$disjoint(SC_i, SC_j)$	Given two spatial scopes SC_i and SC_j, returns true if the SC_i is disjoint in SC_j.
$fulfillO(u \in U, r \in R)$	Returns true if user u satisfies obligations $r.\mathcal{B}$ and *all* his geo-social contracts.

Let $\mathbf{X}$ be a set of contexts dynamically associated with users. The context of user u is denoted as $\mathcal{X}_u$. Let $\mathbf{E}$ be a set of enabling constraints and $\mathbf{I}$ be a set of inhibiting constraints. Additionally, let $\mathbf{GC}$ and $\mathbf{B}$ be the sets of geo-social contracts and geo-social obligations, respectively. Finally, let $\mathbf{W}$ be a set of geo-social traces. All these constraints are formally defined later.

Definition 22. *A* role $r \in R$ *in G-SIR access control model is associated with a constraint vector* $CV_R = \langle SC, \mathcal{E}, \mathcal{W}, \mathcal{GC}, \mathcal{B} \rangle$ *where:*

- SC *is the spatial scope of a role (places where the role can be activated).*
- $\mathcal{E} \subseteq \mathbf{E}$ *and* $\mathcal{I} \subseteq \mathbf{I}$ *represent the constraints enforced over the users in the vicinity, where* $\mathcal{E}$ *defines the required enablers, and* $\mathcal{I}$ *defines inhibitors.*
- $\mathcal{W} \subseteq \mathbf{W}$ *is a set of geo-social trace constraints.*
- $\mathcal{GC} \subseteq \mathbf{GC}$ *is a set of geo-social contracts.*
- $\mathcal{B} \subseteq \mathbf{B}$ *is a set of geo-social obligations.*

To refer to a constraint of a role, we use the dot notation, e.g., $r.SC$ returns the spatial scope of role r and $r.\mathcal{I}$ returns its inhibiting constraint. In section 6.3.6, we specify how a role can be activated in a session to exercise the permissions associated with it. We make use of the functions presented in Table 5. We also use the dot notation to refer to components of tuples. We now define the constraints that can be assigned to roles.

6.3.1 Geo-social Contracts

Geo-social contracts are used to establish acceptable and unacceptable geo-social behavior for different roles. Geo-social contracts are assigned to a user when he is assigned to a role. These contracts need to be constantly fulfilled.

Definition 23. *A* Geo-Social Contract $gc \in \mathbf{GC}$ *is defined as* $gc = \langle \omega, \varphi \rangle$ *where*

- $\omega = \langle SC, \mathcal{S} \rangle$*, here* SC *represents a spatial scope (place that users subject to* gc *are not allowed to visit), and* $\mathcal{S}$ *is a social predicate that defines undesirable acquaintances. When a component in* ω *is set to* $\perp$*, (e.g.,* $\omega.SC = \perp$*), it indicates that it is not considered during the enforcement.*

- $0 \leq \varphi \leq 1$ *represents how critical it is for the organization if a user violates the contract. Here,* $\varphi = 1$ *means that it is very critical while* $\varphi = 0$ *means not critical at all.*

If user u is assigned to a role set $R_u \subseteq R$, to be allowed to activate any role in R_u, he needs to fulfill all geo-social contracts associated with each role in R_u.

Example 6. *(a) Consider a user Bob who is assigned to role* secretary*; by being assigned to this role, he cannot access a laboratory where highly reactive chemicals are located because he is not trained to deal with dangerous chemicals. If Bob accesses the lab, there is an inherent risk of mishandling substances that may lead to accidents and loss of lives and intellectual property. For this reason, a violation of this contract will result in a high risk for the organization. This constraint can be expressed as follows:* $gc_1 = \langle\langle\langle chemicalLab, in\rangle, \perp\rangle, 0.9\rangle$. *(b) Consider a consulting firm that may have projects from multiple competing companies, say* X *and* Y*. The consulting firm needs to ensure that the projects are completely compartmentalized to be able to offer a quality consulting service. Besides enforcing separation of duty –where no user can be assigned both roles, namely consultant for* X, r_x, *and consultant for* Y, r_y *– it is desirable that people belonging to conflicting projects are not together to avoid leakage of information. Contractors that have multiple clients often require this type of control. These constraints can be expressed as follows:* $gc_2 = \langle\langle\perp, assignedToRole(u?, r_x)\rangle, 0.5\rangle$ *and* $gc_3 = \langle\langle\perp, assignedToRole(u?, r_y)\rangle, 0.5\rangle$. gc_2 *is associated with role* r_y *and* gc_3 *is associated with role* r_x.

As the previous scenarios show, not all contracts are the same in terms of risk exposure. An untrained person entering a lab that has a lot of volatile chemicals poses higher risk compared to the same person entering into a meeting room reserved for a team working in a classified advertisement (e.g., an untrained individual may cause a serious accident). Hence, $gc_1.\varphi > gc_2.\varphi$.

6.3.2 Vicinity Constraints

Inhibiting and enabling constraints are designed to classify users in the vicinity as enablers, inhibitors or neutral.

Definition 24. *An* Inhibiting Constraint $ci \in \mathbf{I}$ *is defined as tuple* $\langle X, SC, \mathcal{S}, \alpha\rangle$ *where*

- $X \subseteq \mathbf{X}$ *is a subset of contexts where the inhibiting constraint is applicable,*
- SC *defines the spatial scope where the inhibiting constraint is evaluated,*
- $\mathcal{S}$ *defines the predicate used to classify users in the vicinity as inhibitors and*
- α *is a threshold to determine the minimum level of confidence needed to decide if a user should be made part of the inhibiting group.*

We say that if there is a set of one or more users U_{ci} in location $ci.SC$, who fulfill social predicate $ci.\mathcal{S}$ with a minimum confidence level of α, the constraint is not satisfied and the access should be denied to prevent information leakage.

At the policy evaluation time, G-SIR verifies if the requester's context is one of the context specified in X. If it is, the inhibiting constraint is evaluated otherwise it is ignored. This helps specify policies where the device the user is utilizing may influence the size of the spatial scope evaluated as illustrated in the following example.

Example 7. *Assume* smartphone*,* laptop *and* presenter *are context of interest. Consider requesting user u_r who is assigned to role r_1 with inhibiting constraints $r_1.\mathcal{I} = \{ci_1, ci_2\}$, where* $ci_1 = \langle\{laptop, smartphone\}, \langle radiusAround(u_r, 5feet), in\rangle, belongToCommunity(u?, BadGuys), 0.95\rangle$ *and* $ci_2 = \langle\{presenter\}, \langle conferenceRoom, in\rangle, belongToCommunity(u?, BadGuys), 0.95\rangle$.

When u_r is using a laptop or smartphone, ci_1 is evaluated to verify the presence of inhibitors within a 5feet radius. If u_r is using a presenter (e.g., u_r is making a presentation), ci_2 is evaluated to verify that no inhibitors are present in the conference room. In both ci_1 and ci_2, users in the vicinity are classified as inhibitors if they belong to the community BadGuys with a confidence level of 0.95 or more.

Definition 25. *An* Enabling Constraint *$ce \in \mathbf{E}$ is defined as a tuple $\langle SC, k, \mathcal{S}, \tau_c\rangle$ such that SC is a spatial scope where k users who fulfill social predicate $\mathcal{S}$ with respect to the requester need to be located, and $0 \leq \tau_c \leq 1$, is a threshold that defines the maximum tolerance for colluding users.*

Here, $ce.\tau_c$ is the maximum acceptable probability of collusion and should be specified based on the risk of an access. A larger $ce.\tau_c$ reflects more tolerance to collusion behavior. In fact, if $ce.\tau_c = 1$, the collusion indicators are not considered at all. In contrast, when $ce.\tau_c = 0$ any suspicion of collusion results in invalidating a set of enablers.

Threshold $ce.\tau_c$ provides a way to determine when a set of potential enablers cannot be trusted. It is compared with the value obtained by function *PrCollusion*. Consider a candidate set of enablers U_e, if $PrCollusion(U_e) > ce.\tau_c$, the candidate enablers are rendered untrustworthy.

$ce.\mathcal{S}$ may be evaluated based on uncertain information. For example, a social graph may be evaluated to identify if users belong to dangerous communities through algorithms such as those presented in [78, 5, 40]. These algorithms output a set of communities and a confidence level of the result. $ce.\alpha$ determines the minimum confidence level required to classify a user as part of a community. In contrast, when $ce.\mathcal{S}$ is evaluated based on information that is well-established, $ce.\alpha$ can be set to one.

Example 8. *Consider role r_2 with a set of enabling constraints $r_2.\mathcal{E} = \{ce_1\}$. Enabling constraint ce_1 is defined as: $\langle\langle conferenceRoom, in\rangle, 4, areFriends(u?, u_r), 0.8\rangle$. ce_1 requires four users who are friends of the requester to be in the conference room and for them to be non-colluding with a probability of 0.8 or more.*

Definition 26. *Given a requester u_r, an enabling constraint $ce = \langle SC, k, \mathcal{S}, \tau_c\rangle$ is said to be* satisfied *if and only if there exists a set of enablers U_e such that $\forall\ u_e \in U_e$:*

1. $u_e \in vicinity(ce.SC)$.
2. $fulfillSocialPredicate(u_r, u_e, ce.\mathcal{S})$
3. $PrCollusion(U_e \cup u_r) \leq ce.\tau_c$
4. $fulfillContracts(u_e)$
5. $|U_e| \geq ce.k$

In the previous definition, the risk of including invalid enablers is controlled in two ways. *i)* by verifying that the probability of collusion between the set of enablers is less than the specified confidence threshold and *ii)* by verifying that none of the enablers is violating any of his contracts. This mitigates potential social engineering attacks where an enabler is tricked into going to the required spatial scope $ce.SC$ to satisfy enabling constraint ce. It similarly thwarts attacks where the requester and enablers probe the system to see what accesses they can obtain.

Conflict Resolution: Because inhibiting and enabling constraints are evaluated dynamically –based on who is located in the vicinity at the time of the access request, it is possible that one or more users in the vicinity may be classified as both inhibitor and enabler. We call this a vicinity conflict. It arises when for a given role, $inhibitors(u_r, r) \cap enablers(u_r, r) \neq \emptyset$. For $ce \in r.\mathcal{E}$ and $ci \in r.\mathcal{I}$, recall that $ce.\mathcal{S}$ specifies social relations of the users, whereas $ci.\mathcal{S}$ specifies users in the vicinity suspected of belonging to dangerous communities for an access. Hence, a user may be related to another and at the same time be suspected of participating in a non-desirable community according to ci. This may occur for instance, when a user is suspected of being a spy. By design, this conflict is resolved in G-SIR through *deny overrides*: if a user is classified as inhibitor, the access request is denied.

6.3.3 Geo-Social Obligations

Geo-social obligations establish that after activating a role, the requester needs to visit or cannot visit a particular place or interact with people within a predefined period of time.

Definition 27. *A* Geo-Social Obligation $b \in \mathbf{B}$ *is defined as* $\langle dir, D, \varphi \rangle$ *where*

- *dir is the directive that users subject to b need to fulfill.* $dir \in \{\langle +meet, \mathcal{S}\rangle, \langle +visit, SC\rangle, \langle -meet, \mathcal{S}\rangle, \langle -visit, SC\rangle\}$. +meet, *means that the user should meet a targeted person or group as defined by social predicate* $\mathcal{S}$, *while* -meet *means that the user should not meet the person or population. Similarly,* +visit *means a user needs to visit spatial scope SC and* -visit *that he cannot visit it.*
- *D is a time duration that specifies how much time a user has to fulfill the obligation after the obligation is triggered and assigned to him.*
- $0 \leq \varphi \leq 1$ *is a value that represents how critical it is for the organization if a user violates the obligation. Here,* $\varphi = 1$ *means that it is very critical and* $\varphi = 0$ *means it is not critical at all.*

At the time of activation, our framework instantiates each triggered obligation and creates a record to monitor its state. Suppose user u activates role r, which has associated with it obligation $b \in r.\mathcal{B}$. The framework creates a record that contains the user who triggered the obligation, the obligation triggered, b, the time t when the activation took place, and the state of the obligation, which can be *pending*, *fulfilled* or *violated*. The obligation should be fulfilled within the interval $[t,\ t+b.D]$. The obligation's state is *pending* when user u has not fulfilled it and the deadline has not passed. The state changes to *fulfilled* if u successfully fulfills b and to *violated* if the user does not complete the required condition before $b.D$ elapses.

Example 9. *After activating a role, r, users may not enter the server room where the tenant's machines are stored and cannot meet people associated with community Y.* $b_1 = \langle\langle -visit, \langle serverRoom, in\rangle\rangle, 1month, 0.7\rangle$ *and* $b_2 = \langle\langle -meet, belongsToCommunity(u?, Y)\rangle,$ $1year, 0.5\rangle$. *And* $r.\mathcal{B} = \{b_1, b_2\}$.

6.3.4 Geo-Social Trace Constraints

Geo-social traces specify the locations and social interactions that are required before activating a role. When a user wants to activate a geo-social role, his traces are evaluated to see if they match the expected ones. If they do not match, the access request is denied.

Definition 28. *A* Geo-Social Trace Constraint $w \in \mathbf{W}$ *is a tuple* $w = \langle lst, D, \varphi\rangle$ *where*

- $lst = \langle\langle SC_1, \mathcal{S}_1\rangle_1, ...\langle SC_n, \mathcal{S}_n\rangle_n\rangle$ *is a list of places and/or people that the requester needs to visit and/or meet.* SC_i *represents a spatial scope and* $\mathcal{S}_i$ *a social predicate that defines people that the requester needs to meet. When* SC_i *or* $\mathcal{S}_i$ *is set to* $\perp$*, it indicates that that component needs no consideration.*
- D *is the duration that defines how long ago with respect to the current time in the recent past the trace should have been satisfied.*
- $0 \leq \varphi \leq 1$ *is the criticality associated with not completing the trace as expected.*

In the previous definition $w.D$ specifies that only recent traces are relevant. If a user is requesting access to a role that requires the fulfillment of $w.D$ at time t, the user should have completed the trace within $[t - w.D, t]$.

Recall that a single role may have one or more geo-social trace constraints; for a role r the set of geo-social traces is denoted as $r.\mathcal{W}$. We use function *completeTraces(r, u)* to verify if u's traces satisfy all the geo-social trace constraints $w \in r.\mathcal{W}$ associated with r.

Example 10. *Consider a medical doctor who is required to go to the Sanitizing Facility before entering into the Neo-natal Unit where new babies are born. This constraint can be expressed as* $w_1 = (\langle\langle \textit{Sanitizing Facility}, \textit{in}\rangle, \perp\rangle, \textit{15minutes}, 0.8)$*, which is a trace constraint that requires the doctor to go to the Sanitizing Facility before being able to activate the role that allows him to enter into the Neo-natal Unit. If the user tries to gain access to a new-born unit without passing through the Sanitizing Facility, the impact of his actions may be severe because of the germs that he may be bringing to the newly born babies who are especially susceptible to infectious diseases. Hence, the criticality of the obligation is large,* $w_1.\varphi = 0.8$*. At the verification time, say t, the system verifies that the requester completed the trace within the past 15 minutes. If they do not, completeTraces(r, u) returns false and the role cannot be activated.*

6.3.5 Well-Formed Policy

For G-SIR to work properly, it is necessary to ensure that the policy specification is consistent. Contracts are rules that forbid some interactions and movements; if they are violated access is denied. Hence, they should not conflict with any of the other constraints. Additionally, users should not be subjected to contradictory constraints. Therefore, it is necessary to ensure the policy is *well-formed*.

Definition 29. *A policy is said to be* well-formed iff $\forall u \in U, r_i, r_j \in \textit{assigned}(u)$*:*

1. $\nexists\ gc \in r_i.\mathcal{GC}\ :\ \textit{disjoint}(gc.SC,\ r_j.SC)$
2. $\nexists\ b = \langle\langle +\textit{visit}, SC\rangle, D, \varphi\rangle \in r_i.\mathcal{B},\ gc \in r_j.\mathcal{GC}\ :$
   $\textit{disjoint}(b.\textit{dir}.SC,\ gc.SC)$
3. $\nexists\ b \in r_i.\mathcal{B},\ gc \in r_j.\mathcal{GC}\ :\ b.\mathcal{S} = gc.\mathcal{S}$
4. $\nexists\ w \in r_i.\mathcal{W},\ gc \in r_j.\mathcal{GC}\ :\ \textit{traceContains}(w, r_j.SC) \wedge \nexists \langle SC_i, \mathcal{S}_i\rangle \in w.\textit{lst}\ :\ \mathcal{S}_i = gc.\mathcal{S}$

In the previous definition, condition 1 states that no user should be assigned to a role that requires him to go to a place to obtain certain privileges while at the same time prohibiting him from going to that place according to his social contracts. Conditions 2 and 3 state that a user should not be asked to avoid places or people, while at the same time, he is required to visit and/or meet them to fulfill their obligations. Condition 4 states that contracts should not conflict with traces that the user needs to fulfill to gain access.

Whenever a new role is created or a user is assigned to a role, these properties should be verified. If one or more conditions in Definition 29 is not satisfied, it is necessary to re-evaluate the assignment and/or policy.

6.3.6 Role Activation

With all the geo-social constraints specified, we now define how to make access control decisions in G-SIR.

Definition 30. *A role r with constraint vector $CV_r = \langle SC, \mathcal{E}, \mathcal{I}, \mathcal{W}, \mathcal{GC} \rangle$ is said to be* fulfilled for user u_r, $fulfilled(u_r, r)$, *iff the following conditions are satisfied:*

1. $r \in assigned(u_r)$
2. $validLocation(u_r, r)$
3. $completeTraces(r, u_r)$
4. $fulfillContracts(u_r)$
5. $inhibitors(u_r, r.\mathcal{I}) = \emptyset$
6. *If* $r.\mathcal{E} \neq \emptyset$*, then* $enablers(u_r, r) \neq \emptyset$

Otherwise r is not-fulfilled *for u_r.*

We now define how the system decides to grant or deny an access request $\mathbf{Q}_u$.

Definition 31. *An access request $\mathbf{Q}_u = \langle u_r, P' \rangle$ is granted under context $\mathcal{X}_u$,* if and only if *there exists a set of roles $R' \subseteq R$ such that all of the following conditions are fulfilled:*

1. $\bigcup_{r \in R'} P_{au}(r) \supseteq P'$ *(Roles in R' provide the requested permissions),*
2. $\forall\ r \in R' :\ fulfilled(u_r, r)$ *(Definition 30)*
3. $RiskMan(\mathbf{Q}_u, \mathcal{X}_{u_r}) = true$

The last condition specifies that for $\mathbf{Q}_u$ to be granted, its associated risk should be acceptable according to $RiskMan$. In the next section, we present the methodology used to obtain $RiskMan$.

6.4 G-SIR RISK MANAGEMENT

In this section, we present how to compute *RiskMan*. Because *utility theory* has been recognized as a useful methodology to make decisions under uncertainty [39], we utilize it to formulate our decision-making process. A utility value represents the preferences of a decision maker. It is often useful to think of utility as a measure of *satisfaction*. Hence, a higher utility indicates a higher preference for an outcome and in combination, utility values reflect the preferred order of different outcomes. As it is customary, we define the utility value as a number between 0 to 100.

Consider an access request $\mathbf{Q}_u = \langle u_r, P' \rangle$ and let $R' \subseteq R$ be a set of roles that could satisfy the access request for u_r. We denote by $\mathtt{A}$ the uncertain event of $\mathbf{Q}_u$ being issued to compromise the system (an attack) and $\bar{\mathtt{A}}$ the complementary event ($\mathbf{Q}_u$ is a non-malicious request). We denote the probability of event $\mathtt{A}$ (attack) as q, hence the probability of event $\bar{\mathtt{A}}$ (no-attack) is $(1-q)$. Similarly, let $\mathtt{G}$ represent allowing access and $\bar{\mathtt{G}}$ represents the decision to deny access.

The utility depends on the context of the user $\mathcal{X}_u$ and the permissions that u_r would obtain through R'[3]. Because we are interested in preventing insider attacks, the following analysis assumes that the utilities are defined to reflect the interests of the organization implementing G-SIR. There are four possible outcomes of granting or denying $\mathbf{Q}_u$. Hence, there are four utilities of interest; these are:

1. $\mathtt{U}^{R',\mathcal{X}_u}_{\bar{\mathtt{G}}/\mathtt{A}}$ represents the utility of denying access to roles R' under context $\mathcal{X}_u$ given that the access request is an attack,
2. $\mathtt{U}^{R',\mathcal{X}_u}_{\bar{\mathtt{G}}/\bar{\mathtt{A}}}$ represents the utility of denying access to R' under context $\mathcal{X}_u$ given that the access request is not an attack,
3. $\mathtt{U}^{R',\mathcal{X}_u}_{\mathtt{G}/\mathtt{A}}$ represents the utility of granting access to R' under context $\mathcal{X}_u$ given that the access request is an attack, and

[3]It is necessary to consider the permissions granted by R' rather than the permissions requested P' to analyze the risk exposure for the following reason. In RBAC, permissions cannot be acquired individually; they need to be acquired through the activation of roles. That is why in Definition 31 we allow for extra permissions to be granted: $P_{au}(R')$ may have additional permissions that are not in P'. Hence, the risk exposure depends on all permissions granted rather than the permissions in P'.

4. $\mathtt{U}^{R',\mathcal{X}_u}_{\mathtt{G}/\bar{\mathtt{A}}}$ the utility of granting the access when it is not an attack.

Henceforth, we do not explicitly indicate the request being evaluated $\mathbf{Q}_u$, the set of roles R' and current context $\mathcal{X}_u$ to simplify the notation (unless it is not clear what request or context we are referring to); however, note that the utility depends on the context of the user and permissions being requested. The expected utility (EU) of denying and granting access is computed as follows:

$$EU(\bar{\mathtt{G}}) = q * \mathtt{U}_{\bar{\mathtt{G}}/\mathtt{A}} + (1-q) * \mathtt{U}_{\bar{\mathtt{G}}/\bar{\mathtt{A}}} \qquad (1)$$
$$EU(\mathtt{G}) = q * \mathtt{U}_{\mathtt{G}/\mathtt{A}} + (1-q) * \mathtt{U}_{\mathtt{G}/\bar{\mathtt{A}}} \qquad (2)$$

An access request should be granted when $EU(\bar{\mathtt{G}}) \leq EU(\mathtt{G})$, otherwise it should be denied. Therefore, the threshold to decide when an access should be granted or denied can be computed by equalizing equations (1) and (2), obtaining:

$$q * \mathtt{U}_{\bar{\mathtt{G}}/\mathtt{A}} + (1-q) * \mathtt{U}_{\bar{\mathtt{G}}/\bar{\mathtt{A}}} = q * \mathtt{U}_{\mathtt{G}/\mathtt{A}} + (1-q) * \mathtt{U}_{\mathtt{G}/\bar{\mathtt{A}}} \qquad (3)$$

The only unknown value in equations (3) is q. By solving equation (3) for q, we can find the threshold value for an access. The solution of this equation is provided in the following definition.

Definition 32. *Given the utility values $\mathtt{U}_{\bar{\mathtt{G}}/\mathtt{A}}$, $\mathtt{U}_{\bar{\mathtt{G}}/\bar{\mathtt{A}}}$, $\mathtt{U}_{\mathtt{G}/\mathtt{A}}$ and $\mathtt{U}_{\mathtt{G}/\bar{\mathtt{A}}}$ for context $\mathcal{X}_u$, request $\mathbf{Q}_u$ for which a set of roles R' are enabled for u_r provided the risk management procedure allows it, the decision-making threshold is defined as follows:*

$$\tau(R', \mathcal{X}_u) = \frac{\mathtt{U}_{\mathtt{G}/\bar{\mathtt{A}}} - \mathtt{U}_{\bar{\mathtt{G}}/\bar{\mathtt{A}}}}{\mathtt{U}_{\bar{\mathtt{G}}/\mathtt{A}} + \mathtt{U}_{\mathtt{G}/\bar{\mathtt{A}}} - \mathtt{U}_{\mathtt{G}/\mathtt{A}} - \mathtt{U}_{\bar{\mathtt{G}}/\bar{\mathtt{A}}}}$$

If $\tau > 1$ then $\tau = 1$ and if $\tau < 0$ then $\tau = 0$.

The utility values depend on the context and request; hence, a different threshold is used for each context and request. The risk management procedure is defined as follows.

Definition 33. *Let R' be a set of roles enabled for u_r that satisfies request $\mathbf{Q}_u$ under context $\mathcal{X}_u$. The risk management decision-making process is as follows:*

$$RiskMan(R', \mathcal{X}_u) = \begin{cases} true & if\ \tau(R', \mathcal{X}_u) > Pr[\mathtt{A} \mid \mathcal{X}_u, R'] \\ false & if\ \tau(R', \mathcal{X}_u) \leq Pr[\mathtt{A} \mid \mathcal{X}_u, R'] \end{cases}$$

where $Pr[\mathtt{A} \mid \mathcal{X}_u, R']$ is the probability of $\mathbf{Q}_u$ being an attack given $\mathcal{X}_u$ and R'.

Table 6: Example utility values for two different contexts.

	Utility for $\mathbf{Q}_u$, R'				
Context	$\mathtt{U}_{\mathtt{G}/\mathtt{A}}$	$\mathtt{U}_{\mathtt{G}/\bar{\mathtt{A}}}$	$\mathtt{U}_{\bar{\mathtt{G}}/\bar{\mathtt{A}}}$	$\mathtt{U}_{\bar{\mathtt{G}}/\mathtt{A}}$	τ
$\mathcal{X}_{u1}$ (emergency room)	0	90	5	15	0.85
$\mathcal{X}_{u2}$ (remote access)	0	70	10	25	0.71

Example 11. *Consider a doctor trying to access a patient's record in two different contexts. Suppose that a set of roles R' could satisfy the doctor's request, $\mathbf{Q}_u$, provided the risk management procedure allows it. In context $\mathcal{X}_{u1}$, he is trying to access from an emergency room and in context $\mathcal{X}_{u2}$ he is requesting the same record from his home. The utility values for both contexts and the threshold values are presented in Table 6. Because the utility is a measure of satisfaction, the utility of granting access in an emergency room is larger than denying the access when the doctor is at home when there is no attack. This is true considering that granting access to a patient's data from the emergency room may save the patient's life. Similarly, $\mathtt{U}_{\bar{\mathtt{G}}/\bar{\mathtt{A}}}^{R',\mathcal{X}_{u1}} < \mathtt{U}_{\bar{\mathtt{G}}/\bar{\mathtt{A}}}^{R',\mathcal{X}_{u2}}$, because we would be less satisfied to have an access denied in the emergency room than in the other context. Given these utilities, the thresholds are computed according to Definition 32. Suppose $Pr[\mathtt{A} \mid \mathcal{X}_{u1}, R'] = Pr[\mathtt{A} \mid \mathcal{X}_{u2}, R'] = 0.8$. In this case, we have $\tau(R', \mathcal{X}_{u1}) = 0.85 > 0.8$, so the access is granted. Note that this is equivalent to finding the expected utilities in equations (1) and (2), for which the analysis shows that $EU(\mathtt{G}, \mathcal{X}_{u1}) = 18$ and $EU(\bar{\mathtt{G}}, \mathcal{X}_{u1}) = 13$. Since the expected utility of granting is greater than the utility of denying the access, in this case, the best decision is to grant the access. In context $\mathcal{X}_{u2}$, $\tau(R', \mathcal{X}_{u2}) = 0.71 < 0.8$, so the access is denied. Again, using the threshold values is equivalent to computing the expected utilities, which are $EU(\mathtt{G}, \mathcal{X}_{u2}) = 14$ and $EU(\bar{\mathtt{G}}, \mathcal{X}_{u2}) = 22$; it also results in denying the access. Hence, when the access is from home, $\mathcal{X}_{u2}$, the system requires a larger assurance that the request is not an attack, whereas in a more critical type of access such as $\mathcal{X}_{u1}$, the system is more tolerable to the risk of attack because the associated utility values allow a riskier behavior.*

Obtaining Utility Values: Utility values are subjective in nature and, therefore, each organization should elicit them. An in-depth review of the widely studied *utility elicitation* *process* can be found in [39]. In what follows, we provide some *guidelines* to help the policy administrator find these values.

- Utility values should be in the interval between zero and one hundred, where a higher value reflects a higher preference for a particular outcome.
- Additionally, utility values should satisfy the following relations to be correct. First, $\mathtt{U}_{\mathtt{G/A}} < \mathtt{U}_{\mathtt{G/\bar{A}}}$, because an organization would be clearly more satisfied if an access request is granted and it turns out to be a legitimate access request, than if granting the access results in an attack. Similarly, $\mathtt{U}_{\mathtt{\bar{G}/A}} < \mathtt{U}_{\mathtt{\bar{G}/\bar{A}}}$, because an organization is more satisfied if an access request that aims to attack the organization is denied than if a non-malicious access is denied.
- To find the utilities associated with a request $\mathbf{Q}_u$ that requires the activation of R', one needs to consider the permissions that would be obtained by the requesting user, $P_{au}(R')$, and the inferred purpose of the request. To define a value for $\mathtt{U}_{\mathtt{G/A}}^{R',\mathcal{X}_u}$, all possible misuses of the permissions authorized by R' should be considered. Similarly, to define $\mathtt{U}_{\mathtt{G/\bar{A}}}^{R',\mathcal{X}_u}$ the perceived benefit of granting access to R' should be considered. This may change according to the context of the user, which is often intertwined with the purpose of the access, as example 11 illustrates. Another example of the perceived benefit associated with granting a legitimate access is the benefit associated with the completion of a critical transaction.

Estimating the Probability of Attack and Probability of collusion: G-SIR estimates the probability of events $\mathtt{A}$ and $\mathtt{\bar{A}}$. We use the following methodology to find those probabilities. The probability of attack depends on the behavior exhibited by the requester. To compute it, all available information such as browsing history, email transfer, geo-social behavior, among others can be aggregated. Geo-social information can also be included as part of the information aggregated. Our previously proposed methodology presented in Chapter 5 can be used to analyze the deviations from users' observed behavior with respect to geo-social contracts, geo-social obligations and geo-social trace constraints. Whenever a user violates a contract, attempts an access without completing required traces or violates an obligation, G-SIR creates a record that is later analyzed to determine how trustworthy the user is. To aggregate all such available information, several existing information fusion approaches can be used (e.g., Bayesian networks [60, 61]). These information fusion approaches are dependent on the data available and the organization that is implementing it, and they are out of the scope of this dissertation.

The probability of collusion is also an input for G-SIR (function *PrCollusion*). Depending on the domain, collusion characteristics may vary. As a result indicators and methodologies to find colluding users may change. Methodologies such as the ones presented in [89, 96] may be used to determine the value of *PrCollusion*.

6.5 ENFORCEMENT ALGORITHM

To enforce the G-SIR policy, we propose Algorithm 4. The inputs to the algorithm are the requester u_r, a set of requested permissions P', the location of the requester $\mathcal{L}_u$ and his context $\mathcal{X}_u$. The algorithm looks for a set of roles R' to satisfy the access request. If at the end of the execution R' is empty, the access is denied. Otherwise, it is granted. First, the algorithm verifies if the requester u_r is violating any contract in line 2, and if he is, the access is denied.

Candidate role selection: Next, in line 4 the set of candidate roles R_{avail} is found using function *getCandidateRoles* (presented in line 11). First, in line 12, the function verifies if all permissions in P' can be obtained through the roles assigned to u_r. If not, the request cannot be granted because there are no roles assigned to u_r that provide P'. In which case, an empty set of available roles is returned and the access is denied. Otherwise, the function continues its execution initializing variables R_{avail} and R_i. R_{avail} is a set used to store roles assigned to u_r that have all its constraint vectors fulfilled according to Definition 30. R_i is a set variable used to store roles that provide one or more permissions in P'. Both R_{avail} and R_i are initially empty. In line 15, all roles assigned to u_r are evaluated and only those that provide requested permissions are added to R_i. Then, in line 18 all roles in R_i are verified to see if their constraint vectors are satisfied. This verification consists in evaluating the following conditions (line 19): that u_r's current location allows the activation of r, that

Algorithm 4 Geo-Social Decision Making Process

Input: u_r:= requesting user, P':= Permissions requested, $\mathcal{L}_u$:= location of u_r, $\mathcal{X}_u$:= context of u_r.
Output: R':= set of roles that fulfill Definition 31. If $R' \neq \emptyset$, the access is denied. Otherwise, roles R' can be activated to grant the access request.

```
1: findGeoSocialRoleActivationSet($u_r, P', \mathcal{L}_u, \mathcal{X}_u$)
2: if $\neg fulfillContracts(u_r)$ then
3:   return $\emptyset$ {Request denied}
4: $R_{avail} \leftarrow$ getCandidateRoles($u_r, P', \mathcal{L}_u$) {Candidate roles}
5: if $R_{avail} = \emptyset$ then
6:   return $\emptyset$ {Request denied}
7: $R_{sel} \leftarrow \emptyset$ {Selected roles so far}
8: $P_{rem} \leftarrow P'$ {Set of permissions that haven't been found}
9: $R' \leftarrow selectRolesMinimumRisk(P_{rem}, R_{avail}, R_{sel}, u_r, \mathcal{X}_u)$ {See Chapter 4, Algorithm 1}
10: ------------------------------------------------------------
11: getCandidateRoles($u_r, P', \mathcal{L}_u$)
12: if $(P' \setminus P_{au}(assigned(u_r))) \neq \emptyset$ then
13:   return $\emptyset$ {Authorized roles cannot provides $P'$}
14: $R_{avail}, R_i \leftarrow \emptyset$
15: for all $r \in assigned(u_r)$ do
16:   if $(P_{au}(r) \cap P' \neq \emptyset)$ then
17:     $R_i \leftarrow R_i \cup \{r\}$
18: for all $r \in R_i$ do
19:   if $validLocation(u_r, r) \wedge completeTraces(r, u_r)$
      $\wedge$ $(inhibitors(u_r, r) = \emptyset)$ then
20:     if enoughNonColludingEnablers($r, u_r$) then
21:       $R_{avail} \leftarrow R_{avail} \cup \{r\}$
22: if $(P' \setminus P_{au}(R_{avail})) \neq \emptyset$ then
23:   return $\emptyset$ {Roles in $R_{avail}$ cannot provides $P'$}
24: return $R_{avail}$
25: ------------------------------------------------------------
26: enoughNonColludingEnablers($r, u_r$)
27: for all $ce \in r.\mathcal{E}$ do
28:   $U_{avail} \leftarrow \emptyset$
29:   $U_v \leftarrow vicinity(ce.SC) \setminus u_r$
30:   if $|U_v| < ce.k$ then
31:     return false
32:   for all $u_v \in U_v$ do
33:     if $fulfillContracts(u_v)$
        $\wedge$ $fulfillSocialPredicate(u_r, u_v, ce.\mathcal{S})$ then
34:       $U_{avail} \leftarrow U_{avail} \cup \{u_v\}$
35:   if $ce.k \leq |U_{avail}|$ then
36:     $found \leftarrow false$
37:     while $U_a \subseteq combinations(U_{avail}, ce.k) \wedge \neg found$ do
38:       if $PrCollusion(U_a \cup \{u_r\}) < ce.\tau$ then
39:         $found \leftarrow true$
40:     if $\neg found$ then
41:       return false {Couldn't find enablers for $ce$}
42:   else
43:     return false {Not enough users in $U_{avail}$}
44: return true {All enabling constraints are satisfied.}
```

u_r has completed the traces required for the activation of r and that there are no users in the vicinity who conflict with r's inhibiting constraint. If these conditions are satisfied, the function proceeds to evaluate if the enabling constraints associated with r are also fulfilled. For this purpose, in line 20 a function that verifies r's enabling constraints is invoked (we discuss this function later). If r's constraint vector is satisfied, r is added to R_{avail} in line 21. Hence, R_{avail} only contains roles with fulfilled constraint vector. Because R_{avail} may be a subset of R_i, one last verification is performed. In line 22, roles in R_{avail} are verified to see if they can provide all the permissions in P'. If they cannot, the function returns an empty set and the access is denied. Otherwise, R_{avail} is returned in line 24.

Finding non-colluding enablers: To find the set of non-colluding enablers function *enough-NonCollludingEnablers* is invoked in line 20. This function is presented in line 26. Variable U_{avail} is initially empty and is used to store users who are potential enablers. For each enabling constraint ce associated with role r (line 27), the set of users in the vicinity are retrieved and stored in U_v (line 29). Users in U_v are examined to determine if they are violating their contracts or do not fulfill the required social predicate (line 33). Only users who are not violating their contracts and fulfill ce's social predicate are added to U_{avail}. After that, if U_{avail} does not have the required $ce.k$ the function returns false to show that there are no valid enablers for r (line 43). Otherwise, groups of size k are evaluated in line 37. If none of the groups evaluated are collusion free, the function returns *false* to show that there are no valid enablers for ce. If a group U_a is found to be non-colluding with the required probability, ce is satisfied and variable *found* is set to true to show that there is no need to continue examining other groups. It is necessary to ensure that all enabling constraints in $r.\mathcal{E}$ are satisfied; hence, the function continues evaluating all constraints (for loop line 27). If after all constraints $ce \in r.\mathcal{E}$ have been evaluated and a set of collusion free enablers has been found for each ce, the function returns *true* in line 44.

Selection of roles to activate with minimum risk: After R_{avail} is found, it is guaranteed to have uniquely roles assigned to u_r for which constraint vectors are fulfilled. If R_{avail} is empty, there are no roles and the access is denied (line 6). Otherwise, the algorithm proceeds to find the roles to be activated. There may be multiple subsets of roles in R_{avail} that could satisfy the request. To select the set of roles to be activated, we leverage on Algorithm

1 presented in Chapter 4, which selects the set of roles that minimizes the risk exposure. This function, *selectRolesMinimumRisk*, is called in line 9. The only caveat is that the threshold used in the algorithm is computed differently. Letting τ be the threshold found through Definition 32 and τ_x be the threshold used in the algorithm presented in Chapter 4, when implementing function *selectRolesMinimumRisk* the following replacement should be performed: $\tau_x = 1 - \tau$. This follows because utility-based risk methodology proposed in this chapter is built so that a request with higher risk has a smaller τ while τ_x follows the opposite relation. After function *selectRolesMinimumRisk* is invoked, it returns the set of roles with minimum risk exposure that can be activated by u_r to satisfy the request. If the function returns an empty set, no role can be activated to satisfy the access request and the access is denied. Otherwise, the access is granted by activating R'.

6.6 EXPERIMENTAL EVALUATION

We evaluate the proposed system using a discrete indoor simulator implemented in Java. We describe the experimental setup and then the experimental results.

6.6.1 Experiment Setup

6.6.1.1 Generation of social graph and user mobility For simulating user mobility, we randomly generated a map where the assumed organization is located, as follows. First, we specified a size of a Cartesian rectangle. Then, we randomly selected the points where places are located on the map. These places were also randomly connected according to the parameters specified in Table 7. In our implementation, we used a graph abstraction where vertices represent the places on the map and edges represent connections between places (for e.g., corridors).

At the beginning of the simulation, all users were randomly placed on the map. Each policy was evaluated at multiple time instants, and at every time instance users could move around the map to adjacent places or stay in their current positions. The speed of the users was set to be 5 feet per second.

Social graphs were generated using the Jung API provided in [3]. We evaluated the effect of representative types of network topologies on the system. The topologies evaluated include *preferential attachment* [10] *small world* [77], *power law* [48] and a fully connected network. These topologies are commonly observed in different social networks. All graphs evaluated were undirected.

6.6.1.2 Generation of policy, access requests and threats Policies were randomly generated using the parameters presented in Table 7. We ensured that all policies used in the experiments were well-formed according to Definition 29. We selected the values of the parameters inspired by previous works such as [11, 117], which evaluate RBAC policy enforcement. Geo-social policies have not been evaluated in previous work. Hence, we adjusted some parameters and included new ones to incorporate unique geo-social features. In the following experiments, we test different values for those parameters to show their effect. Role' activation thresholds (which represent the maximum tolerable probability of attack, Definition 32) were randomly assigned between 0 and 0.5 because it is only justifiable to use roles with spatial scope and other geo-social constraints when the information protected is valuable. The probabilities of attack used for the risk management procedure were randomly generated for each user and assumed to be accurate. Initially, the probabilities of attack were set to 0.01. Throughout the simulation, the probabilities of attack for each user were randomly updated.

To generate inhibiting constraints, we created three classes of confidential data and assigned to each class a color that represents the type of individuals who should not be allowed to access it. When a role was generated with an inhibiting constraint, one color was randomly selected. At the beginning of the simulation, we randomly selected inhibiting users and tainted them with a random color.

Enabling constraints were randomly generated to required k users related by friendship to the requester in the spatial scope of the role to be activated. Colluding communities were randomly generated. We considered two parameters, the number of colluding communities and the number of members per colluding community. For each community, we randomly selected a user and marked him as colluding and then, continued choosing some of his friends as colluding until the number of colluding users per community was reached.

Trace constraints were generated randomly verifying that the path required to arrive at the place of access did indeed exist. The number of previous places users needed to visit was set as 2. The time required to fulfill the constraint (Definition 28) was generated considering the distance between places and the speed of users to allow enough time.

Request generation, events counted as threats and measure of improvement: To generate access requests, every time a user stepped into a place where there was a role with spatial scope, a request was issued on his behalf. We consider the following as potential insider threats:

(a) *Not authorized for role:* A user issues a request to obtain access to information he is not authorized for.

(b) *Inhibiting users:* A user issues an access request, but there are inhibiting users in the vicinity that may launch a proximity attack, e.g., if an access request is evaluated for a role with color *red*, if any user tainted *red* is in the vicinity, he is classified as inhibitor.

(c) *Lack of enablers:* A request is issued, but there are not enough enablers at the required place to authorize the request.

(d) *Colluding users:* A user issues an access request that requires enablers and the only people who could serve as enablers are colluding according to the collusion threshold.

(e) *Enablers violating contracts:* A user issues a request for which all potential enablers are violating their contracts (they are in places where they should not be).

(f) *Suspicious requester:* A user issues a request for which the probability of attack is too high compared to the role's activation threshold.

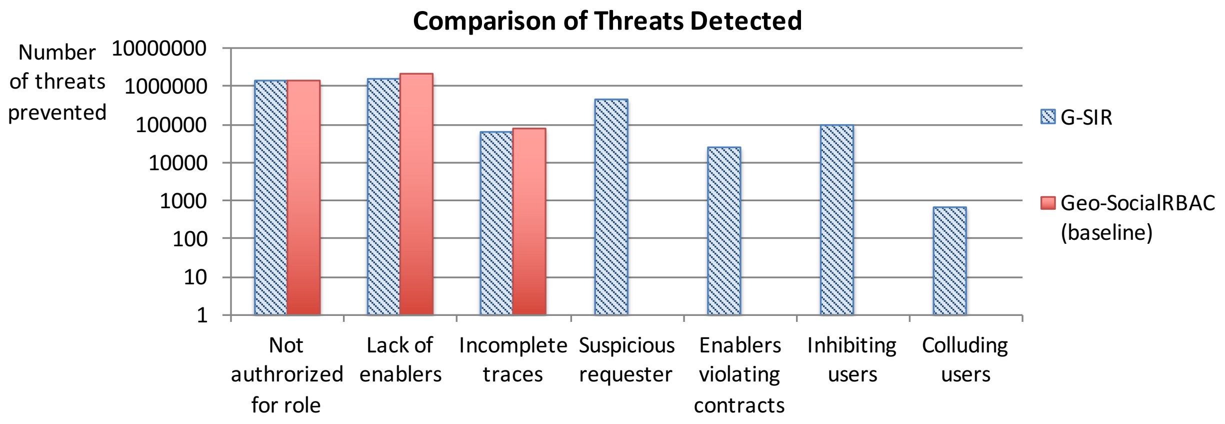
(g) *Incomplete traces:* A user issues a request without completing the required traces.

Each access request was only classified in a single category according to the order of constraint evaluation in Algorithm 4.

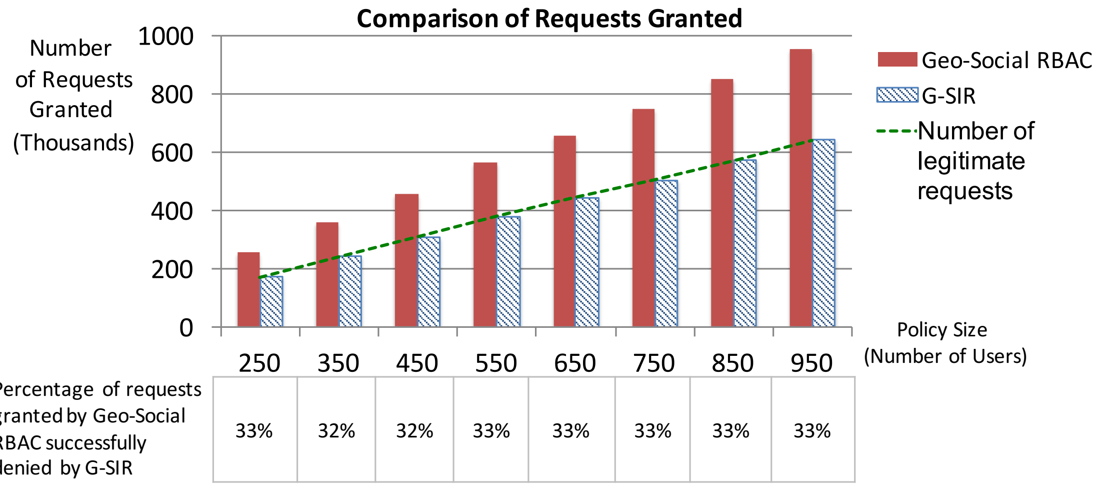
To evaluate our proposed approach, we use as a baseline the Geo-Social RBAC model introduced in Chapter 2, section 2.6. Some of our experiments aim to measure the percentage of threats detected by our framework in comparison to the baseline. Our objective is to

Table 7: Default experiment parameters. The *number of users* is used to scale the size of the evaluated policies.

Parameter	**Value**
Ratio of total number of places to total number of users	1:3
User speed	5 feet per second
Map coordinates (size of Cartesian map)	(300 feet x 300 feet)
Ratio of number of users to roles	4:1
Ratio of roles assigned per user to number of roles	1:2
Inhibiting constraints per role	1
Percentage of roles with inhibiting constraints	50%
Number of inhibitors	3 types of inhibiting users (colors), 40% of users were assigned a random color.
Enabling constraints per role	1
Range of k	[1, 3]
Required social relation	Friendship
Collusion threshold	0.9
Number of colluding communities	5% of number of users
Number of colluding users per community	5
Roles with trace-based constraints	5%
Roles with geo-social contracts	40%
Simulation time	8 hours

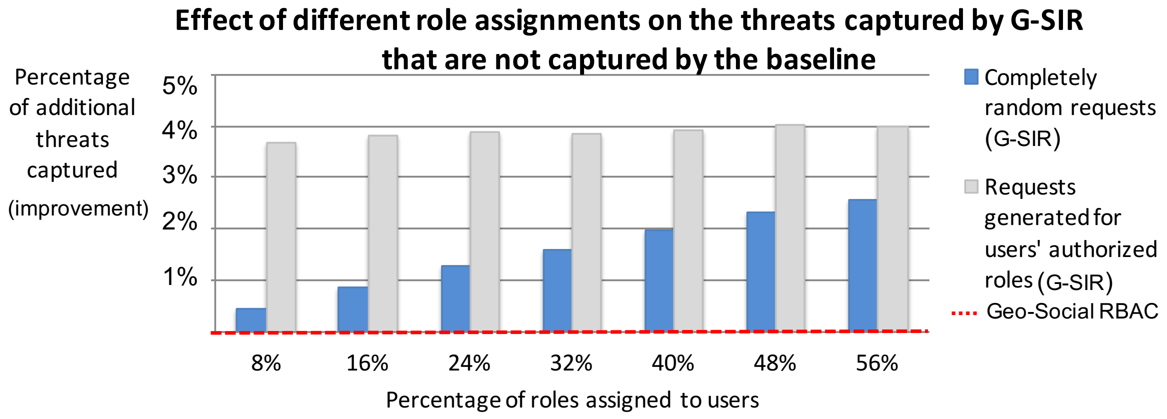


(a) Detailed comparison between the threats prevented by the proposed approach G-SIR and the baseline Geo-Social RBAC. Plot in logarithmic scale.



Policy Size (Number of Users)	250	350	450	550	650	750	850	950
Percentage of requests granted by Geo-Social RBAC successfully denied by G-SIR	33%	32%	32%	33%	33%	33%	33%	33%

(b) Comparison between the number of requests granted.



(c) Percentage of threats not captured by the baseline, which are prevented by our framework (the line in 0% represents the baseline).

Figure 20: Comparisons between the proposed G-SIR and the baseline (Geo-Social RBAC).

present an overall picture of the effect of including a relevant policy constraint in G-SIR with respect to the baseline. We refer to this number as *improvement* and it is computed as follows[4]: $improvement = \left[\frac{n_{proposed}}{n_{baseline}}\right] - 1$, where $n_{proposed}$ is the number of potentially malicious access requests (threats) detected by G-SIR which is equal to the sum of all previously described threats (a) to (g) and $n_{baseline}$ is the number of threats detected by the baseline, which is the addition of threats of type (a), (c) and (g). In the following figures, we show the improvement as a percentage.

6.6.2 Analysis of Results

In this subsection, we discuss and analyze the results of our various experiments. Each reported experimental measurement is the average of running the simulation 30 times (each time a different randomly generated policy was used). The results presented were found using 30 randomly generated OSNs, 10 of each topology. The number of users in the simulation were fixed at 250. Some experiments change the *policy size*; this value is determined by the number of users according to the parameters in Table 7.

Baseline Comparisons: First, we present an overview of the types of attacks prevented by the proposed G-SIR with respect to the baseline, Geo-Social RBAC. Figure 20a shows the number of threats detected by G-SIR that could not be captured by the baseline. The first column in the figure presents the number of access requests denied because the requester was not authorized for a role; hence, this number is the same for both approaches. The second and third columns show the number of requests denied due to *lack of enablers* and *incomplete traces*, this is slightly larger for the baseline, because our approach finds other policy violations first, according to the order presented in Algorithm 4. Geo-Social RBAC does not capture any of the remaining violations: *suspicious requester*, *enablers violating contracts*, *inhibiting users* and *colluding users*. We note that there may be some misclassifications in the counts of *suspicious requester* and *colluding users*. This is caused by the uncertainty in the estimation of the probability of attack and the probability of collusion, respectively. We present the number of false positives and false negatives in a later experiment. In this exper-

[4]This formula is typically used to determine the percentage of new features captured by a proposed approach with respect to a baseline.

iment, we assume that these two values are accurate. Given this assumption, the proposed G-SIR mitigates more threats.

Figure 20b presents the number of requests granted by the baseline and our proposed approach. In the x-axis, we show the results for multiple policy sizes. The dotted line represents the number of requests granted that are legitimate. All requests that are above that line are malicious ones and should not have been granted. The table below the figure contains the exact percentage of malicious requests granted by the baseline that G-SIR was successfully able to deny. Overall, the results show that the baseline granted around 33% of malicious requests irrespective of the policy size. The policy size uniquely influenced the total number of requests granted. Overall, G-SIR was able to identify 33% more policy violations than the baseline.

The percentage of additional threats captured by G-SIR depends on the type of policy enforced, in particular, the number of roles assigned to each user. Figure 20c presents the improvement measured as the percentage of threats detected by our framework in comparison to the baseline Geo-Social RBAC. In the x-axis of Figure 20c, we show the percentage of roles assigned to users. The bars represent the percentage of threats detected by the proposed system that are not detected by the baseline[5]. We present the percentage of threats for two simulation techniques. The first one, when access requests are completely randomly generated, and the second one, when requests are randomly generated but users only issue requests for their authorized roles. Figure 20c shows that the improvement increases as the percentage of roles assigned to users also increases. This is the case because, for smaller percentage of role assignments, the majority of access requests denied are denied because users are not authorized for roles. Since both the baseline Geo-Social RBAC and the proposed G-SIR detect this type of threat, the improvement in comparison to the baseline is smaller. For this reason, the percentage of new threats captured by our framework is larger when we run the simulation such that users only request an access if they are authorized for a role. As it can be seen, the improvement increases with respect to the results shown for completely random requests.

[5]We found this number using the improvement formula previously presented in this sub-section; hence, the line 0% represents Geo-Social RBAC.

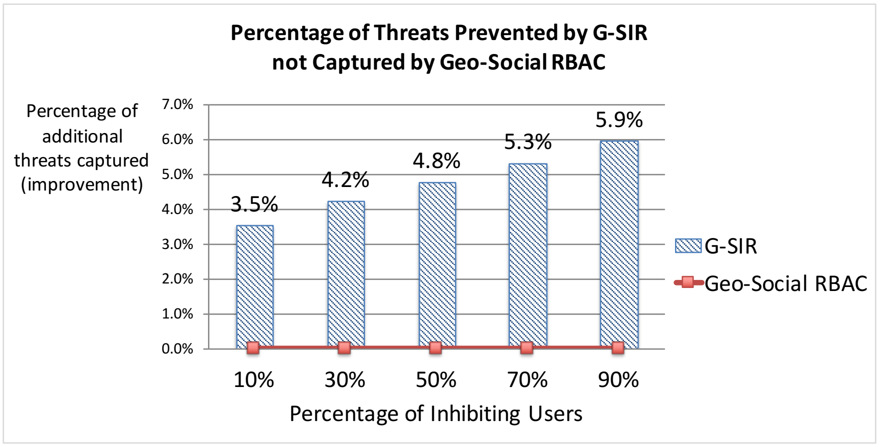
We note that the difference between the percentage of threats captured in Figure 20b and the malicious request granted by Geo-Social RBAC in 20b is due to the difference between the number of requests generated by the simulator that were granted and the number of requests that were denied. The number of requests denied due to the lack of assigned role, and lack of enablers is substantially larger than all other types of requests (Figure 20a) including the number of granted requests. Hence, the improvement reported is greater than when only the number of granted requests is considered.

In the following, we present the effect of increasing both the percentage of attacks that are addressed by G-SIR and the different types of constraints in the system. Unless explicitly mentioned, parameters are maintained to their default values (Table 7).

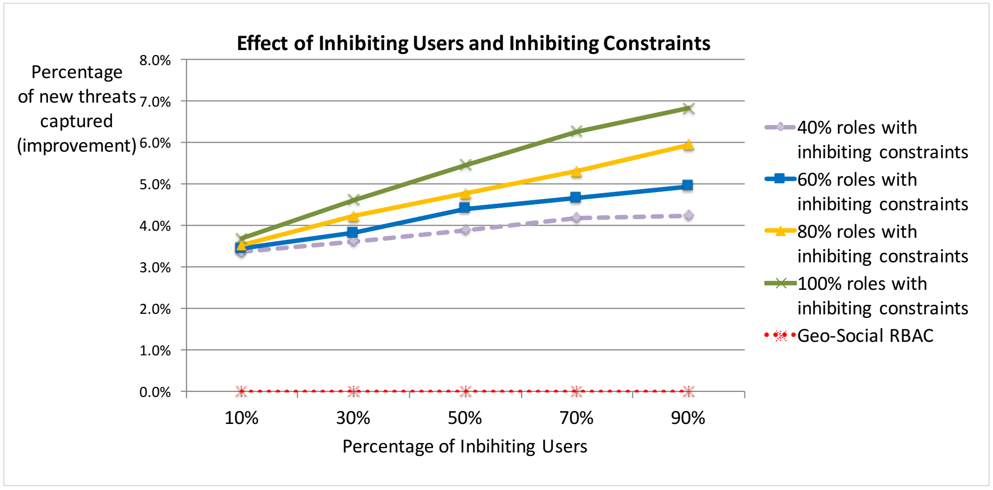
Proximity Attacks: In Figure 21a we present the percentage of threats prevented by G-SIR that were not captured by the baseline as the number of inhibiting users increases. In this figure, the baseline is represented by the line in 0%. When only 10% of the users were inhibitors, G-SIR was able to capture 3.5% more threats than the baseline. While a system where 90% of the users cannot learn some information (recall that there are three colors), resulted in an improvement of 5.9% of threats captured. Figure 21a, was built for a policy where 80% of roles had an inhibiting constraint.

Figure 21b presents the effect of environments with a different number of inhibiting users under policies with various number of roles with inhibiting constraints. As the number of roles with inhibiting constraints increases, there are more confidentiality leaks prevented. Similarly, as the number of inhibitors in the vicinity increases, the number of leaks of confidential information due to proximity attacks is also higher. Since the baseline does not prevent this type of attacks, the overall number of threats prevented by G-SIR increases. When 100% of users are assigned an inhibiting color and all roles have an inhibiting constraint associated with them, the number of threats mitigated goes up to 6.8%.

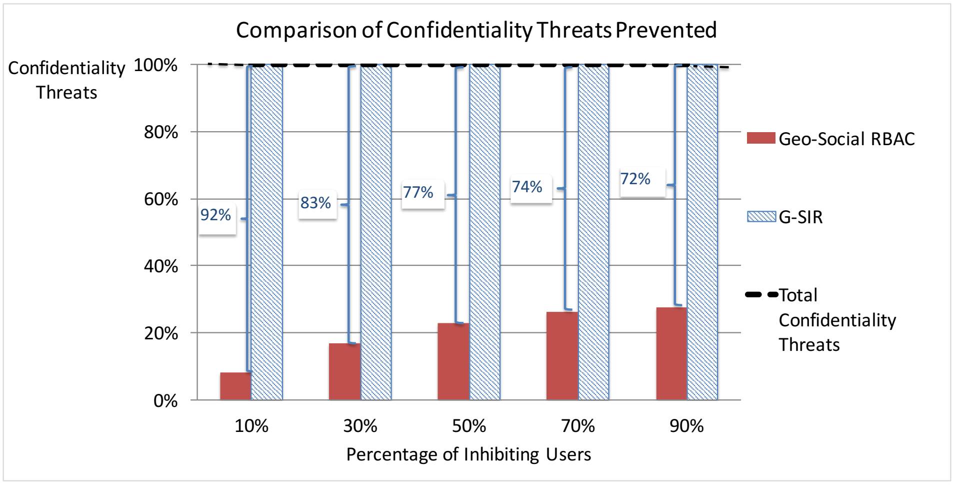
In our next experiment, we counted as a confidentiality threat an attempt to do any of the following: *i)* when a user tries to access a role that he is not assigned to and the role has an inhibiting constraint that conflicts with the color of the user, *ii)* when there are inhibitors in the vicinity, *iii)* when there is a collusion to access confidential information and *iv)* when there is a contract violation and the violating user is trying to serve as an enabler.



(a) Effect of inhibiting users on the overall number of threats captured by G-SIR that are not captured by Geo-Social RBAC.



(b) Effect of different ratios of roles with inhibiting constraints over the number of threats captured for different number of inhibiting users.



(c) Percentage of confidentiality threats prevented by G-SIR in contrast to Geo-Social RBAC.

Figure 21: G-SIR proximity threat results.

Using this classification, in Figure 21c, we present a comparison between the percentage of confidentiality threats detected by G-SIR in contrast to Geo-Social RBAC. In this experiment, the proposed G-SIR captures more confidentiality threats than those captured by the baseline (Geo-Social RBAC). Geo-Social RBAC only captures confidentiality threats of type *i).* As the percentage of inhibiting users increases, the number of all types of confidentiality violation attempts enumerated before also increases; including those of type *i).* That is why we see that the percentage of confidentiality threats captured by Geo-Social RBAC does increase with the number of inhibiting users. However, there is always a large percentage of threats that are not detected by Geo-Social RBAC. Figure 21c was generated for policies where 60% of roles have inhibiting constraints. For these policies, the percentage of threats not captured by Geo-Social RBAC vary between 92% to 72%. For policies with a higher number of roles with inhibiting constraints, the number of threats not captured by Geo-Social RBAC that are captured by our G-SIR is larger. For instance, when all roles have inhibiting constraints and there are 90% of inhibiting users, the percentage of threats not captured by the baseline captured by G-SIR increases to 76%. This corresponds to 4% more than the same data point in Figure 21c. These experiments show that G-SIR is effective capturing proximity and confidentiality threats.

Collusion Attacks: Figure 22 presents the effect of increasing the number of colluding users per community (x-axis) and the number of colluding communities. Colluding attacks are not prevented by the baseline, hence all the lines in the figure represent attacks thwarted by G-SIR. The results reported were generated for policies with enabling constraints that required one enabler ($k = 1$). In this experiment, we assumed that the colluding communities and users were known. Hence, all attacks presented in Figure 22 can be prevented by G-SIR. In a real system, the *accuracy* depends on the accuracy of the community detection algorithms used, e.g., [89, 96]. Figure 22 shows that the number of colluding attacks prevented by G-SIR increases with the number of communities. This follows because the probability of detecting an attack when more communities exist is larger. Similarly, as the number of colluding members per community increases, the probability of a collusion attack increases and the number of collusion threats increases.

Geo-Social Contract Violations: Figure 23 presents the number of contract violations

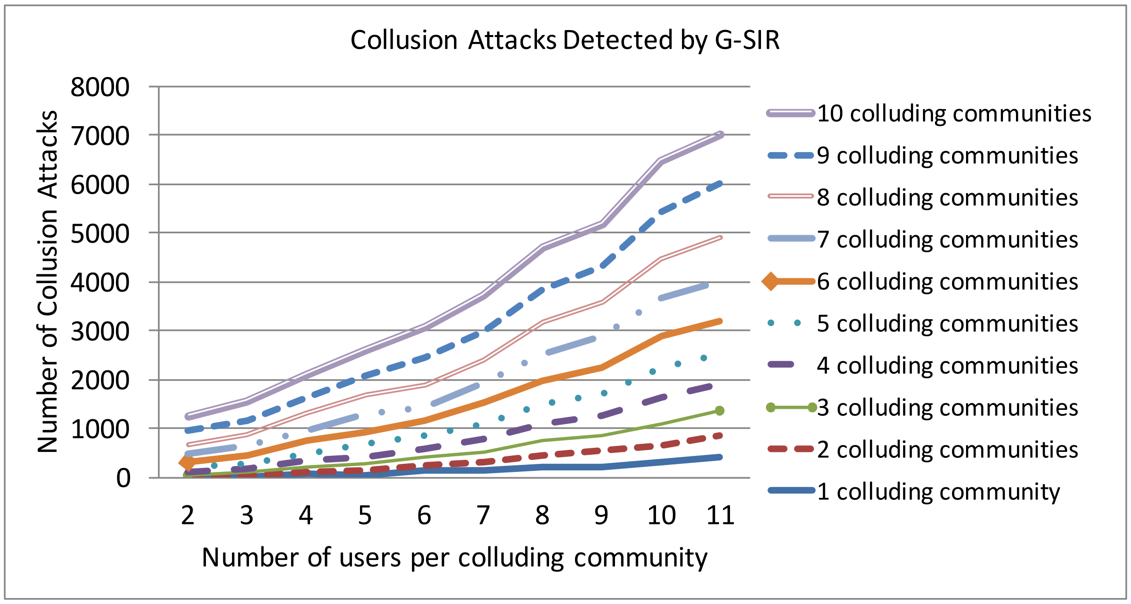
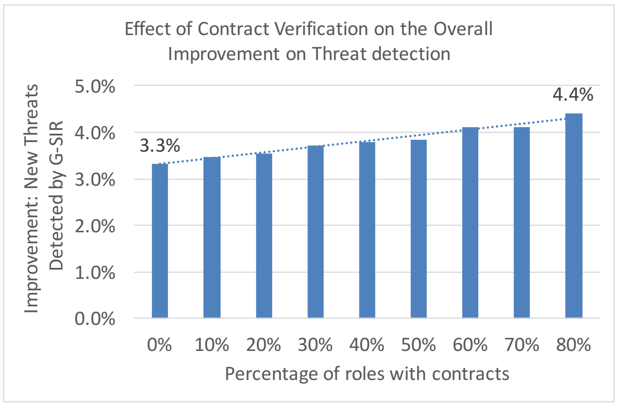
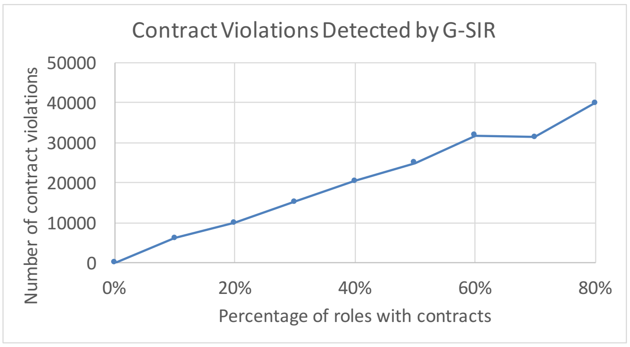


Figure 22: Collusion threats captured by G-SIR.

stopped by G-SIR as the percentage of roles with contract constraints increases. Note that the baseline does not prevent any of these attacks. In Figure 23a, we present the overall increase on the number of threats uniquely prevented by G-SIR and Figure 23b the absolute number of contract violations. In both figures, as the number of geo-social contracts in the policy increases, the threats captured by the G-SIR also increases. This implies that for organizations that require more protection against users wandering through unauthorized places, G-SIR performs better. Recall that policies randomly generated by our simulator are well-formed and all roles have a spatial scope assigned to them. Hence, the number of contract violations uniquely contains threats where a potential enabler is violating a contract. Had we used policies that contain conflicts, the number of violations reported would be larger, as the verification in line 2 of Algorithm 4 never evaluated to true during our simulations. Therefore, these figures uniquely show attacks that aim at using enablers that are not qualified to be in the required spatial scope. These figures show a clear trend where the number of attacks stopped increases as the roles with contracts is incremented. The fluctuations shown, reflect users' random movements.



(a) Effect of contract enforcement in the overall threat detection.



(b) Contract violations detected by G-SIR.

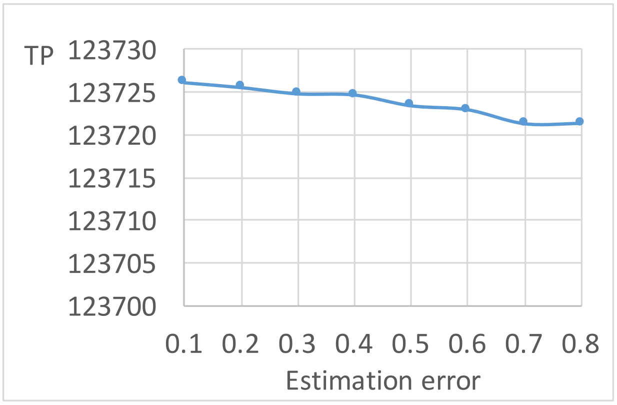
Figure 23: Effect of geo-social contracts on the number of threats captured.

Sensitivity and specificity analysis: G-SIR takes as input the estimated probability of attack. In this experiment, we measure the effect of using estimation methodologies with different values of average error, ε, on the number of threats detected by G-SIR. For this purpose, we generated synthetic data as follows. We randomly selected a probability of attack, q, for each user; this value was considered as the ground truth. Then, the *estimated probability*, $\hat{q}$, was randomly selected in the interval $[q - \varepsilon/2, q + \varepsilon/2]$. We changed the value of ε between 0.1 and 0.8. The observations generated by the simulation runs were classified according to Figure 24a as true positives (TP), false negatives (FN), false positives (FP) and true negatives (TN).

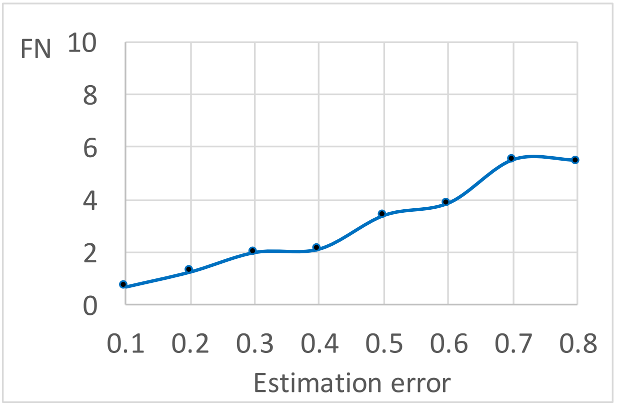
Figure 24 presents the results of this experiment, which include the average number of TP, FN, FP and TN as well as the average sensitivity and specificity. Figure 24b shows that the number of TP decreases very little as the estimation error increases. In the worst case, when ε=0.8, the number of TP is reduced on average by four observations which is relatively small compared to the total number of observations. As expected, the number of FN increases with the increase in the average estimation error as shown in Figure 24c. The average number of FN for the largest estimation error (ε=0.8) is 5.4 which is relatively small in comparison to the total number of observations. These results indicate that G-SIR is capable of stopping

Classification	Ground truth	Decision
True Positive (TP)	Attacker	Deny
False Negative (FN)	Attacker	Grant
False Positive (FP)	Legitimate	Deny
True Negative (TN)	Legitimate	Grant

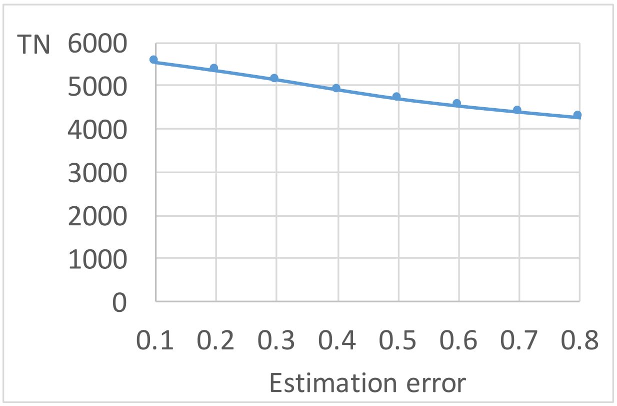
(a) Classification of observations



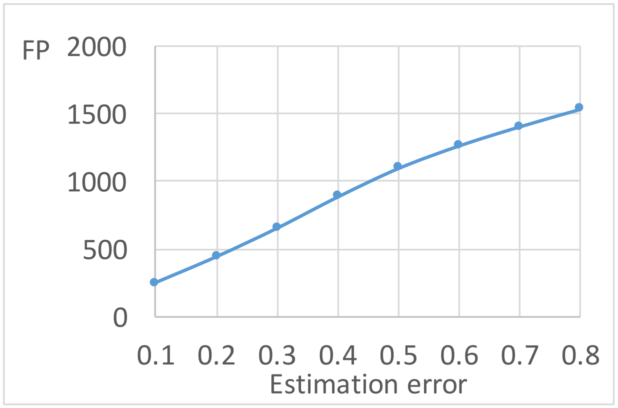
(b) True positives



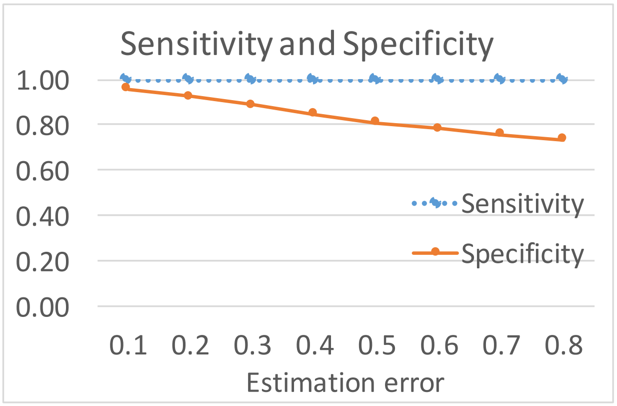
(c) False negatives



(d) True negatives



(e) False positives



(f) Sensitivity and Specificity

Figure 24: Effect of the estimation error, ε, of the inference technique used on the number of threats captured by G-SIR.

most threats generated by the simulator even when the performance of the information module is not good. This is the case because not all access request decisions are based on inferred information. Hence, access decisions that do not require inferred information are correctly made even if the inference module predictions are inadequate. Additionally, in our simulation, the thresholds related to roles reflect the fact that only valuable information is protected with geo-social policies. In Figure 24d as the estimation error increases, the number of TN decreases. The effect of the estimation error can be seen on the average FP shown in Figure 24e. This indicates that under poor estimation methodologies, the number of requests denied to legitimate users increases; however, in real systems, the estimation error in most cases should be small.

Sensitivity and specificity are measures that provide an overview of all the previous results and are shown in Figure 24f. Sensitivity represents the percentage of attackers who are correctly identified as attackers while specificity shows the percentage of legitimate insiders who are correctly identified as not being a threat. The sensitivity measure shows that G-SIR is good in capturing attacks even when the estimation error increases. This is a consequence of the following two facts. First, some of the policy constraints that are part of G-SIR do not depend on the inferred input data, so they can be enforced correctly without any influence of the estimation error. Secondly, the thresholds used in the simulation were selected to ensure that, as in real policies, only relevant information would be protected by G-SIR policies. Thus, when the inferred probability of attack is too high, even under certain error, the enforcement mechanism will deny access to the most important information. The specificity shows that as the estimation error increases, the number of honest insiders that are denied access to very critical resources increases as well. This shows that G-SIR does capture imminent insider threats even under suboptimal input.

Runtime overhead: In Figure 25, the difference between the time required by Geo-Social RBAC and G-SIR is shown for policies of multiple sizes. Our proposed G-SIR introduces some additional runtime overhead due to the extra verifications performed. However, the overhead is acceptable in comparison to Geo-Social RBAC.

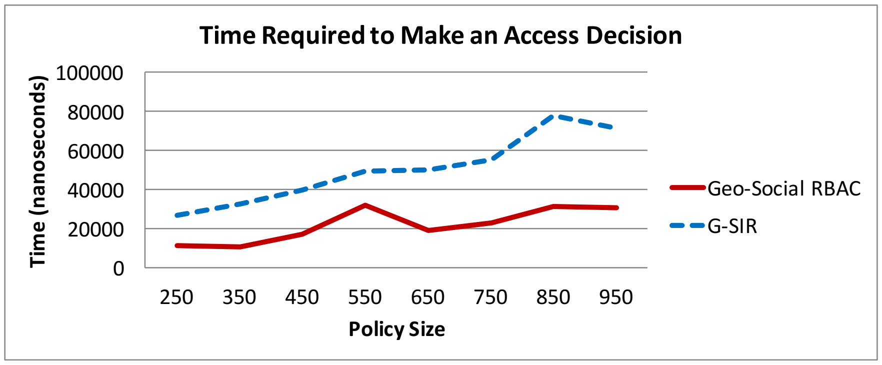


Figure 25: Average time as the policy size increases.

6.6.3 Limitations of the Experiments

Evaluating insider threat mitigation systems is a challenging task given the lack of ground truth data, standard metrics or methods to assess new approaches [62]. Furthermore, the abnormalities in the patterns of mobility of users during working hours have not been studied previously. Hence, there is not a well-known way to introduce insider attacks of this type into our simulation. In the previous experiments, we worked around this problem by randomly generating threats. The same randomly generated threats were used to compare G-SIR and Geo-Social RBAC. By randomly generating the threats, we tried to reduce as much as possible biasing the results. To better understand how the model would work under different circumstances, we run experiments under multiple parameter configurations.

In some experiments, we assumed that the inference module was accurate. This module is in charge of finding the colluding communities that lead to the computation of function $PrCollusion$ as well as finding the probability of attack $Pr[\mathtt{A} \mid \mathcal{X}_u, R']$ used by the risk management procedure. These probabilities are an estimation and hence they have an associated uncertainty. The false positives and false negatives that result due to the enforcement of our model do depend on the error associated with the methodologies to estimate these probabilities. Despite the difficulty of using estimation methodologies as part of this simulation, we run the experiment presented in Figure 24 to understand how robust G-SIR is under different estimation errors. This experiment shows that the G-SIR is robust preventing insider threats.

Other types of attacks that were not included in our simulation, but that certainly may impact a real production system include the following. We assumed that the locations of all users were known with high accuracy during the simulation. In real systems, the location may not be known with 100% accuracy for reasons that include technological limitations, malicious insiders tampering equipment or using other ways to spoof their location. Other attacks that may influence the security of the system are tampering or manipulation of the social graph. In our simulation, social graphs were generated at the beginning and did not change throughout the simulation. Hence, this threat was not measured by our simulation. Despite these limitations, we believe the experimental results show that the proposed framework can mitigate threats that existing approaches fail to prevent.

6.7 CHAPTER SUMMARY

In this chapter, we presented G-SIR. As part of this framework, we developed several policy constraints that can serve to specify appropriate and inappropriate geo-social behavior. These constraints include *geo-social contracts*, *geo-social obligations*, *inhibiting*, *enabling* and *trace-based constraints*. Their enforcement can help understand the context of a requester and subsequently deny accesses that impose too high of a risk. Additionally, by monitoring G-SIR policy compliance logs, it is possible to identify users whose geo-social behavior is questionable, dangerous or suspicious. We also presented a utility-based decision making approach to make access control decisions. Finally, we evaluated our approach through simulations and showed that enforcing the proposed G-SIR policy can help capture some threats that cannot be prevented using existing techniques.

In the next chapter, we present the conclusions, limitation and future work related to this dissertation.

7.0 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

As part of this dissertation, we have presented three complementary frameworks that aim at mitigating insider threats and unintentional damages in systems that use RBAC, obligations and Geo-Social information to perform access control.

We have presented an approach to perform access control considering the behavior of the users, and the risk exposure that an organization is ready to accept when granting access to certain roles in the system. Our approach adapts to negative behaviors of users by denying access to permissions whose misuse would negatively impact an organization. In this way, our approach is able to mitigate possible attacks when there are technical precursors that indicate a user is behaving maliciously. In order to reduce the risk exposure further, we have also defined an optimization problem, and an algorithm that reduces the risk exposure. We have presented experimental results for different types of policies. We believe the features offered by our framework make it difficult for insider attackers to misuse their privileges.

Additionally, we have proposed a framework to control the risk exposure caused by *a posteriori* obligations. As part of this framework, we proposed and evaluated a methodology to identify how trustworthy a user is to fulfill *a posteriori* obligations. Our methodology considers the latest, historical and sudden changes on users' behavior as well as users' behavior compared to his peers. The obligation-based trust value associated with each user is used to decide whether to grant or deny accesses that create *a posteriori* obligations. When a user is not considered trusted enough to fulfill *a posteriori* obligations, accesses that require the assignment of highly critical obligations are denied. In this way, our framework reduces the risk exposure caused by *a posteriori* obligations and identifies and deters insider threats without compromising the privacy of the users. In addition, we proposed a clustering-based methodology to find patterns of misbehavior and outliers in the system. Our methodology can serve to identify policy misconfigurations and suspicious users. This information allows the system administrator to take appropriate measures.

Our trust methodology can be integrated into any access control model that includes *a posteriori* obligations (e.g., UCON) and risk aware role activation mechanisms (e.g., [11]). We believe that considering the inherent risk of *a posteriori* obligations can help the systems better understand a user's intentions and mood as well as reduce the risk exposure of an organization.

Our third proposed framework is designed to take advantage of the increasing availability of geo-social information to prevent insider attacks. Few approaches have incorporated geo-social information into the access control decision-making process and none of them have considered the intricacies of incorporating geo-social information as part of the access control system for insider threat mitigation. First, we performed an analysis of insider threats that arise when geo-social information is used to perform access control decisions. To capture these new threats, we proposed Geo-Social Insider Threat Resilient Access Control Framework (G-SIR). To the best of our knowledge, this is the first effort to use geo-social information to deter insider threats by incorporating it into the access control mechanism. We proposed an access control methodology that includes several novel policy constraints that include geo-social contracts, geo-social risk aware trace constraints, inhibiting constraints, enabling constraints, and geo-social obligations. Enforcing these constraints helps in reducing the risk of proximity and social engineering attacks. Additionally, monitoring the fulfillment of these constraints may help identify suspicious users. We proposed an enforcement algorithm and presented simulation results to evaluate the proposed framework. We believe that as a result of our proposed research, organizations with geo-social security requirements will be more likely to embrace the advantages that geo-social access control systems offer.

7.0.1 Limitations and future work

There are two limitations related to the proposed work. First, we assume that all insiders access the system through a reference monitor (the PEP in Figure 2). Any attack vectors that subvert or circumvent the PEP cannot be detected by our approach. Examples of these vectors include exploiting vulnerabilities in the operating systems, physical attacks to unencrypted disks, among others. Hence, complementary measures are necessary to protect against these threats.

Secondly, our framework uses a threshold-based approach to adapt to negative changes in behavior. The threshold may be computed based on multiple anomaly detection indicators and anomaly detection techniques. For this reason, there are certain types of attacks that cannot be prevented using our approach. In particular, we distinguish between two types of attacks. *i) Threshold manipulation:* Carefully crafted insider attacks may not be noticed if the insider manages to maintain a profile that is not recognized as suspicious by the predictive system. Although we have proposed a methodology to find obligation-based trust values that is difficult to manipulate by powerful adversaries (Chapter 5), there may be multiple sources of information used to calculate the thresholds used by our first and third framework. Attack vectors to achieve this effect may include patiently training the anomaly detection systems to trick them to believe the actions of insider attackers are legitimate [55]. Another way to achieve this effect is by colluding with other users to modify the profile. *ii) Rage attacks:* Similarly, users whose previous behavior is exemplary and in a moment of rage decide to perform an attack that requires a single or few accesses that cause a great deal of damage may not be prevented by our proposed solution. We have proposed the use of enabler constraints and geo-social contracts as part of our work; these may help to deter some of these attacks. Albeit, they may not be enough. Therefore, complementary mitigation techniques are necessary to ensure that these types of attacks are prevented.

As it is the case with any interesting problem, there is a great deal of future work that remains to be done in this area. With respect to the prevention of inference of unauthorized information, there are some interesting research directions. As part of this dissertation, we proposed a model that assumes the existence of inference tuples. Inference tuples capture what we believe is a significant portion of inferences. However, there might be more complex inference patterns of interest or there may be multiple insiders colluding to infer unauthorized information. Future research work includes developing approaches to capture more complex inference threats.

With respect to data collection, our frameworks require monitoring users' behavior. This information is fed to a probabilistic system to determine the probability of an access request resulting in an attack. In this dissertation, we assumed that it is possible to obtain and analyze this information without constraints. However, to apply this system in real environments, it is necessary to consider several issues along the legal, privacy and ethical dimensions. Future work may include the design of technical solutions that provide privacy guaranties for users. Additionally, techniques to ensure that geo-social data is not spoofed are needed.

With respect to policy specification, future work includes designing graphical interfaces to specify risk-and-trust policies as well as performing usability studies to help select interfaces that reduce policy specification errors.

Finally, we believe that the methodologies, approaches and analysis presented in this dissertation are important to understand and prevent insider threats. Our proposed research is useful for multiple types of organizations.

APPENDIX

ENTROPY AND PURITY OF CLUSTERING SOLUTIONS

Entropy and Purity are two measures typically used to evaluate the quality of clustering solutions when the ground truth (classes) are known [121]. *Entropy* is a function of the distribution of classes in the resulting clusters. The entropy for each cluster S_r of size n_r is defined as:

$$E(S_r) = -\frac{1}{log(q)} \sum_{i=1}^{q} \frac{n_r^i}{n_r} log \frac{n_r^i}{n_r}$$

where q is the number of classes in the data set, and n_r^i is the number of users of the class i^{th} that were assigned to the r^{th} cluster. The entropy of the entire solution is computed as follows:

$$Entropy = \sum_{r=1}^{k} \frac{n_r}{n} E(S_r)$$

where n is the total number of users in each cluster and k is the number of found clusters. An algorithm that provides a perfect solution, according to the entropy metric, will result in clusters that contain users from a single class, in which case $Entropy = 0$. The smaller the entropy the better.

Purity is a function of the relative size of the largest class in the resulting clusters. The purity of cluster S_r is defined as

$$P(S_r) = \frac{1}{n_r} max_i(n_r^i)$$

which is the number of users of the largest class in a cluster divided by the cluster size. The total purity of a clustering solution is the weighted average of the clusters' purities:

$$Purity = \sum_{r=1}^{k} \frac{n_r}{n} P(S_r)$$

A higher purity represents a better solution.

BIBLIOGRAPHY

[1] Opengis simple features specification for sql, technical report ogc 99-049. Technical report, OpenGIS Consortium, 1999.

[2] American Medical Association, http://www.ama-assn.org, 2015.

[3] Jung: Java universal network/graph framework. http://jung.sourceforge.net, 2015.

[4] J. O. Aagedal, F. d. Braber, T. Dimitrakos, B. A. Gran, D. Raptis, and K. Stølen. Model-based risk assessment to improve enterprise security. In *Proceedings of the 6th International Enterprise Distributed Object Computing Conference*, Washington, DC, USA, 2002. IEEE Computer Society.

[5] J. Adibi, H. Chalupsky, E. Melz, A. Valente, et al. The kojak group finder: Connecting the dots via integrated knowledge-based and statistical reasoning. In *Proceedings of the national conference on Artificial Intelligence*, pages 800–807. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[6] G.-J. Ahn and R. Sandhu. Role-based authorization constraints specification. *ACM Trans. Inf. Syst. Secur.*, 3:207–226, November 2000.

[7] C. Alberts, S. Behrens, R. Pethia, and W. Wilson. Operationally critical threat, asset, and vulnerability evaluation (octave), 1999.

[8] B. Aziz, S. N. Foley, J. Herbert, and G. Swart. Reconfiguring role based access control policies using risk semantics. In *Journal of High Speed Networks: Special Issue on Managing Security Policies, Modelling Verification and Configuration*, 2006.

[9] S. R. Band, D. M. Cappelli, L. F. Fischer, A. P. Moore, E. D. Shaw, and R. F. Trzeciak. Comparing insider it sabotage and espionage: A model-based analysis, 2006.

[10] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[11] N. Baracaldo and J. Joshi. A trust-and-risk aware rbac framework: tackling insider threat. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 167–176, New York, NY, USA, 2012. ACM.

[12] N. Baracaldo and J. Joshi. An adaptive risk management and access control framework to mitigate insider threats. *Computers & Security*, 39:237–254, 2013.

[13] N. Baracaldo and J. Joshi. Beyond accountability: using obligations to reduce risk exposure and deter insider attacks. In *Proceedings of the 18th ACM symposium on Access control models and technologies*, pages 213–224. ACM, 2013.

[14] N. Baracaldo, B. Palanisamy, and J. Joshi. Geo-social-rbac: A location-based socially aware access control framework. In *In Proc. of the 8th International Conference on Network and System Security (NSS 2014)*, NSS '14. Springer, 2014.

[15] D. Basin, S. J. Burri, and G. Karjoth. Optimal workflow-aware authorizations. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, pages 93–102. ACM, 2012.

[16] L. Bauer, S. Garriss, and M. K. Reiter. Detecting and resolving policy misconfigurations in access-control systems. *ACM Transactions on Information and System Security (TISSEC)*, 14, 2011.

[17] L. Bauer, Y. Liang, M. K. Reiter, and C. Spensky. Discovering access-control misconfigurations: New approaches and evaluation methodologies. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*, pages 95–104. ACM, 2012.

[18] M. Beiter, M. C. Mont, L. Chen, and S. Pearson. End-to-end policy based encryption techniques for multi-party data management. *Computer Standards & Interfaces*, 36(4):689 – 703, 2014. Security in Information Systems: Advances and new Challenges.

[19] D. Bell. The bell-lapadula model. *Journal of computer security*, 4(2):3, 1996.

[20] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca. Geo-rbac: a spatially aware rbac. In *Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 29–37. ACM, 2005.

[21] E. Bertino, E. Terzi, A. Kamra, and A. Vakali. Intrusion detection in rbac-administered databases. In *Computer Security Applications Conference, 21st Annual*, 2005.

[22] C. Bettini, S. Jajodia, X. Wang, and D. Wijesekera. Obligation monitoring in policy management. In *Policies for Distributed Systems and Networks, 2002. Proc. 3rd International Workshop on*, pages 2 –12, 2002.

[23] J. Biskup. History-dependent inference control of queries by dynamic policy adaption. In *Proceedings of the 25th annual IFIP WG 11.3 conference on Data and applications security and privacy*, DBSec'11, pages 106–121, Berlin, Heidelberg, 2011. Springer-Verlag.

[24] D. Boucher and P. Kelly. *The social contract from Hobbes to Rawls.* Routledge, 2003.

[25] O. Brdiczka, J. Liu, B. Price, J. Shen, A. Patil, R. Chow, E. Bart, and N. Ducheneaut. Proactive insider threat detection through graph learning and psychological context. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, pages 142–149, May 2012.

[26] A. Brodsky, C. Farkas, and S. Jajodia. Secure databases: constraints, inference channels, and monitoring disclosures. *Knowledge and Data Engineering, IEEE Transactions on*, 12(6):900 –919, nov/dec 2000.

[27] J. Buford, L. Lewis, and G. Jakobson. Insider threat detection using situation-aware mas. In *Information Fusion, 2008 11th International Conference on*, pages 1–8, 2008.

[28] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. A semantic web based framework for social network access control. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 177–186. ACM, 2009.

[29] CERT. 2014 us state of cybercrime survey. https://resources.sei.cmu.edu/asset_files/Presentation/2014_017_001_298322.pdf, 2014.

[30] S. Chakraborty and I. Ray. Trustbac: integrating trust relationships into the rbac model for access control in open systems. In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, SACMAT '06, pages 49–58, New York, NY, USA, 2006. ACM.

[31] S. M. Chandran and J. B. Joshi. Lot-rbac: A location and time-based rbac model. In *Web Information Systems Engineering–WISE 2005*, pages 361–375. Springer, 2005.

[32] L. Chen and J. Crampton. Risk-aware role-based access control. In *Proc. of the 7th International Workshop on Security and Trust Management.*, 2001.

[33] Y. Chen and W. Chu. Protection of database security via collaborative inference detection. *Knowledge and Data Engineering, IEEE Transactions on*, 20(8):1013 –1027, aug. 2008.

[34] J.-H. Cho, A. Swami, and I.-R. Chen. A survey on trust management for mobile ad hoc networks. *Communications Surveys Tutorials, IEEE*, 13:562 –583, 2011.

[35] Y.-S. Cho, A. Galstyan, P. J. Brantingham, and G. Tita. Latent self-exciting point process model for spatial-temporal networks. *arXiv preprint arXiv:1302.2671*, 2013.

[36] O. Chowdhury, M. Pontual, W. H. Winsborough, T. Yu, K. Irwin, and J. Niu. Ensuring authorization privileges for cascading user obligations. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 33–44, New York, NY, USA, 2012.

[37] C. Y. Chung, M. Gertz, and K. Levitt. Demids: A misuse detection system for database systems. In *In Proceedings of the Integrity and Internal Control in Information System*, pages 159–178, 1999.

[38] R. Clemen. *Making Hard Decisions: An Introduction to Decision Analysis.* Duxbury Press, 1991.

[39] R. Clement. Chapter 13: Risk attitudes, utility function assessment. In *Making Hard Decisions, An Introduction to Decision Analysis.* Duxbury Press, second edition edition, 1995.

[40] C. D. Corley, D. J. Cook, A. R. Mikler, and K. P. Singh. Text and structural data mining of influenza mentions in web and social media. *International journal of environmental research and public health*, 7(2):596–615, 2010.

[41] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad, and G. D. Abowd. Securing context-aware applications using environment roles. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*, SACMAT '01, pages 10–20, New York, NY, USA, 2001. ACM.

[42] CPN. Cpn tools, 2013. http://cpntools.org/.

[43] J. Crampton and G. Gutin. Constraint expressions and workflow satisfiability. In *Proceedings of the 18th ACM symposium on Access control models and technologies*, pages 73–84. ACM, 2013.

[44] T. Das, R. Bhagwan, and P. Naldurg. Baaz: A system for detecting access control misconfigurations. In *USENIX Security Symposium*, pages 161–176, 2010.

[45] H. S. Delugach and T. H. Hinke. Using conceptual graphs to represent database inference security analysis. *Jour. Computing and Info. Tech.*, 4(4):291–307, 1994.

[46] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody. Using trust and risk in role-based access control policies. In *In Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies SACMAT'04*. ACM Press, 2004.

[47] N. Einwechter. Preventing and detecting insider attacks using ids, Nov. 3 2002.

[48] D. Eppstein and J. Wang. A steady state model for graph power laws. *arXiv preprint cs/0204001*, 2002.

[49] F. Feng, C. Lin, D. Peng, and J. Li. A trust and context based access control model for distributed systems. In *Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*, HPCC '08, pages 629–634, Washington, DC, USA, 2008. IEEE Computer Society.

[50] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4:224–274, August 2001.

[51] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4:224–274, August 2001.

[52] L. Flynn, G. Porter, and C. DiFatta. Cloud service provider methods for managing insider threats: Analysis phase ii, expanded analysis and recommendations. Technical report, Computer Emergency Response Team (CERT), 2014.

[53] P. W. Fong. Relationship-based access control: protection model and policy language. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 191–202. ACM, 2011.

[54] D. Garg, L. Jia, and A. Datta. Policy auditing over incomplete logs: theory, implementation and applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 151–162, New York, NY, USA, 2011. ACM.

[55] C. Gates, N. Li, Z. Xu, S. Chari, I. Molloy, and Y. Park. Detecting insider information theft using features from file access logs. In *Computer Security - ESORICS 2014*, volume 8713 of *Lecture Notes in Computer Science*, pages 383–400. Springer International Publishing, 2014.

[56] V. D. Gligor and C. S. Chandersekaran. Surviving insider attacks: A call for system experiments. In S. J. Stolfo, S. M. Bellovin, A. D. Keromytis, S. Hershkop, S. W. Smith, and S. Sinclair, editors, *Insider Attack and Cyber Security*, volume 39 of *Advances in Information Security*, pages 153–164. Springer US, 2008. 10.1007/978-0-387-77322-3_9.

[57] T. Grandison and M. Sloman. A survey of trust in internet applications, 2000.

[58] F. Greitzer, D. Frincke, and Z. M. Social/ethical issues in predictive insider threat monitoring, 2011.

[59] F. Greitzer and R. Hohimer. Modeling human behavior to anticipate insider attacks, 2011.

[60] F. Greitzer, L. Kangas, C. Noonan, A. Dalton, and R. Hohimer. Identifying at-risk employees: Modeling psychosocial precursors of potential insider threats. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 2392–2401, 2012.

[61] F. Greitzer, P. Paulson, K. L., L. Franklin, T. Edgar, and F. D. Predictive modeling for insider threat mitigation, 2009.

[62] F. L. Greitzer and T. A. Ferryman. Methods and metrics for evaluating analytic insider threat tools. In *Proceedings of the 2013 IEEE Security and Privacy Workshops*, SPW '13, pages 90–97, Washington, DC, USA, 2013. IEEE Computer Society.

[63] A. Gupta, M. S. Kirkpatrick, and E. Bertino. A formal proximity model for rbac systems. *Computers & Security*, 2013.

[64] M. Hosenball. Nsa chief says snowden leaked up to 200,000 secret documents. http://www.reuters.com/article/2013/11/14/us-usa-security-nsa-idUSBRE9AD19B20131114, 2013.

[65] IBM. Resource access control facility (racf), 2012. http://www-03.ibm.com/systems/z/os/zos/features/racf/.

[66] K. Irwin, T. Yu, and W. Winsborough. Assigning responsibility for failed obligations. In *Trust Management II*, pages 327–342. Springer Boston, 2008.

[67] K. Irwin, T. Yu, and W. H. Winsborough. On the modeling and analysis of obligations. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 134–143, New York, NY, USA, 2006. ACM.

[68] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, Sept. 1999.

[69] K. Jayaraman, V. Ganesh, M. Tripunitara, M. Rinard, and S. Chapin. Automatic error finding in access-control policies. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 163–174, New York, NY, USA, 2011. ACM.

[70] K. Jensen. Coloured petri nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties*, volume 254 of *Lecture Notes in Computer Science*, pages 248–299. Springer Berlin / Heidelberg, 1987. 10.1007/BFb0046842.

[71] A. Jsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision, 2006.

[72] S. Kandala, R. Sandhu, and V. Bhamidipati. An attribute based framework for risk-adaptive access control models. In *Proceedings of the 2011 Sixth International Conference on Availability, Reliability and Security*, ARES '11, pages 236–241, Washington, DC, USA, 2011. IEEE Computer Society.

[73] M. Kandias, V. Stavrou, N. Bozovic, L. Mitrou, and D. Gritzalis. Can we trust this user? predicting insider's attitude via youtube usage profiling. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 347–354. IEEE, 2013.

[74] S. Kaplan and B. J. Garrick. On the quantitative definition of risk. *Risk analysis*, 1(1):11–27, 1981.

[75] S. Kenny and L. Korba. Applying digital rights management systems to privacy rights management. *Computers & Security*, 21(7):648–664, 2002.

[76] M. S. Kirkpatrick, M. L. Damiani, and E. Bertino. Prox-rbac: a proximity-based spatially aware rbac. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 339–348. ACM, 2011.

[77] J. M. Kleinberg. Navigation in a small world. *Nature*, 406(6798):845–845, 2000.

[78] J. Kubica, A. Moore, J. Schneider, and Y. Yang. Stochastic link and group detection. In *Proceedings of the national conference on Artificial Intelligence*, pages 798–806. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[79] N. Li, H. Chen, and E. Bertino. On practical specification and enforcement of obligations. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*, CODASPY '12, pages 71–82, New York, NY, USA, 2012. ACM.

[80] J. Ma, K. Adi, M. Mejri, and L. Logrippo. Risk analysis in access control systems. In *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, pages 160 –166, aug. 2010.

[81] A. Moore, D. Cappelli, and T. R. The "big picture" of insider it sabotage across u.s. critical infrastructures, 2008. CERT, http://www.cert.org/insider_threat.

[82] L. Mui and M. Mohtashemi. A computational model of trust and reputation. In *In Proceedings of the 35th Hawaii International Conference on System Science (HICSS)*, 2002.

[83] R. Murray-Webster et al. *Management of risk: guidance for practitioners.* The Stationery Office, 2010.

[84] N. Nissanke and E. 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.

[85] NIST. 2010 economic analysis of role-based access control, 2010.

[86] U. D. of Health and H. Services. The health insurance portability and accountability act (hipaa), 1996.

[87] D. Olmedilla, O. F. Rana, B. Matthews, and W. Nejdl. Security and trust issues in semantic grids. In *In Proceedings of the Dagsthul Seminar, Semantic Grid: The Convergence of Technologies*, page 05271, 2005.

[88] Oracle. Application access controls governor, 2012. http://www.oracle.com/us/solutions/corporate-governance/access-controls/index.html.

[89] G. K. Palshikar and M. M. Apte. Collusion set detection using graph clustering. *Data Mining and Knowledge Discovery*, 16:135–164, 2008.

[90] J. Park and R. Sandhu. The uconabc usage control model. *ACM Trans. Inf. Syst. Secur.*, pages 128–174, 2004.

[91] S. Pearson and A. Charlesworth. Accountability as a way forward for privacy protection in the cloud. In M. Jaatun, G. Zhao, and C. Rong, editors, *Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science*, pages 131–144. Springer Berlin Heidelberg, 2009.

[92] S. Perreault and S. Brennan. Criminal victimization in canada, 2009, 2010.

[93] R. Project. Package for hierarchical clustering with p-values (pvclust), 2012.

[94] I. Ray, M. Kumar, and L. Yu. Lrbac: a location-aware role-based access control model. In *Information Systems Security*, pages 147–161. Springer, 2006.

[95] Q. M. S. Osborn, R. Sandhu. Configuring role-based access control to enforce mandatory and discretionary access control policies. In *ACM Transaction on Information and System Security*, 2000.

[96] A. S. Sabau et al. Survey of clustering based financial fraud detection research. *Informatica Economica*, 16(1):110–122, 2012.

[97] M. B. Salem, S. Hershkop, and S. J. Stolfo. A survey of insider attack detection research. In *Insider Attack and Cyber Security*, pages 69–90. Springer, 2008.

[98] M. B. Salem and S. J. Stolfo. Modeling user search behavior for masquerade detection. In *Recent Advances in Intrusion Detection*, pages 181–200. Springer, 2011.

[99] 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.

[100] F. Salim, J. Reid, U. Dulleck, and E. Dawson. Budget-aware role based access control. *Computers and Security*, 2012.

[101] R. Sandhu. Role activation hierarchies. In *In Proceedings of 3rd ACM Workshop on Role-Based Access Control*, pages 33–40. ACM, 1998.

[102] SAP. Access risk management, 2012. www.sap.com/solutions/sapbusinessobjects/large/governance-risk-compliance/accessandauthorization/index.epx.

[103] R. Shaikh, K. Adi, and L. Logrippo. Dynamic risk-based decision methods for access control systems. *Computers and Security*, 31(4):447 – 464, 2012.

[104] A. Sharifi, P. Bottinelli, and M. V. Tripunitara. Property-testing real-world authorization systems. In *Proceedings of the 18th ACM symposium on Access control models and technologies*, pages 225–236. ACM, 2013.

[105] H. Shimodaira. Approximately unbiased tests of regions using multistep-multiscale bootstrap resampling, 2004.

[106] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 422–431, New York, NY, USA, 2005. ACM.

[107] S. D. Stoller, P. Yang, C. R. Ramakrishnan, and M. I. Gofman. Efficient policy analysis for administrative role based access control. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS'07, pages 445–455, New York, NY, USA, 2007. ACM.

[108] G. Stoneburner, A. Goguen, and A. Feringa. Risk management guide for information technology systems, recommendations of the national institute of standards and technology, 2002.

[109] B. Systems. Identity and access governance, 2012. http://www.betasystems.com/en/portfolio/identityaccessgovernance/index.html.

[110] E. Tarameshloo and P. Fong. Access control models for geo-social computing systems. In *SACMAT*, 2014.

[111] P. J. Taylor, C. J. Dando, T. C. Ormerod, L. J. Ball, M. C. Jenkins, A. Sandham, and T. Menacere. Detecting insider threats through language change. *Law and human behavior*, 37(4):267, 2013.

[112] C. I. T. Team. Unintentional insider threats: A foundational study. Technical report, 2013.

[113] R. K. Thomas. Team-based access control (tmac): a primitive for applying role-based access controls in collaborative environments. In *Proceedings of the second ACM workshop on Role-based access control*, pages 13–19. ACM, 1997.

[114] M. Toahchoodee, I. Ray, and R. M. McConnell. Using graph theory to represent a spatio-temporal role-based access control model. *International Journal of Next-Generation Computing*, 1(2), 2010.

[115] J. S. Valacich, J. L. Jenkins, J. F. Nunamaker Jr, S. Hariri, and J. Howie. Identifying insider threats through monitoring mouse movements in concealed information

tests. In *HICSS-46 Symposium on Credibility Assessment and Information Quality in Government and Business*, 2013.

[116] G. T. Wickramaarachchi, W. H. Qardaji, and N. Li. An efficient framework for user authorization queries in rbac systems. In *Proc. of the 14th ACM SACMAT technologies*, SACMAT '09, pages 23–32. ACM, 2009.

[117] G. T. Wickramaarachchi, W. H. Qardaji, and N. Li. An efficient framework for user authorization queries in rbac systems. In *Proc. of the 14th ACM SACMAT technologies*, SACMAT '09, pages 23–32. ACM, 2009.

[118] R. Yip and E. Levitt. Data level inference detection in database systems. In *Computer Security Foundations Workshop, 1998. Proceedings. 11th IEEE*, pages 179 –189, jun 1998.

[119] Y. Zhang and J. B. D. Joshi. Uaq: a framework for user authorization query processing in rbac extended with hybrid hierarchy and constraints. In *Proc. of the 13th ACM SACMAT*, SACMAT '08, pages 83–92. ACM, 2008.

[120] G. Zhao, D. Chadwick, and S. Otenko. Obligations for role based access control. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 01*, AINAW '07, pages 424–431, Washington, DC, USA, 2007. IEEE Computer Society.

[121] Y. Zhao and G. Karypis. Criterion functions for document clustering experiments and analysis. *Mach. Learn.*, 55(3):311–331, June 2002.