Design and Management of Collaborative Intrusion Detection Networks

by

Carol Fung

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In recent years network intrusions have become a severe threat to the privacy and safety of computer users. Recent cyber attacks compromise a large number of hosts to form botnets. Hackers not only aim at harvesting private data and identity information from compromised nodes, but also use the compromised nodes to launch attacks such as distributed denial-of-service (DDoS) attacks.

As a counter measure, Intrusion Detection Systems (IDS) are used to identify intrusions by comparing observable behavior against suspicious patterns. Traditional IDSs monitor computer activities on a single host or network traffic in a sub-network. They do not have a global view of intrusions and are not effective in detecting fast spreading attacks, unknown, or new threats. In turn, they can achieve better detection accuracy through collaboration. An Intrusion Detection Network (IDN) is such a collaboration network allowing IDSs to exchange information with each other and to benefit from the collective knowledge and experience shared by others. IDNs enhance the overall accuracy of intrusion assessment as well as the ability to detect new intrusion types.

Building an effective IDN is however a challenging task. For example, adversaries may compromise some IDSs in the network and then leverage the compromised nodes to send false information, or even attack others in the network, which can compromise the efficiency of the IDN. It is, therefore, important for an IDN to detect and isolate malicious insiders. Another challenge is how to make efficient intrusion detection assessment based on the collective diagnosis from other IDSs. Appropriate selection of collaborators and incentive-compatible resource management in support of IDSs' interaction with others are also key challenges in IDN design.

To achieve efficiency, robustness, and scalability, we propose an IDN architecture and especially focus on the design of four of its essential components, namely, trust management, acquaintance management, resource management, and feedback aggregation. We evaluate our proposals and compare them with prominent ones in the literature and show their superiority using several metrics, including efficiency, robustness, scalability, incentive-compatibility, and fairness. Our IDN design provides guidelines for the deployment of a secure and scalable IDN where effective collaboration can be established between IDSs.

Acknowledgements

From the formative stages of this thesis, to the final draft, I owe an immense debt of gratitude to my supervisor, Dr. Raouf Boutaba. His sound advice and strong support were invaluable during my entire PhD program.

I would also like to thank my colleagues Jie Zhang and Quanyan Zhu, for, without their time and cooperation, many projects would not have been possible.

Finally, I like to thank all who have participated in my projects for their contributions, including, Isam, Olga, Disney, Jiyong, Shameel, and Shaylan.

To each of the above, I extend my deepest appreciation.

Dedication

For Jerome and Timothy, my immense courage and strength.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

In November 2008, a new type of computer worm started to spread quickly. It used three different types of attack on Windows hosts: exploiting vulnerabilities, guessing passwords, and infecting removable devices [14]. In three months it took over about 9 million Microsoft Windows systems around the world and formed a massive botnet [3]. The estimated economic loss brought by this worm was USD 9.1 billion [24]. The worm was named "Conficker", and it was only one of the thousands of other worms that appear every year.

Nowadays the vast majority of computers are connected to the Internet. A number of applications used by billions of users on a day-to-day basis including email, web-browsing, video/audio streaming, social networking, online gaming, e-commerce, and online chatting rely on the Internet. At the same time, *network intrusions* have become a severe threat to the privacy and safety of computer users. Each year billions of malicious *cyber attacks* are reported [50, 115]. Attacks are becoming more sophisticated and stealthy, driven by an "underground economy" [51]. By definition, network intrusions are unwanted traffic or computer activities that may be malicious or destructive, including viruses, worms, trojan horses, port scanning, password guessing, code injection, and session hijacking. The consequences of a network intrusion can be user identity theft (ID theft), unwanted advertisement and commercial emails (spam), the degradation or termination of the host service (denial of service), or using fraudulent sources to obtain sensitive information from users (phishing). Network intrusions are usually accomplished with the assistance of malicious code (a.k.a. malware). In recent years, network intrusions have become more sophisticated and organized. Attackers can control a large number of compromised hosts/devices to form Botnets [3], and then launch organized attacks, such as Distributed Denial of Service (DDoS)attacks.

As a countermeasure, *Intrusion Detection Systems (IDSs)* are used to identify intrusions by comparing observable behavior against suspicious patterns [67]. Based on the technology used for detection, IDSs can be categorized as signature-based or anomaly-based. Based on the targets they are monitoring, they can be host-based or network-based. Examples of IDSs include antivirus software [19, 2], Snort [17], Bro [4], Tripwire [21], OSSEC [13], and HoneyNets [20]. Traditional IDSs monitor computer activities on a single host, or monitor network traffic in a sub-network. They do not have a global (i.e., Internet-wide) view of intrusions and are not effective in detecting fast-spreading attacks. In addition, traditional IDSs acquire detection rules only from their corresponding vendors. Various security vendors usually employ distinct intrusion detection technologies and knowledge. In practice, no single security vendor has the entire knowledge to detect all types of intrusions. Therefore, traditional IDSs are not effective in detecting unknown or new threats. In turn, they can achieve better detection accuracy through collaboration. A good example for this is anti-virus software, where it is common knowledge that a malware file that has not been detected by one antivirus software may be detected by another. If IDSs are allowed to communicate with each other and exchange intrusion information, each IDS can benefit from the collective expertise of the others. Therefore, collaboration between IDSs is envisioned to be a promising approach to improve intrusion detection.

Some early works on IDS collaboration include Indra [67] and DOMINO [120], where IDSs shared information to prevent fast-spreading attacks, and IA-NSM [33], where intelligent agents (such as IDS) exchange observations to detect attacks such as doorknob rattling [63] and IP spoofing [104]. However, their collaboration was limited to selected nodes that followed predefined communication protocols such as DOMINO. Later, in 2008, standardized models and communication protocols provided a method for various IDSs to communicate with each other. The two important standards are IDMEF (Intrusion Detection Message Exchange Format) [10] and CIDSS (Common Intrusion Detection Signatures Standard) [6]. IDMEF provides a communication standard enabling different intrusion detection analyzers from different origins (commercial, open-source, and research systems) to report to a managing entity for data analysis, aggregation, correlation, etc. It is XML-based and includes two types of messages: heartbeat messages sent periodically to state that an IDS in the distributed system is still alive, and alert messages sent when a suspicious event occurs. Those events can be augmented with additional information in the form of XML compound classes such as the scanner type, timestamps, and classifications in the case of an alert, or even self-defined attributes (see Appendix A). The IDMEF is specified in RFC4765 [15] and implemented by many IDSs such as Snort and OSSEC. CIDSS defines a common XML-based data format for storing signatures from different intrusion detection systems and shares the signatures amongst them. In this way, it is primarily aimed at IDS administrators to exchange, evaluate, and criticize signatures. Also, a future scenario is considered in which independent contributors exist, enabling the provision of signatures independent of a particular product or software.

The standardization of communication protocols between different IDSs allows each IDS to obtain intrusion information and detection knowledge from other IDSs in the network. An *Intrusion Detection Network* (IDN) is such a collaboration network, allowing IDSs to exchange information with each other and to benefit from the collective knowledge and experience shared by others. IDNs enhance the overall accuracy of intrusion assessment as well as the ability to detect new intrusion types. There are two types of IDNs in the literature: information-based and consultation-based. In an *information-based IDN*, nodes share observations and detection knowledge with other nodes in the network, such as observations related to new attacks. This type of IDN is effective in detecting fast-spreading attacks such as worms. However, it may generate large communication overhead, and not all exchanged information may be useful to others. In a *consultation-based IDN*, when an IDS detects suspicious activities but does not have enough confidence to make a decision, it may send *consultation requests* to others in the network. *Feedback* from the collaborators can be used to make a final decision as to whether it is an intrusion or not. Consultation-based IDNs have much lower communication overhead, are more effective in terms of communication efficiency, and are the focus of this thesis.

Although communication and collaboration among IDSs is feasible, building an effective IDN is a challenging task. For example, adversaries may compromise some IDSs in the network and then leverage the compromised nodes to send false information and spam, to free-ride, or even to attack other nodes in the network, which can compromise the efficiency of the IDN. It is therefore important for an IDN to detect and isolate malicious insiders. Another challenge is how to make efficient intrusion detection assessments based on the collective information and knowledge from other IDSs. Appropriate selection of IDN participants and incentive-compatible resource management in support of IDS interactions with peers are also key challenges in IDN design.

This thesis focuses on the design of IDNs leveraging effective and efficient collaboration between participant IDSs. We emphasize "collaboration" from the perspective of an IDS to provide a systematic approach for determining who to collaborate with and how to make intrusion detection decisions based on collective knowledge. The thesis will answer the following questions: why build intrusion detection networks; what are the problems underlying the design of intrusion detection networks; and what are the solutions to those problems? We overview existing IDN designs and discuss the underlying challenges, including privacy, malicious insiders, scalability, free-riders, collaboration incentives, and intrusion detection efficiency.

Privacy is important since IDN users may be discouraged to participate in IDNs if there is potential information breaching during collaboration. In an IDN, participating IDSs can be malicious. A trust management framework is required to identify dishonest or malicious insiders. Our research results [56, 57, 53] show that an efficient trust management system can effectively identify malicious/dishonest or incapable IDSs in the network, thus improving the quality of collaboration by eliminating the impact of malicious IDSs. In particular, we present in chapter 4 a Bayesian-learning-based trust management model where each participanting IDS evaluates the trustworthiness of its collaborators through past experiences with them. A Dirichlet model is described as a means to integrate past experiences and calculate trust values as well as the confidence levels in the trust estimation.

Another important problem pertaining to IDS collaboration in an IDN is how IDSs use others' opinions to make a decision. The problem for IDSs in the IDN is to determine whether to raise an intrusion alarm or not, based on the feedback from collaborators. In our work [58], we consider two types of false decision cost: false positive cost and false negative cost. We model the risk cost of decisions using a Bayesian hypothesis model and choose the decision which has the lower risk cost. In our extended work [129], we further investigate the minimum amount of feedback an IDS needs to achieve a low enough cost. Chapter 5 first discusses how Bayesian decision models can be used to make optimal intrusion decisions which have minimal false decision cost, and how sequential hypothesis models can be used to decide the smallest list of collaborators to consult in order to achieve a decision satisfying a given confidence level. The optimal decision model is used to compare the expected costs of raising or not raising an intrusion alarm, and then to choose the decision which bears the lowest cost. The sequential hypothesis model is used to find the minimal number of collaborators to consult before a confident decision is made, which can effectively reduce the amount of communication overhead between IDSs.

Once collaboration connections are established, determining how many resources are required for each collaborator in order to maintain a fair, incentive-compatible, and with no-free-rider collaboration environment is another interesting research question. We adopt a game-theoretic approach to model the resource allocation strategy of IDN participants [133, 132]. Specifically, as shown in Chapter 6, the nodes in the IDN are modeled as a set of uncooperative game players, and all the nodes follow a predefined strategy to play the game. The game strategy is for each node to decide how to allocate resources to their neighbors fairly. We prove that the game has a Nash Equilibrium (NE), and under the NE the amount of help received by each node is proportional to the amount of its contribution to others. Free-riding is thus not practical under this resource allocation design.

In a dynamic IDS collaboration environment, participating IDSs may join, leave the network, or become compromised. How to select and maintain collaborators effectively is of paramount importance. We formulate the acquaintance selection as an optimization problem [54, 55] where an optimal collaborator set should lead to minimal false decision and maintenance costs. In Chapter 7, we describe a collaborator management model which allows each IDS to select the best combination of collaborators to minimize its cost. Since the optimal selection of collaborators is a NP hard problem, heuristic approaches are sought to find near-optimal solutions.

As discussed above, this thesis not only discusses efficient IDN design, it also provides a collection of solutions to key IDN design challenges and shows how various theoretical tools can be used in this context. Another highlight of this thesis is the comprehensive evaluation of IDN designs, including various evaluation metrics; e.g., efficiency of intrusion detection, robustness against malicious insiders, fairness and incentive-compatibility for all participants, and scalability in network size.

This thesis is organized as follows. Chapter 2 presents an overview of network intrusions, their potential damage, and corresponding detection methods. We then introduce Intrusion Detection Systems and Intrusion Detection Networks. Chapter 3 discusses distributed IDN architecture design. Chapter 4 and chapter 5 are respectively dedicated to trust management and intrusion detection decision-making. Resource management and collaborator management are discussed in Chapter 6 and Chapter 7 respectively. Chapter 8 summarizes and concludes the thesis.

Chapter 2

Background

This chapter provides an overview of cyber intrusions and the methodologies they use. It then discusses intrusion detection systems and intrusion detection networks. Finally, it provides a brief survey of existing intrusion detection networks.

2.1 Overview of Cyber Intrusions

There are many different ways to launch cyber intrusions, including malware infection, software/service vulnerability exploitation, denial of service, or phishing. We describe some of the major types of cyber intrusions and their potential damage in this section.

2.1.1 Malware

A network intrusion accomplishes its goal by executing malicious software/code on the victim machine. *Malware* is a term for all software or code designed to cause damage to a device or a network. There are many different types of malware, such as computer viruses, worms, trojans, or spyware.

A *computer virus* is a computer program that can insert/copy itself into one or more files without the permission or knowledge of the user, and then performs some (possibly null) operations [32]. Malicious viruses may cause a program to run incorrectly or corrupt a computer's memory, while non-malicious viruses may do no harm. A computer can be infected with a virus when copying data from other computers or when using an infected external drive such as a flash memory or removable disk. As their name suggests, viruses can replicate themselves to infect other hosts, but typically do so after user interaction. For instance, a virus received as an email attachment infects the user host when opened by the user and eventually spreads to other hosts by sending the same email to contacts in the user's address book.

In general, most computer viruses do not actively search for victims through a network. Malware which actively searches for victims is known as a *worm*. A computer worm is a program which propagates itself through the network automatically by exploiting security flaws in widely-used services [113]. Worms can cause the most extensive and widespread damage of all types of computer attacks because of their automatic spreading capability. A large number of different worms have been documented over the years. Some of the most famous ones include Morris (1988), CodeRed (2001), SQL Slammer (2003), the Witty worm (2004), the Conficker worm (2009), and Stuxnet (2010).

A distinguishing characteristic of computer viruses and worms is their ability to self-replicate and spread within networks. There are some other types of harmful software/code which do not self-replicate, such as Trojan horses (trojans). A trojan (also called a back-door) is a program with an overt (documented or known) effect and a covert (undocumented or unexpected) effect [32]. For many years, trojans have been the most widely used source of malware by hackers [90]. Trojans appear to perform desirable functions, but in fact facilitate unauthorized access to users' computers. A typical trojan requires interactions with a hacker. Hackers can access the infected hosts and manipulate the hosts using commands.

Finally, *spyware* is a type of malware that is installed surreptitiously on a personal computer to collect information about the user without their informed consent, such as their browsing habits. Spyware can report user information to the attacker, such as email addresses, credit card information, bank account information, passwords, and other sensitive information. The difference between spyware and trojans is that spyware aims at collecting information from users and a trojan allows hackers to access the infected host.

2.1.2 Vulnerability and Service Attacks

In the past few years, a plethora of services and applications have become available online and accessible by users worldwide. However, due to the increasing size and complexity of these services and applications, design and implementation flaws are commonplace, making them vulnerable to attackers. A *software vulnerability* is a weakness in a computer program which can be exploited by an attacker and used to gain unauthorized access or to degrade service performance. There are thousands of software vulnerabilities discovered and documented each year in vulnerability databases such as the National Vulnerability Database [12] and US-CERT [22]. An exploitable vulnerability is the combination of three elements: a system flaw, attackers' access to the flaw, and attackers' capability to exploit the flaw. To exploit a vulnerability, an attacker must have at least one applicable tool or technique that allow to connect to a system weakness.

A vulnerability that is unknown or freshly discovered and not yet patched by system developers is called a *zero-day vulnerability.* Attacks which are targeted at a zero-day vulnerability are called *zero-day attacks.* Zero-day attacks occur during the vulnerable time window that exists between the time the vulnerability is known to attackers and when software developers start to patch and publish a countermeasure.

A typical example of vulnerability is the *buffer overflow*, where attackers can manipulate an already-running program to overrun the buffer's boundary and overwrite its adjacent memory, and eventually cause the program to execute the attacker's code. A buffer overflow can be triggered by injecting malicious code through inputs when running the program. Attackers can take advantage of the buffer overflow vulnerability of a service to crash the service or run malware.

2.1.3 Web-based Attacks

Although malware is a very popular way to attack computers or devices on the Internet, it usually requires victims to receive and run malicious code [40], which can be avoided by careful Internet users. Web-based attacks are another type of attack on Internet users and web services. Typical examples of web-based attacks include *SQL-injection* and *cross-site-scripting* [40].

SQL-injection is a way to exploit a type of vulnerability known as a *command injection vulnerability*. Typically, SQL-injection arises when untrusted data is inserted for malicious purposes into a query or command to a web service. SQL-injection attacks can be used to retrieve information from compromised web services and thereby cause information breaches. Information such as social security numbers, dates of birth, and maiden names are collected by hackers, as part of *identity theft*. Another popular target of this type of attack is unprotected credit card information. Massive credit card information loss can cause significant damage to an organization's most valued asset, its customers. Solutions to mitigate the impact of SQL-injection attacks include applying data validation, encrypting sensitive data in the database, and limiting privileges [40], among others. SQL-injection attacks can be detected through anomaly detection methods (see Section 2.2) employed by intrusion detection systems (IDS).

Cross-site-scripting (XSS) lies in the category of cross-domain security issues [40]. This type of attack takes advantage of security vulnerabilities found in web applications, such as web browsers. It allows attackers to inject client-side script into web pages and retrieve the session data of the user. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same origin security policy. Cross-site scripting carried out on websites accounted for roughly 84% of all security vulnerabilities documented by Symantec, as of 2007 [108]. Solutions to prevent XSS attacks include input validation and output sanitization, the usage of HTTP-only cookies, and binding session cookies to IP addresses [40].

2.1.4 Organized Attacks and Botnets

Recent network intrusions have evolved to be more sophisticated and organized. Intruders are able to control a group of compromised computers/devices to launch distributed attacks; for example, the *Distributed Denial of Service* (DDoS) attack. Compromised nodes which are infected with malware communicate with a master through a command and control (C&C) server [111]. A group of compromised nodes and a master together form a *Botnet*. The compromised nodes are called *"Bot nodes"*, and the master is called a *"Bot master"*. Bot nodes can be used to commit profit-driven cyber crimes such as DDoS attacks, spam propagation, ID theft, or phishing. A CSI report [94] indicates that in 2008, the financial loss caused by "Bot" computers in US enterprises was the second-highest, following financial fraud. Another report [27] predicted that the percentage of computers compromised into bot nodes worldwide would be 15% in 2009. Cooperative attacks are complex forms of distributed attacks where compromised nodes are organized to play different roles in the entire attack scenario. In the next subsection, we give an example of a botnet, which coordinates compromised nodes to launch phishing attacks.

2.1.5 Spam and Phishing

Spam is the activity of using electronic messaging systems to send unsolicited bulk messages indiscriminately to users, especially for advertising products or services. While the most well known spam is email spam, the term also applies to similar abuses in other media, such as instant messaging spam, social network spam, and spam in blogs.

Spam is a widely used method for spreading malware, delivering advertisements, and posting phishing links. For example, the famous “Love Letter” computer virus (2000) was spread by sending emails with the subject line “I Love You” and the attachment “Love-Letter-For-You.txt.vbs”. When the receivers opened the attached executable file, it then activated the attached script and infected the host machine. The “Love Letter” worm infected more than 50 million users in 10 days and caused at least a 2 billion USD loss worldwide [65].

Another usage of spam emails is to post phishing weblinks. *Phishing* is a criminal activity consisting of stealing users’ personal identity data and financial account credentials. Phishing attacks typically use two mechanisms. The first mechanism, known as social engineering, makes use of spoofed emails appearing to be from legitimate businesses and agencies in order to lead consumers to counterfeit websites designed to trick recipients into divulging personal data such as usernames and passwords. The second mechanism, known as technical subterfuge, plants crimeware onto user computers to steal credentials directly through intelligent keyloggers and/or by corrupting browser navigation in order to mislead customers to counterfeit websites. Gartner estimated an increase in the cost of identity theft from 2 billion USD to 3.2 billion USD in 2007 in the USA alone [66].

Like any large-scale online service, large-scale phishing web sites rely on online availability. Phishing sites, however, may be relatively easy to bring down if they use fixed IP addresses. This is not only specific to phishing sites. In fact, any illegal online organization which targets victims on a large scale requires high availability for the continuation of its operation. Recently, Fast-Flux Service Networks [25] have appeared to fulfill this requirement ensuring a high availability yet evasiveness of illegal sites. Fast-Flux Service Network (FFSN) is a term coined by the anti-spam community to describe a decentralized botnet used to host online criminal activities. FFSNs employ domain name service(DNS) techniques to establish a proxy network on the compromised machines. These compromised machines are used to host illegal online services, like phishing websites, malware delivery sites, etc., with very high availability. An FFSN generally has hundreds or even thousands of IP addresses assigned to it. These IP addresses are swapped in and out of flux with extremely high frequency, using a combination of round-robin IP addresses and a very short Time-To-Live (TTL) for any given particular DNS Resource Record (RR).

Website hostnames may be mapped to a new set of IP addresses as often as every 3 minutes [25]. This makes it extremely hard to take down the actual service launcher, as the control node (mothership) is not known. The proxy agents do the work for the control node, and they also change rapidly. ATLAS is a system from Arbor Networks which identifies and tracks new Fast-Flux Networks [87]. In an investigation conducted in 2008, ibankhalifax.com was the largest detected fast flux domain, with a size of 100,379 hosts and a DNS entry life of two months. When an FFSN is detected, the domain registrars can be contacted to shut down the corresponding domain, hence removing the FFSN. Although this mitigation technique sounds doable, it is often a tedious and time-consuming task given the fact that not all registrars respond to abuse complaints [7].

2.1.6 Mobile Device security

With the rapid advances in the so-called “Internet of Things”, desktop computers are no longer the dominant form of computing. For example, smartphone usage has been growing exponentially and is replacing desktop usage to become the next popular tool for email, news, chatting, and Internet access. Following the growth of smartphone use, smartphone exploitation techniques are also growing. A key feature of modern smartphone platforms is a centralized service for downloading third-party applications. The convenience to users and developers of such an “app market” has led to an explosion in the number of apps available. Apple’s App Store served nearly 3 billion application downloads after only 18 months [26]. Many of these applications combine data from remote cloud services with information from local sources, such as a GPS receiver, camera, microphone, or accelerometer. Applications often have legitimate reasons for accessing this privacy-sensitive data, but users may not be aware of whether their data is used properly or not. Many incidents have occurred where developers relayed private information back to the cloud [41, 86], and the privacy risks illustrate the danger [49].

In addition to the risk of downloading malware, mobile phone vulnerabilities are also targets for exploitation. Hundreds of vulnerabilities were discovered in the years 2009 and 2010 [50]. While it may be difficult to exploit many of these vulnerabilities successfully, there were two vulnerabilities affecting Apple's iPhone iOS operating system that allowed users to *"jailbreak"* their devices. The process of jailbreaking a device through exploits is to install malicious code, which can gain the user root privileges through exploiting a vulnerability of the iOS.

2.2 Intrusion Detection Systems

Intrusion Detection Systems (IDSs) are software/hardware systems designed to monitor network traffic or computer activities and emit alerts/alarms to administrators when suspicious intrusions are detected. Intrusion Detection Systems are different from firewalls. A firewall is a device that filters all traffic between a protected or "internal" network and a less trustworthy or "external" network, while IDSs sniff or monitor network traffic or computer activities but do not drop or block them. A firewall can be used along with an IDS to block identified malicious traffic in order to protect internal computers from being further exploited. Based on the technology used for detection, IDSs can be divided into signature-based and anomaly-based types. Also, based on data sources, they can be host-based or network-based.

2.2.1 Signature-based and Anomaly-based IDSs

Signature-based IDSs compare data packets with the signatures or attributes of known intrusions to decide whether the observed traffic is malicious or not. A signature-based IDS is efficient in detecting known intrusions with monomorphic signatures. However, it is not efficient in detecting unknown intrusions or intrusions with polymorphic signatures.

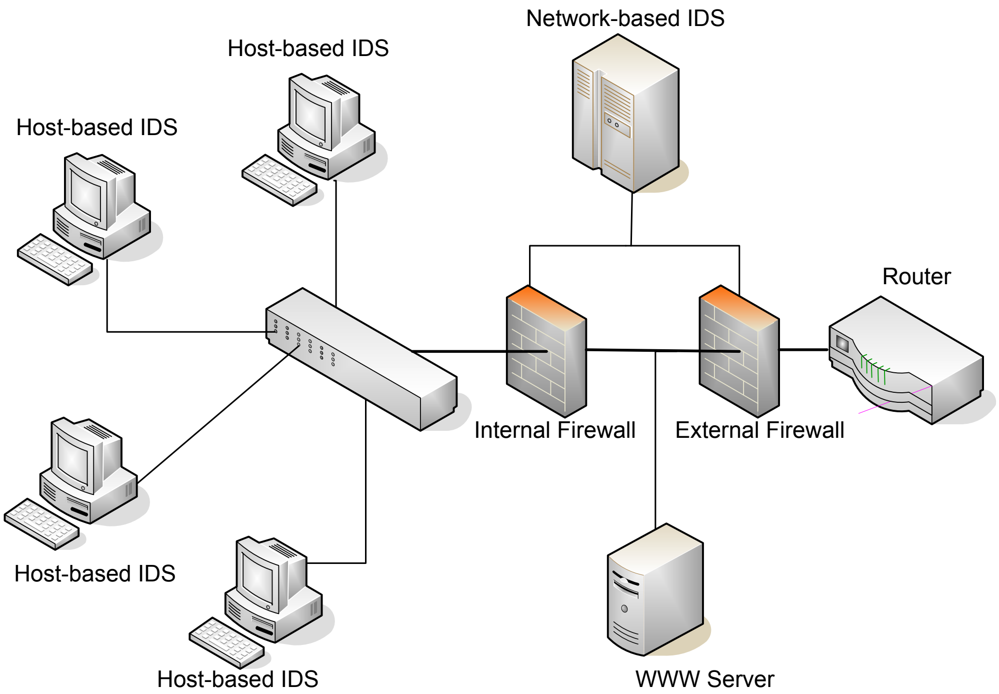


Figure 2.1: An example of host-based IDS and Network-based IDS

Anomaly-based IDSs observe traffic or computer activities and detect intrusions by identifying activities distinct from a user's or system's normal behavior. Anomaly-based IDSs can detect unknown intrusions or new intrusions. However, they usually suffer from a high false positive rate. Most current IDSs employ both techniques to achieve better detection capability.

2.2.2 Host-based and Network-based IDSs

A Host-based IDS (HIDS) runs on an individual host or device in the network (Figure 2.1). It monitors inbound/outbound traffic to/from a computer as well as internal activities such as system calls. A HIDS views an individual device only, and may not be aware of the overall network environment. Examples of HIDSs include OSSEC [13] and Tripwire [21].

Tripwire is a brand of software used to ensure the integrity of critical system files and directories by identifying all changes made to them. Tripwire configuration options include the ability to receive alerts via email if particular files are altered, and automated integrity checking. Using Tripwire for intrusion detection and damage assessment helps in keeping track of system changes and can speed up the recovery from a break-in by reducing the number of files that must be restored to repair the system. Tripwire compares files and directories against a baseline database of file locations, dates modified, and other data. It generates the baseline by taking a snapshot of specified files and directories in a known secure state. After creating the baseline database, Tripwire compares the current system to the baseline and reports any modifications, additions, or deletions.

Network-based IDSs (NIDS) monitor network traffic to/from the network. A NIDS contains sensors to sniff packets, and a data analyzer to process and correlate data. Alarms are raised whenever suspected intrusions are found. However, a NIDS does not have knowledge about the internal activities of individual computers. Examples of NIDSs include Snort [17] and Bro [4].

Snort is a free and open-source network intrusion detection system (NIDS), created in 1998 and developed by Sourcefire. Snort has the ability to perform real-time traffic analysis and packet logging on IP networks. Snort performs protocol analysis, content searching, and content matching. It relies on a set of predefined policies named "Snort rules" to detect suspicious traffic. The rules specify the patterns of potential attacks, including IP addresses, port numbers, protocols, and pattern strings. Snort rules need to be updated frequently to keep up with new attacks. Snort can also be used to detect probes or attacks, including but not limited to operating system fingerprinting attempts, common gateway interfaces, buffer overflows, server message block probes, and port scans.

2.2.3 Other Types of IDSs

Early intrusion detection systems, such as [43, 42] and [77], relied on logging the system activities to spot potential misuses that had occurred. The administrator reviews the outputs of the IDS to find attacks, remove threats, and patch vulnerabilities of the system. Modern IDSs analyze network traffic and/or system logs and perform correlations in real-time before sending alerts/alarms to administrators. Besides the traditional IDSs we listed in previous sections such as Snort, Bro, OSSEC, other applications/devices can be used as intrusion detection systems. In this section, we briefly describe Honeypots, Antiviruse systems, and firewalls.

Honeypots are systems set up for the purpose of trapping attackers/hackers and collecting traces for security analysis. A honeypot generally consists of a computer, data, or a network site that appears to be part of a network but is actually isolated and monitored, and which seems to contain information or a resource of value to attackers. Based on their level of interaction with attackers, honeypots can be divided into low-interaction honeypots and high-interaction honeypots [99]. Low-interaction honeypots are usually emulated services that are frequently requested by attackers. They have limited interaction with attackers since they are not real services. Most emulated services are only restricted to the first few interactions. Examples of such honeypots include Honeyd [9], Spector [18], and KFsensor [8]. High-interaction honeypots imitate real operating systems that host a variety of services and applications. Therefore, an attacker may be allowed to perform many types of attacks on this type of honeypot. An example of such a honeypot is Honeynets [20].

Antivirus systems are software systems which monitor, prevent, detect, and remove malware such as computer worms, viruses, adware, Trojan horses, rootkits, and keyloggers. A variety of strategies are employed by antivirus systems. There is signature-based detection, which involves searching for known patterns of data within executable code, and heuristic-based detection, which can identify new viruses or variants of existing viruses by looking for known malicious code, or slight variations of such code, in files. Some antivirus systems also use anomaly detection techniques to identify malware by running it in a sandbox and analyzing the behavior of a file under execution in order to detect any malicious actions. Examples of antivirus software include Symantec [19], Avira [2], and Avast [1].

A firewall is a network security system that monitors the network's incoming/outgoing traffic and determines whether it should be allowed through or not, based on its firewall rules. A network's firewall is a bridge between the internal network or computer it protects and external (inter)network, such as the Internet. Firewalls can use blacklists to detect and block potential intruders. Firewalls can also work together to achieve higher intrusion detection efficiency [109, 28, 46].

2.2.4 Strength and Limitations of IDSs

Intrusion detection systems are constantly evolving. Research on IDSs began in the 1980s, and products appeared in the 1990s. As new vulnerabilities and attack types become known, IDSs evolve and become more and more sophisticated. Indeed, IDSs are improving continuously and are able to detect an ever-growing number of attacks by including more and more attack signatures and attack models. Recall that IDSs look for known vulnerabilities and weaknesses, either through patterns of known attacks (signature-based) or models of normal behavior (anomaly-based). Whenever new attacks are discovered, the corresponding detection rules/signature are created by the IDS manufacture and distributed to users' IDSs. Many commercial IDSs are quite effective at identifying new attacks.

However, it is difficult for IDSs to detect all potential attacks. Indeed, attackers only need to evade the IDS once to successfully compromise the system, while IDSs need to know all possible attacks to guarantee successful defense. In practice, an IDS vendor has knowledge about some attacks, but no single one knows all attacks.

Another limitation of IDSs is their sensitivity control. It is typically the case that a sensitive IDS raises too many intrusion alerts (most of them are false positive alerts), which makes it difficult for administrators to handle. However, when an IDS is less sensitive, it may miss critical attacks (false negatives) and hence fail to protect networks and hosts. Determining the optimal sensitivity of IDSs is a difficult problem.

2.3 Collaborative Intrusion Detection Networks

A *Collaborative Intrusion Detection Network* (IDN) is an overlay network that connects IDSs so that they can exchange information, such as intrusion alerts, blacklists, signatures, suspicious files, and intrusion detection rules. Several IDNs have been proposed in the past few years. In an IDN, IDSs collect data from distributed peer IDSs and use it to achieve better intrusion detection. In this section, we categorize IDNs using three features, namely cooperation topology, cooperation scope, and specialization. We also provide a taxonomy of some of the most prominent IDNs.

2.3.1 Motivation for IDS Collaboration

Isolated intrusion detection systems rely strictly on security updates from their respective vendors and are vulnerable to new or unknown attacks. Collaboration between IDSs allows each IDS to use collective knowledge from other IDSs to achieve more accurate intrusion detection, which is particularly useful for preventing new attacks. For example, when one IDS detects a new attack, it can alert its collaborators, which then can block similar attacks when they occur. Through knowledge-sharing, collaboration between IDSs intuitively benefits each participating IDS and allows the creation of an IDN with a much stronger intrusion detection capability. Building an effective collaborative IDN, however, raises a number of challenges, which we will discuss next.

2.3.2 Challenges of IDS Collboration

Collaboration among intrusion detection systems has the potential to improve the effectiveness of intrusion detection, since IDSs leverage the collective intrusion detection information received from their collaborators. As such participating IDSs are less likely to be compromised by threats unknown to them. However, IDS collaboration introduces communication overhead in the network. Since collaboration is based on information exchange, each participant receives help from others in the network but also has to spend resources (e.g., CPU, memory, network) to help others in return. Therefore, IDSs with low resource capacity may be constrained in collaboration.

Another challenge for IDNs is that the participating IDSs may become the target of malicious attacks. For example, adversaries may compromise some IDSs in the network and then leverage the compromised nodes to send false information or spam, or even to attack other nodes in the network, which can compromise the efficiency of the collaboration network. Therefore, it is important for an IDN to detect and isolate malicious insiders in order to eliminate their negative impact. In addition, how to make efficient intrusion detection assessments based on the collective information and knowledge from other peers is another challenge. In the following paragraphs, we discuss some of the key challenges in IDN design including privacy, malicious insiders, free-riders, scalability, incentives, and intrusion detection efficiency. Then we present an overview of some of the most prominent IDN designs in the literature.

Privacy is a primary issue, since IDN users can be discouraged from participating in the IDN if there is potential information breaching during collaboration. To address this issue, a trust management model can be used to identify dishonest and malicious nodes. An effective trust management model should be able to distinguish honest nodes from dishonest ones, and high-expertise nodes from low-expertise ones. Free-riding the IDN is another important problem, where selfish nodes (a.k.a., free-riders) exploit the network seeking knowledge from others but do not contribute themselves. To handle this problem, an incentive-compatible resource allocation design can reward active participants and discourage free-riders. A scalable IDN can accommodate a large number of nodes in the network without overburdening any single node. A scalable IDN architecture design is necessary for a large-scale collaboration network. Although IDS collaboration can improve overall intrusion detection accuracy, its efficiency is limited by the quality of the individual intrusion detection systems. Collaboration cannot detect an intrusion that no single IDS in the network can detect. Therefore, improving the intrusion detection accuracy of each IDS is still an essential problem to solve. In our work, we will demonstrate the effectiveness of IDS collaboration and the amount of improvement in terms of detection accuracy over individual IDSs.

2.4 Selected Intrusion Detection Networks

2.4.1 Indra

Indra [67] was one of the first to propose a cooperative intrusion detection system. In the proposed system, host-based IDSs in a local area network take a pro-active approach and send warnings to other trusted nodes about the intruder through a peer-to-peer network. For example, as shown in Figure 2.2, if an attacker compromises node B and then launches attacks from B to hosts in the trusted network, then node C detects the attack from B and multicasts a security warning to its trusted neighbors. Subsequently, if B tries to attack other nodes in the network, it will be repelled right away by the forewarned nodes. Indra is a fully distributed system which is targeted towards local area networks.

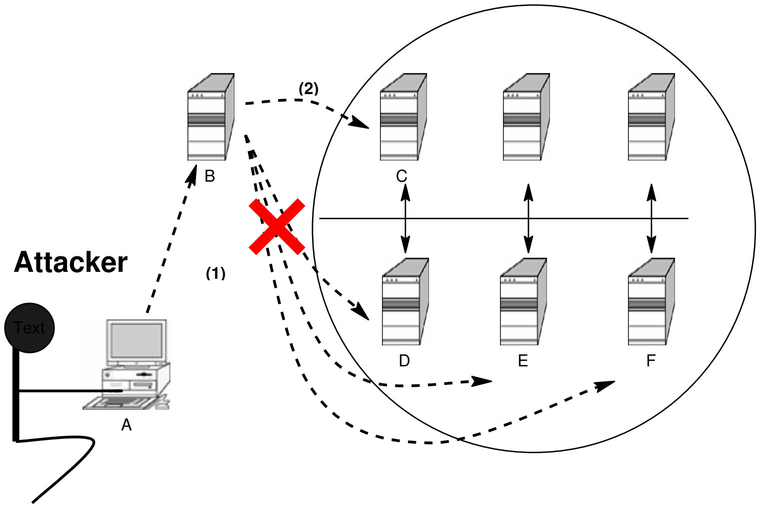


Figure 2.2: Indra Architecture (Adapted from [67])

2.4.2 DOMINO

DOMINO [120] is an IDS collaboration system which aims at monitoring Internet outbreaks at large-scale. In DOMINO (Figure 2.3), heterogeneous IDSs located at diverse locations share their intrusion information with each other. There are typically three types of nodes: axis nodes, satellite nodes, and terrestrial contributors. Satellite nodes are organized hierarchically and are responsible for gathering intrusion data and sending them to parent nodes in the hierarchy. Parent nodes aggregate intrusion data and further forward data up the hierarchy till they reach axis nodes. Axis nodes analyze intrusion data, generate digested summary data and then multicast them to other axis nodes. Network-based IDSs and active sink nodes (such as Honeypot [37]) are integrated into axis nodes to monitor unused IP addresses for incoming worms. Terrestrial contributors do not follow DOMINO protocols but can also contribute to the system through DOMINO access points. In DOMINO, heterogeneous nodes are involved in the cooperation overlay. Information from axis nodes, satellite nodes, and terrestrial contributors is distinguished by different trust levels. This feature enables DOMINO to handle inter-administration-zone cooperation. DOMINO is a decentralized system organized in a hierarchical structure for better scalability.

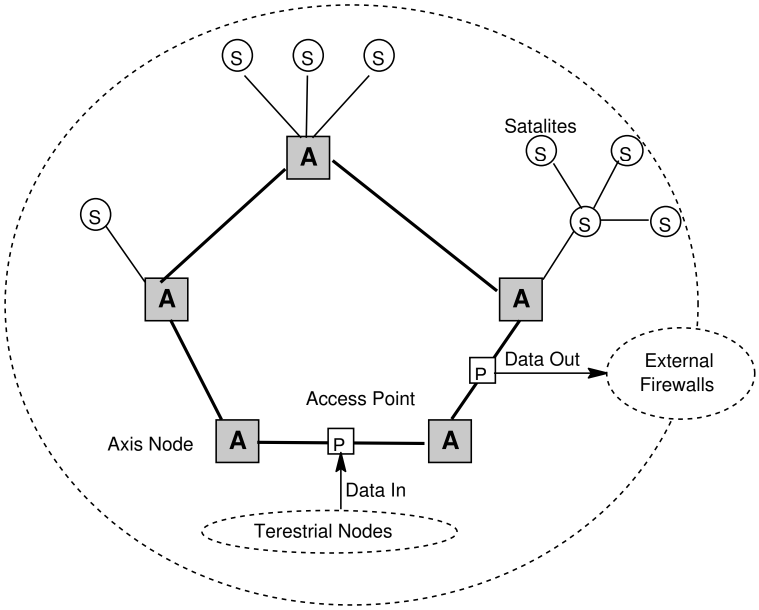


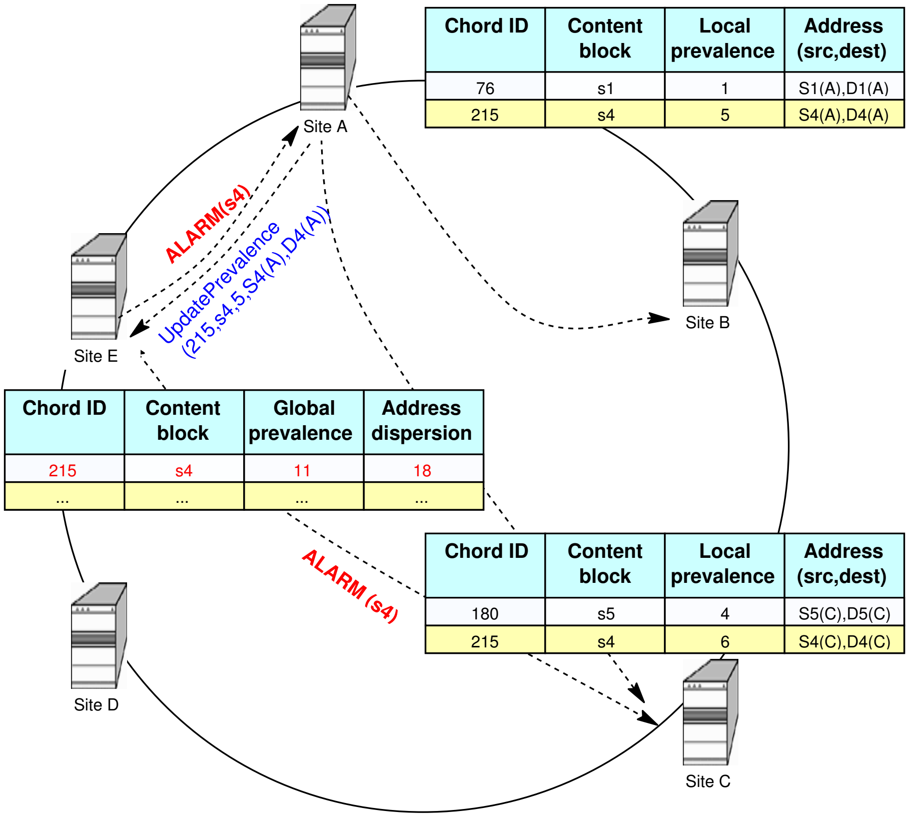
Figure 2.3: DOMINO Architecture (Adapted from [120])

2.4.3 DShield

DShield [109] is a community-based firewall log correlation system. The central server receives firewall logs from worldwide volunteers and then analyzes attack trends based on the information collected. Similar systems include myNetWatchMan [11] and CAIDA [5]. DShield is used as a data collection engine behind the SANS Internet Storm Center (ISC) [16]. Analysis provided by DShield has been used in the early detection of several worms, such as "Code Red" and "SQL Snake". Due to the number of participants and the volume of data collected, DShield is a very attractive resource, and its data is used by researchers to analyze attack patterns. However, DShield is a centralized system and it does not provide real-time analysis or rule generation. Also, due to privacy issues, payload information and some headers can not be shared, which makes classification of attacks often impossible.

2.4.4 NetShield

NetShield [35] is an IDN which uses the Chord DHT [102] to reduce communication overhead. In this system, however, within the system architecture (Figure 2.4), IDSs contribute and retrieve information from the system through a P2P overlay (the Chord DHT). Each IDS maintains a local prevalence table to record the number of occurrences of each content block signature locally as well as its corresponding source address and destination address. An update will be triggered if the local prevalence of the content block exceeds a local threshold (for example, site E in Figure 2.4). If the global prevalence is higher than a given threshold, and the address dispersion exceeds a certain threshold, then an alarm is raised regarding the corresponding content block. Netshield targets epidemic worm outbreaks or DoS attacks. However, using content blocks as attack identification is not effective against polymorphic worms (i.e., worms with changing signatures). Also, NetShield assumes all IDN participants are honest, which makes it vulnerable to collusion attacks and malicious nodes.



Chord ID	Content block	Local prevalence	Address (src,dest)
76	s1	1	S1(A),D1(A)
215	s4	5	S4(A),D4(A)

Chord ID	Content block	Global prevalence	Address dispersion
215	s4	11	18
...	...	...	...

Chord ID	Content block	Local prevalence	Address (src,dest)
180	s5	4	S5(C),D5(C)
215	s4	6	S4(C),D4(C)

Figure 2.4: NetShield Architecture (Adapted from [35])

2.4.5 CIDS

Another Collaborative Intrusion Detection System (CIDS) proposed in [127] also uses Chord DHT system to organize IDSs into a peer-to-peer network. Each IDS shares its blacklist with others through a fully distributed P2P overlay. If a suspicious IP address is reported more than a threshold N, then all the IDSs which reported it will be notified. CIDS is considered to be scalable and robust since it is built on a P2P overlay. However, the limitation of this system is that it only identifies potential intruders by IP addresses. Thus, it is not effective against worms having a spreading degree of less than N. Also, the system can be vulnerable to colluding malicious nodes.

2.4.6 Gossip

Denver et al. [39] proposed a gossip-based collaborative worm detection system (Gossip) for enterprise-level IDNs for host-based IDSs. A fully-distributed model is adopted to avoid a single point of failure. In their system, host-based IDSs (local detectors) raise alerts only if the number of newly-created connections per unit time exceeds a certain threshold. The alert will then be propagated to neighbors for aggregation. A Bayesian network-based alert aggregation model is used for alert aggregation at global detectors. Their proposed system is aimed at detecting slow-propagating worms in a local area network. However, their system only uses the new connection rate as a sign of possible worm spread. This is not effective for worms that are spread in a connectionless manner, such as UDP worms.

2.4.7 Worminator

Worminator [76] was proposed to enable IDSs to share alert information with each other to detect worm propagation. Alert correlation is used to gain better detection accuracy. Different from most other systems, Worminator is concerned with the privacy of exchanging alerts, and uses a bloom filter to encode IP addresses and port numbers in the alerts in order to preserve the privacy of collaborators. The authors claimed that the system topology can be either centralized or decentralized depending on the size of the network.

2.4.8 ABDIAS

Ghosh et. al. proposed an Agent-Based Distributed Intrusion Alert System (ABDIAS) [60]. In the architecture design (Figure 2.5), IDSs (agents) are grouped into communities (neighborhoods). Each agent collects information inside its neighborhood and uses a Bayesian network analysis model to diagnose possible threats. Inter-neighborhood communication only happens if a consensus cannot be reached within a neighborhood. This system supports early warnings for pre-attack activities in order to gain time for administrators to respond to potential attacks. This system also supports a simple majority-based voting system to detect compromised nodes.

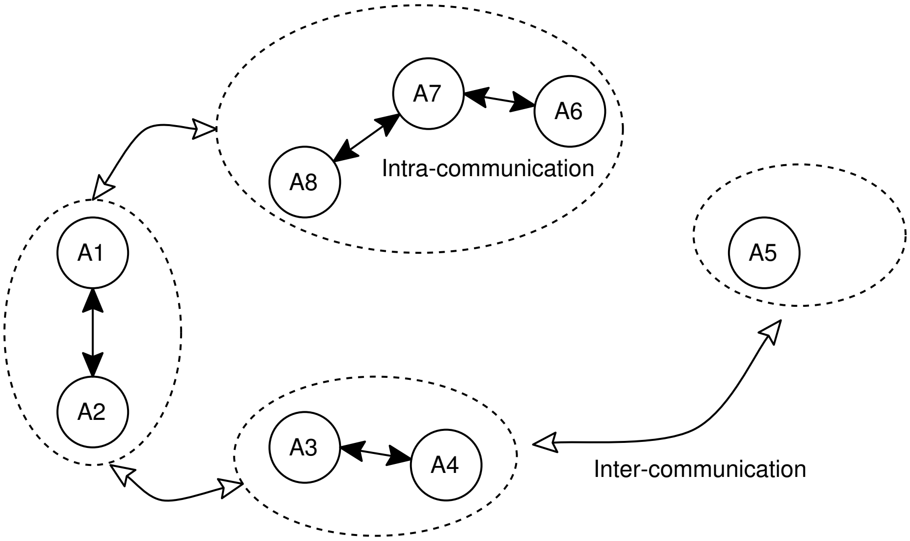


Figure 2.5: ABDIAS Architecture (Adapted from [60])

2.4.9 CRIM

CRIM [36] is a cooperative IDS where alerts from individual IDSs are sent to a central analyzer for clustering and correlation. A set of correlation rules are generated offline by security administrators by analyzing attack descriptions. These correlation rules are then used to analyze alerts collected from IDSs in order to recognize global attack scenarios.

CRIM is a semi-automatic alert correlation system, since it relies on human interactions to define attack descriptions. It is also a centralized system.

2.4.10 ALPACAS

ALPACAS[124] is a cooperative spam filtering system that is aimed at preserving the privacy of emails as well as maintaining the scalability of the system. The system is built on a peer-to-peer overlay to avoid the deficiency of a centralized system. Spam mails and Ham mails are distributed to agents based on the range of their feature signatures. An email is divided into feature trunks, and trunks are digested into feature fingerprints to preserve the content privacy of emails. The fingerprints of an email are then sent to corresponding agents to compare with stored spam emails and Ham emails by estimating the maximum signature overlap with spam (MOS) and the maximum signature overlap with Ham (MOH). An email is labeled as spam if the difference between MOS and MOH exceeds a certain threshold. ALPACAS is a fully distributed system.

2.4.11 CDDHT

The Cyber Disease Distributed Hash Table (CDDHT) [74] was proposed as a distributed data fusion center. In its architecture, each node is a local intrusion detection system which attempts to locally detect attacks and generate corresponding alerts. Each alert is assigned a disease key based on the related intrusions. The alert is then sent to a corresponding sensor fusion center (SFC) using a DHT-based P2P system. SFCs are selected among nodes based on their capacity and resources. The goal of this system is to avoid the bottleneck problem inherent to a centralized fusion center and to use alert categorization techniques for balancing the load among the SFCs. CDDHT is a decentralized system.

2.4.12 SmartScreen Filter

SmartScreen Filter [23] is a tool in Microsoft^{TM} Internet Explorer 8 that helps users avoid socially engineered malware phishing websites and online fraud when browsing the web. A centralized mechanism is used to maintain a list of phishing sites and malicious websites urls. Users browsing listed phishing sites or malicious websites will receive warnings to prevent them from being defrauded. Users are allowed to report suspicious websites to the central server through a secure channel. Users' feedback is analyzed together with input from the SmartScreen Spam filter and input from other trusted sources to generate the urls blacklist. Other similar phishing filters are provided by EarthLink and eBay.

2.4.13 FFCIDN

Fast-flux service networks (FFSN) are one type of Botnet which uses compromised nodes to form a robust phishing domain. To detect fast-flux networks and prevent them from causing further damage, Zhou et al. [126], [125] proposed a collaborative IDN to detect FFSNs. The work is based on an observation that the number of IP addresses returned after a DNS request is larger than usual. The collaboration system collects query results from nodes from different locations and correlates them to obtain the number of unique IP addresses and the number of unique fast-flux domains. The relationship between the number of DNS queries and the number of unique IP addresses and domains is traced. A corresponding DNS query threshold is derived to speed up FFSN detection. Zhou et al.'s results showed that detecting FFSNs using collaboration from nodes in different name domains is more efficient than detecting them from a single node. This system is a centralized system.

2.5 Summary

IDSs are important countermeasure for cyber attacks. However, a single IDS is vulnerable to attacks which are unknown to its security vendors of system administrators. Intrusion detection networks (IDNs) allow IDSs to exchange intrusion information and detection knowledge, hence improving the intrusion detection accuracy by using the collective knowledge from others. Several IDNs have been proposed in literature. However, most of them focused on designing efficient and scalable network overlays for the exchange of intrusion information. Some IDNs investigated information aggregation, but only few have addressed the problems of malicious insiders and free riders. Malicious insiders pose a significant challenge to IDNs since adversaries have high motivation to attack and compromise the IDSs in the network. Designing IDNs that are robust to malicious insiders is therefore of paramount importance. Free-riders also pose a significant challenge to collaboration in an IDN. They are self-interested, do not share their resources, and try to take advantage of the resources shared by others in the network. To address the free-rider problem, an incentive mechanism design should be in place to discourage selfish behaviors. My goal, in this thesis is to propose an IDN design which is not only scalable and efficient in intrusion detection, but is also robust against malicious insiders and discourage free-riding.

Chapter 3

Collaborative Intrusion Detection Networks Architecture Design

A Intrusion Detection Network (IDN) is an overlay network which enables IDSs to exchange intrusion information and knowledge in order to improve the overall detection accuracy. In chapter 2, we surveyed a number of existing intrusion detection networks where IDSs share information with others in order to detect intrusions which otherwise would not be detected by a single IDS. However, most of the previous studies focused on the efficiency of information exchange and the aggregation of collected information to make intrusion decisions. Only few studies addressed the problems of malicious insiders, free-riders, and how to select and maintain IDN participants. In this work, we propose an IDN architecture design for IDSs to communicate and enhance intrusion detection efficiency through collaboration. We specifically focus on four IDN components design, namely, trust management, feedback aggregation, resource management, and acquaintance management. Through simulation, we show that our design is scalable in network size, efficient in intrusion detection, and robust to various insider attacks.

3.1 Collaboration Framework

In our design, IDSs from different vendors or open source providers are connected in a peer-to-peer overlay. As discussed in the introduction, we choose a consultation-based

IDN design, where IDSs send consultation requests to collaborators to ask for a diagnosis when suspicious activities are detected but the host IDS does not have enough confidence to make a correct decision. For this purpose, each IDS maintains a list of "good" collaborators. For example, IDSs may choose to collaborate with other IDSs with which they had good experience in the past (e.g., have been helpful in identifying intrusions). The reason that we choose a peer-to-peer network structure instead of a cloud-based service such as CloudAV [89] is that cloud-based services are centralized and may constitute a performance bottleneck and a single point of failure. In turn, in a peer-to-peer IDN, workload is distributed to all peers and there is no communication bottleneck. Also, IDNs are more robust to failures and can scale indefinitely in network size. In our design, we consider the case where the IDN participants have differing detection expertise levels and may act dishonestly or selfishly in collaboration. For collaboration to be sustainable and efficient, we identify the following IDN design requirements:

(1) IDN nodes should have an effective trust evaluation capability to reduce the negative impact of dishonest and incompetent nodes.

(2) Allocation of IDN node resources for collaboration should be incentive-compatible to discourage selfish behavior and encourage active collaboration.

(3) IDN nodes should possess an efficient feedback aggregation capability to minimize the cost of false intrusion detections.

(4) The IDN should be robust against insider attacks.

(5) The IDN should be scalable in network size.

To satisfy the above requirements, we propose a Collaborative Intrusion Detection Network (IDN) architecture design similar to a social network. The IDN topology, as shown in Figure 3.1, consists of IDSs (nodes), which may be Network-based IDSs (NIDSs) or Host-based IDSs (HIDSs). IDN nodes are connected if they have a collaborative relationship. Each node maintains a list of other nodes which it currently collaborates with. We call such nodes *acquaintances*. Each node in the IDN has the freedom to choose its acquaintances based on their trustworthiness. The communication between collaborating nodes consists of intrusion evaluation requests and corresponding feedback. There are two types of requests: *intrusion consultations* and *test messages*. The architecture of the IDN is

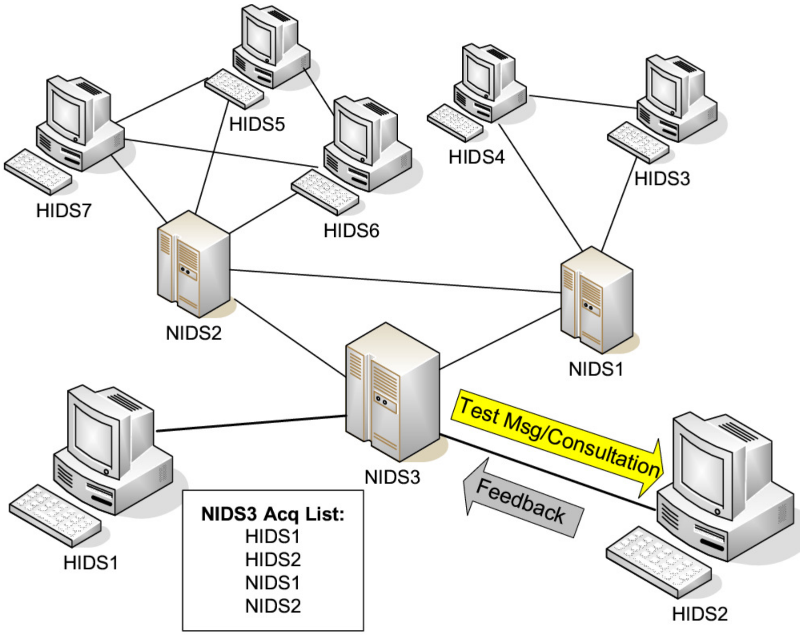


Figure 3.1: Topology of a Collaborative Intrusion Detection Network.

shown in Figure 3.2. It is composed of seven components, namely, the intrusion detection system, communication overlay, trust management, acquaintance management, resource management, feedback aggregation, and mediator. In the following subsections, we will describe the consultation and test messages and the functionality of each component in the architecture.

3.1.1 Consultation Message

When an IDS detects a suspicious activity but is unable to make a decision as to whether it should raise an alarm or not (e.g., anomaly detection detects suspicious activities but no matching signature is found), it sends consultation requests to its acquaintances for diagnosis. Feedback from acquaintances is aggregated and a final intrusion detection decision is made based on the aggregated results. The amount of information in the consultation request depends on the trust level of each acquaintance. For example, a node may want to share all alert information, including data payload, with the nodes inside its local area network, and digest or even remove some alert information when sent to acquaintances in

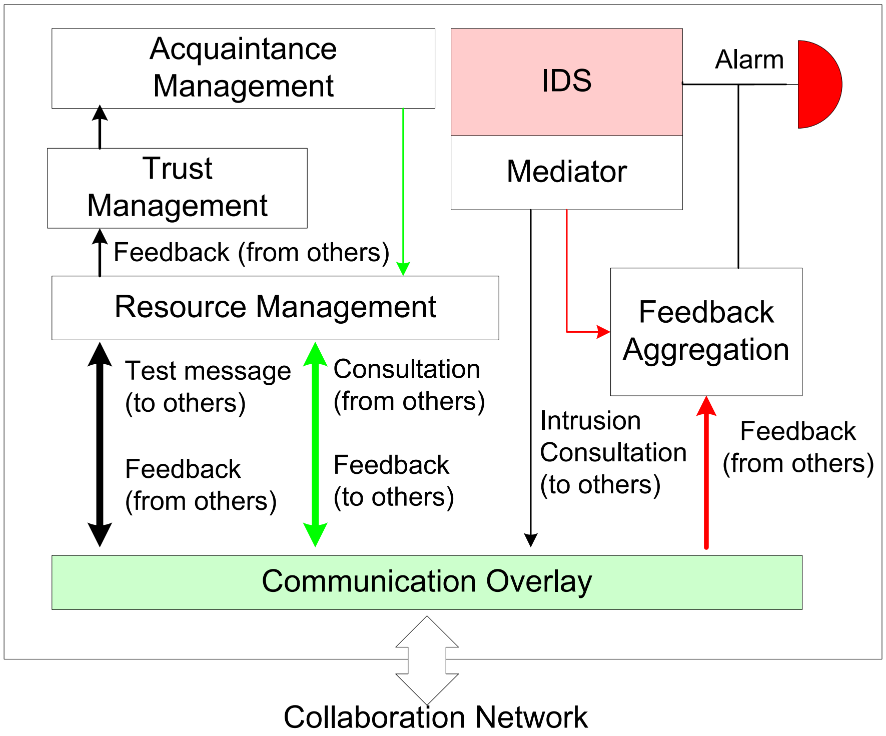


Figure 3.2: Architecture Design of an IDN.

the broader Internet.

3.1.2 Test Messages

In order for the nodes in the IDN to gain experience with each other, we propose that IDSs use test messages to evaluate the trustworthiness of others. Test messages are "bogus" consultation requests which are sent to measure the trustworthiness of another node in the acquaintance list. They are sent out in a way that makes them difficult to be distinguished from a real consultation request. The testing node knows the true diagnosis result of the test message and uses the received feedback to derive a trust value for the tested node. This technique can discover inexperienced and/or malicious nodes within the collaborative network. A test message can be a previous consultation message with which the ground truth has been verified, or a random pick taken from its knowledge base.

3.1.3 Communication Overlay

The Communication Overlay is the component which handles all the communications with other peers in the collaborative network. The messages passing through the communication overlay include: test messages from the host node to its acquaintances; intrusion consultations from the host node to its acquaintances; feedback from acquaintances; consultation requests from acquaintances, and; feedback to acquaintances. The Communication Overlay dispatches incoming requests and messages to corresponding components in the system and routes outgoing requests and messages to their destinations. For example, when the Communication Overlay component receives a consultation request, it calls the local IDS component for diagnosis and returns the received feedback (diagnosis result) back to the sender.

3.1.4 Mediator

The mediator is the component which helps heterogeneous IDSs communicate with each other. It translates consultation requests and consultation feedback into a common protocol (such as IDMEF [10]) and data format understood by different IDSs.

3.1.5 Trust Management

The trust management component allows IDSs in the IDN to evaluate the trustworthiness of others based on previous experience with them. The host node can use test messages to gain experience quickly. Indeed, the verified consultation results can also be used as experience. In our proposed IDN, we have adopted a Dirichlet-based trust management model [57, 53] to evaluate the trustworthiness of IDSs. In this trust model, IDSs evaluate the trustworthiness of others based on the quality of their feedback. The confidence of trust estimation is modeled using Bayesian statistics, and the results show that the frequency of test messages is proportional to the confidence level of trust estimation. The trust management model is closely connected to the resource management and acquaintance management models, since the trust values of the collaborators are essential inputs for the latter models.

3.1.6 Acquaintance Management

It is intuitive that when an IDS consults more acquaintances, it achieves higher accuracy and confidence in intrusion detection. However, more acquaintances results in a higher maintenance cost, since the IDS needs to allocate resources for each acquaintance because sending and receiving test messages to those acquaintances is a necessary resource expenditure, and it is needed to maintain the confidence of trust evaluation and to maintain the collaboration connection. In addition to the acquaintances list, our system also maintains a consultation list. The nodes on the consultation list are randomly selected from the acquaintances which have passed the probation period. Test messages are sent to all acquaintances, while consultation requests are only sent to the nodes in the consultation list. The acquaintance list is updated on a regular basis to recruit new nodes or remove unwanted ones. In our system, we use a dynamic acquaintance management system [54, 55] to recruit higher-quality peers and remove less helpful peers based on their trustworthiness and expertise in intrusion detection.

3.1.7 Resource Management

In an IDN, malicious or compromised peers can launch a Denial-of-Service attack by sending a large number of consultation messages to overwhelm the targeted IDSs. Some peers may also free-ride the system by only receiving help from others without contributing to the collaboration network. To address the above problems, a resource management system is required to decide whether the host should allocate resources to respond to a given consultation request. An incentive-compatible resource management can assist IDSs to allocate resources to their acquaintances so that other IDSs are fairly treated based on their past assistance to the host IDS. Therefore, an IDS which abusively uses the collaboration resource will be penalized by receiving fewer responses from others. The resource allocation system also decides how often the host should send test messages to its acquaintances, protecting the system from being overloaded. In our IDN, we use an incentive-compatible resource allocation system [132, 133] based on a multi-player non-cooperative game design for IDSs in the IDN.

3.1.8 Feedback Aggregation

When the IDS of the host node cannot make a confident intrusion diagnosis for a suspicious event, the host node may consult the other IDSs in the collaboration network for opinions/diagnosis. The received feedback is then used to make a decision as to whether the host IDS should raise an alarm to its administrator or not. The feedback aggregation component is responsible for making a decision based on the feedback. It decides not only on which criteria to use to measure the quality of decisions, but also on how to reach a decision in an efficient way. This component is one of the most important, since it has a direct impact on the accuracy of the collaborative intrusion detection. If an alarm is raised, the suspicious intrusion flow will be suspended and the system administrator must investigate the intrusion immediately. On one hand, false alarms may waste human resources. On the other hand, undetected intrusions may cause damage. In this thesis, we use a Bayesian approach [58, 129] to measure the rate of false alarms, i.e., false positive (FP) rate, and the rate of missing intrusions, i.e., false negative (FN) rate, of participating IDSs based on collected experience with them in the past. We model the cost of collaborative decision-making using false positive cost and false negative cost. We then provide a hypothesis testing model to find a decision which leads to minimum overall cost.

In the following chapters, we focus on four major components of the propose IDN architecture: trust management, acquaintance management, resource management, and feedback aggregation. For each component, we provide the underlying model and algorithms, and evaluate their efficiency against several metrics, including robustness, scalability, efficiency, fairness, and incentive-compatibility.

Chapter 4

Trust Management

4.1 Introduction

Collaborative intrusion detection networks can be an effective way to improve intrusion detection accuracy and detect new attacks. However, a malicious (or malfunctioning) IDS can degrade the performance of others by sending false intrusion assessments. Furthermore, if some nodes are controlled by the same adversaries, they can easily collude and send false intrusion assessments. Moreover IDSs may have different levels of expertise in intrusion assessment. To protect an IDN from malicious attacks as well as find expert IDSs to consult for intrusion assessment, it is important to evaluate the trustworthiness of participating IDSs. Since the trust model itself may also be the target of malicious attacks, robustness is a desired feature of the trust management scheme in collaborative intrusion detection networks.

In this chapter, we present a Bayesian trust management model that is robust, scalable, and suitable for distributed IDS collaboration. More specifically, a Dirichlet trust model is used for estimating the likely future behavior of an IDS based on its past history. This theoretical model allows us to track the uncertainty in estimating the trustworthiness of the IDS. The use of sophisticated trust model in IDN not only effectively improves the intrusion detection efficiency of the IDN, it also improves the robustness of the system against various insider attacks. Our proposals can be used to deploy a secure and scalable IDN where effective collaboration can be established between IDSs.

We evaluate our approach based on a simulated collaborative IDS network. The IDSs are distributed and may have different expertise levels in detecting intrusions. An IDS may also turn malicious due to runtime bugs, having been compromised, having been updated with a faulty new configuration, or having been deliberately made malicious. We also simulate several potential threats; e.g., betrayal attacks where malicious IDSs masquerade as honest ones to gain trust, and then suddenly act dishonestly. Our experimental results demonstrate that our trust management model yields a significant improvement in detecting intrusions, is robust against various attacks, and improves the scalability of the system, as compared to existing collaborative IDS systems.

The remainder of this chapter is organized as follows. The trust management model is presented in Section 4.2. The scalability of the trust model is discussed in Section 4.3 and its robustness against common threats in Section 4.4. Section 4.5 provides experimental evidence of the efficiency, robustness and scalability of our approach. Section 4.6 surveys related work. Section 4.7 summarizes our contributions and proposes future work.

4.2 Trust Management Model

In this section, we propose a robust and scalable trust model which uses a Bayesian approach to evaluate the trustworthiness between each pair of IDSs. Specifically, we use a Dirichlet family of probability density functions to estimate the likely future behavior of an IDS based on its past history. A weighted majority method is used to aggregate feedback to make intrusion decisions.

4.2.1 Satisfaction Mapping

In our model, an IDS sends requests to its peers and evaluates the satisfaction level of received feedback. Note that the request can be a test message or a real request. The true answer of a test message is known beforehand, while that of a real request is verified by administrators after some delay through the observed impact of the corresponding alert.

IDSs may have different metrics to rank alerts. Snort [17], for example, uses three levels (low, medium, high), while Bro [4] allows up to 100 different levels. We assume the existence of a function H, which maps an IDS alert ranking onto the [0,1] interval where

0 denotes benign traffic and 1 highly dangerous intrusions. H preserves the "more severe than" partial order relationship. That is, if alert a_j is more severe than alert a_i then H preserves that relationship by having $H(a_j) > H(a_i)$.

The satisfaction level of feedback is determined by three factors: the expected answer ($r \in [0,1]$), the received answer ($a \in [0,1]$) and the difficulty level of the test message ($d \in [0,1]$). The larger is d the more difficult it is to correctly answer the request. Note that the difficulty of the test message can be roughly estimated by the age of the corresponding signatures or knowledge. For example, the difficulty level is low for test messages generated from old signatures; medium difficulty is for test messages generated from new signatures; high difficulty for malicious traffic taken from Honeypots and no local signature is able to detect it.

To quantitatively measure the quality of feedback, we use a function $Sat(r, a, d)$ ($\in [0,1]$) to represent the level of satisfaction of the received answer based on its distance to the expected answer and the difficulty of the test message, as follows:

$$Sat(r,a,d) = \begin{cases} 1 - \left(\frac{a-r}{\max(c_1 r, 1-r)}\right)^{d/c_2} & a > r \\ \\ 1 - \left(\frac{c_1(r-a)}{\max(c_1 r, 1-r)}\right)^{d/c_2} & a \leq r \end{cases} \tag{4.1}$$

where $c_1 \in R^+$ controls the extent of penalty for wrong estimates. $c_1 > 1$ reflects the situation that estimates lower than the exact answer get stronger penalty than those that are higher. Parameter $c_2 \in R^+$ controls satisfaction sensitivity, with larger values reflecting more sensitivity to the distance between the correct and received answers. The equation also ensures that low difficulty level tests are more severe in their penalty to incorrect answers. The shape of the satisfaction function is depicted in Figure 4.1.

4.2.2 Dirichlet-based Model

Bayesian statistics provides a theoretical foundation for measuring the uncertainty in a decision that is based on a collection of observations. We are interested in knowing the distribution of satisfaction levels of the answers from each peer IDS and, particularly, using this information to estimate the satisfaction level of future consultations. For the case of a

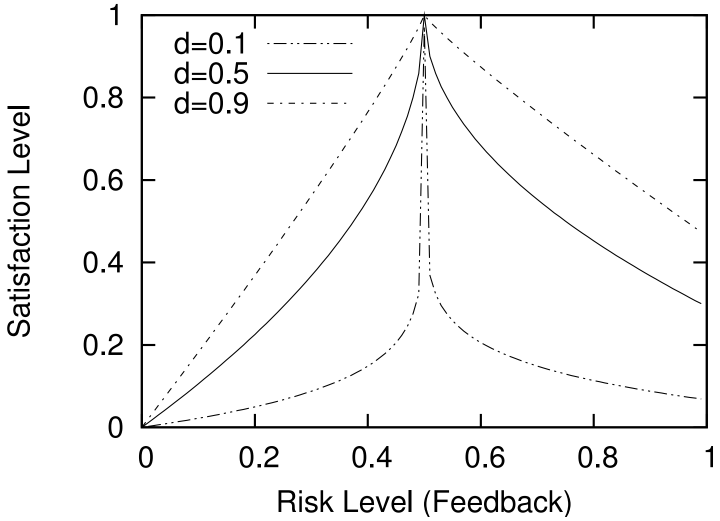


Figure 4.1: Satisfaction level for feedback (r=0.5, $c_1 = 2$, $c_2 = 1$)

binary satisfaction level {satisfied,¬satisfied}, a Beta distribution can be used as appeared in [122]. For multi-valued satisfaction levels, Dirichlet distributions are more appropriate.

A Dirichlet distribution [95] is based on initial beliefs about an unknown event represented by a prior distribution. The initial beliefs combined with collected sample data can be represented by a posterior distribution. The posterior distribution well suits our trust management model since the trust is updated based on the history of interactions.

Let X be the discrete random variable denoting the satisfaction level of the feedback from a peer IDS. X takes values in the set $\mathcal{X} = \{x_1, x_2, ..., x_k\}$ ($x_i \in [0, 1]$, $x_{i+1} > x_i$) of the supported levels of satisfaction. Let $\vec{p} = \{p_1, p_2, ..., p_k\}$ ($\sum_{i=1}^{k} p_i = 1$) be the probability distribution vector of X, i.e. $P\{X = x_i\} = p_i$. Also, let $\vec{\gamma} = \{\gamma_1, \gamma_2, ..., \gamma_k\}$ denote the vector of cumulative observations and initial beliefs of X. Then we can model $\vec{p}$ using a posterior Dirichlet distribution as follows:

$$f(\vec{p}|\xi) = Dir(\vec{p}|\vec{\gamma}) = \frac{\Gamma(\sum_{i=1}^{k} \gamma_i)}{\prod_{i=1}^{k} \Gamma(\gamma_i)} \prod_{i=1}^{k} {p_i}^{\gamma_i - 1} \tag{4.2}$$

where ξ denotes the background knowledge, which in here is summarized by $\vec{\gamma}$.

Let

$$\gamma_0 = \sum_{i=1}^{k} \gamma_i \tag{4.3}$$

The expected value of the probability of X to be x_i given the history of observations $\vec{\gamma}$ is given by:

$$E(p_i|\vec{\gamma}) = \frac{\gamma_i}{\gamma_0} \tag{4.4}$$

In order to give more weight to recent observations over old ones, we embed a *forgetting factor* λ in the Dirichlet background knowledge vector $\vec{\gamma}$ as follows:

$$\vec{\gamma}^{(n)} = \sum_{i=1}^{n} \lambda^{t_i} \times \vec{S^i} + c_0 \lambda^{t_0} \vec{S^0} \tag{4.5}$$

where n is the number of observations; $\vec{S^0}$ is the initial beliefs vector. If no additional information is available, all outcomes have an equal probability making $S^0_j = 1/k$ for all $j \in \{1, .., k\}$. Parameter $c_0 > 0$ is a priori constant, which puts a weight on the initial beliefs. Vector $\vec{S^i}$ denotes the satisfaction level of the i^{th} evidence, which is a tuple containing $k-1$ elements set to zero and only one element set to 1, corresponding to the selected satisfaction level for that evidence. Parameter $\lambda \in [0, 1]$ is the forgetting factor. A small λ makes old observations quickly forgettable. Parameter t_i denotes the time elapsed (age) since the i^{th} evidence $\vec{S^i}$ was observed. Let $\Delta t_i = t_i - t_{i+1}$. For the purpose of scalability, the $\vec{\gamma}^{(n)}$ in Equation 4.5 can be rewritten in terms of $\vec{\gamma}^{(n-1)}$, $\vec{S^n}$ and Δt_n as follows:

$$\vec{\gamma}^{(n)} = \begin{cases} c_0 \vec{S^0} & n = 0 \\ \lambda^{\Delta t_n} \times \vec{\gamma}^{(n-1)} + \vec{S^n} & n > 0 \end{cases} \tag{4.6}$$

4.2.3 Evaluating the Trustworthiness of a Peer

After a peer receives the feedback for an alert evaluation, it assigns a satisfaction value to the feedback according to Equation 4.1. This satisfaction value is assigned with one of the satisfaction levels in the set $\mathcal{X} = \{x_1, x_2, ..., x_k\}$ that has the closest value. Each satisfaction level x_i also has a weight w_i.

Let p_i^{uv} denote the probability that peer v provides answers to the requests sent by peer u with satisfaction level x_i. Let $\vec{p}^{uv} = (p_i^{uv})_{i=1...k} \mid \sum_{i=1}^{k} p_i^{uv} = 1$. We model $\vec{p}^{uv}$

using Equation 4.2. Let Y^{uv} be the random variable denoting the weighted average of the probability of each satisfaction level in $\vec{p}^{uv}$.

$$Y^{uv} = \sum_{i=1}^{k} p_i^{uv} w_i \tag{4.7}$$

The *trustworthiness* of peer v as noticed by peer u is then calculated as:

$$T^{uv} = E[Y^{uv}] = \sum_{i=1}^{k} w_i E[p_i^{uv}] = \frac{1}{\gamma_0^{uv}} \sum_{i=1}^{k} w_i \gamma_i^{uv} \tag{4.8}$$

where γ_i^{uv} is the cumulated evidence that v has replied to u with satisfaction level x_i. The variance of Y^{uv} is equal to (superscript uv is omitted for clarity):

$$\sigma^2[Y] = \sum_{i=1}^{k} \sum_{j=1}^{k} w_i w_j cov[p_i, p_j] \tag{4.9}$$

Knowing that the covariance of p_i and p_j (i $\neq$ j) is given by:

$$cov(p_i, p_j) = \frac{-\gamma_i \gamma_j}{\gamma_0^2(\gamma_0 + 1)} \tag{4.10}$$

We get:

$$\begin{aligned}
\sigma^2[Y] &= \sum_{i=1}^{k} w_i^2 \sigma^2[p_i] + 2 \sum_{i=1}^{k} \sum_{j=i+1}^{k} w_i w_j cov[p_i, p_j] \\
&= \sum_{i=1}^{k} w_i^2 \frac{\gamma_i(\gamma_0 - \gamma_i)}{\gamma_0^2(\gamma_0 + 1)} + 2 \sum_{i=1}^{k} \sum_{j=i+1}^{k} w_i w_j \frac{-\gamma_i \gamma_j}{\gamma_0^2(\gamma_0 + 1)} \\
&= \frac{1}{\gamma_0^3 + \gamma_0^2} \sum_{i=1}^{k} w_i \gamma_i \left(w_i(\gamma_0 - \gamma_i) - 2 \sum_{j=i+1}^{k} w_j \gamma_j \right)
\end{aligned} \tag{4.11}$$

Let $C^{uv} \in (-1, 1]$ be the confidence level for the value of T^{uv}, and we describe it as:

$$C^{uv} = 1 - 4\,\sigma[Y^{uv}] \tag{4.12}$$

where $4\,\sigma[Y^{uv}]$ is roughly the 95% confidence interval.

Lemma 4.2.1 *The confidence level C^{uv} formulated by Equation 4.12 lies in bound (-1, 1].*

Proof From Equation 4.12 and Equation 4.11, we have,

$$C^{uv} = 1 - \frac{4}{\sqrt{1+\gamma_0}}\sqrt{\sum_{i=1}^{k} w_i^2 \frac{\gamma_i}{\gamma_0} - (\sum_{i=1}^{k} w_i \frac{\gamma_i}{\gamma_0})^2} \tag{4.13}$$

where $w_i \in [0, 1], \forall i$ is the weight of the satisfaction level i, and $\gamma_0 = \sum_{i=1}^{k} \gamma_i > 0$. To prove the boundary of C^{uv}, we construct a discrete random variable $Z \in \{w_1, w_2, ..., w_k\}$, where $w_1 \le w_2 \le ... \le w_k$ and $\mathbb{P}[Z = w_i] = \frac{\gamma_i}{\gamma_0}, \forall i$. Then we have,

$$\sigma^2[Z] = \mathbb{E}(Z^2) - \mathbb{E}^2(Z) = \sum_{i=1}^{k} w_i^2 \frac{\gamma_i}{\gamma_0} - (\sum_{i=1}^{k} w_i \frac{\gamma_i}{\gamma_0})^2 \tag{4.14}$$

We can see that the variation of Z is the major component of C^{uv}. It is not difficult to see that $\sigma^2[Z]$ reaches its maximum when $\mathbb{P}[Z = w_1] = \mathbb{P}[Z = w_k] = 0.5$ and $\mathbb{P}[Z = w_j] = 0, \forall j (1 < j < k)$. Therefore, we have $0 \le \sigma^2[Z] \le \frac{1}{4}$. After replacing Equation 4.14 back into Equation 4.13, we have $-1 < C^{uv} \le 1$.

4.3 Test Message Exchange Rate and Scalability of Our System

Each IDS u in our system maintains an acquaintance list and a probation list with maximum length l^u_{max}. This length can be fixed according to the resource capacity of node u or slightly updated with the changes in IDN size. However, it is always set to a value small

Table 4.1: Acquaintance Categorization

Peer category	Criterion	Rate
Highly Trustworthy	$0 < th \leq T_l$	R_l
Trustworthy	$T_l < th \leq T$	R_h
Untrustworthy	$T < th \leq T_h$	R_m
Highly Untrustworthy	$T_h < th \leq 1$	R_l

enough to account for scalability. Equation 4.6 ensures that the process of updating the trustworthiness of a peer after the reception of a response is performed with only three operations, making it linear with respect to the number of answers.

There is a trade-off to be resolved in order to account for scalability in the number of messages exchanged in the IDN. On one hand, the forgetting factor in Equation 4.6 decays the importance given to existing highly trusted peers. This implies that their corresponding test message rates need to be above a certain minimal value. On the other hand, sending too many requests to other peers may compromise scalability. To solve this issue, we adapt the rate of test messages to a given peer according to its estimated trustworthiness. The adaptation policy is provided in Table 4.1, where acquaintances are categorized into highly trustworthy, trustworthy, untrustworthy, and highly untrustworthy. There are three levels of test message rates: $R_l < R_m < R_h$. We can see in Table 4.1 that the test message rate to highly trustworthy or highly untrustworthy peers is low. This is because we are confident about our decision of including or not their feedback into the aggregation. A higher test message rate is assigned to trustworthy or untrustworthy peers because their trust values are close to the threshold and hence need to be kept under close surveillance.

Each peer in the system needs to actively respond to others' requests in order to keep up its trustworthiness and be able to receive prompt help when needed. However, actively responding to every other peer may cause bandwidth and/or CPU overloading. Therefore, as a consultant to others, a peer would like to limit the rate of answers it provides. In this regard, each peer in our system would respond to requests with a priority proportional to the amount of trust it places on the source of the request [133]. It will give higher priority to highly trusted friends. This obeys the social norm: "Be nice to others who are nice to you", and also provides incentives for encouraging peers to act honestly in order to receive prompt help in times of need.

4.4 Robustness against Common Threats

Trust management can effectively improve network collaboration and detect malicious peers. However, the trust management system itself may become the target of attacks and be compromised. In this section, we describe common attacks and provide defense mechanisms against them.

Sybil attacks

Sybil attacks occur when a malicious peer in the system creates a large amount of pseudonyms (fake identities) [44]. Such a malicious peer uses fake identities to gain larger influence in the network and use it in false ranking of alerts. Our defense against sybil attacks rely on the authentication mechanism in place (e.g., a certificate issuing authority) and our acquaintance management system. Authentication makes registering fake identities difficult. The registration of new user IDs requires puzzle solving (such as CAPTCHA) which requires human intelligence to handle. In this way, creating a large number of fake IDs is not practical for an attacker. In addition, our trust management model requires IDSs to first build up their trust before they can affect the decision of others, which is costly to do with many fake identities. This way, our security and trust mechanisms protect our collaborative network from sybil attacks.

Newcomer attacks

Newcomer attacks occur when a malicious peer can easily register as a new user [92]. Such a malicious peer creates a new ID for the purpose of erasing its bad history with other peers in the network and create immediate damages. Our model handles this type of attacks by assigning low trust values to all newcomers and enforcing the probation period for each new node. In this way, their feedback on the alerts is simply not considered by other peers during the aggregation process. Newcomers may gain more trust over time and eventually move to acquaintance list if they behave consistently well.

Betrayal attacks

Betrayal attacks occur when a trusted peer suddenly turns into a malicious one and starts sending false feedbacks. A trust management system can be degraded dramatically because of this type of attacks. We employ a mechanism which is inspired by the social norm: "It takes a long-time interaction and consistent good behavior to build up a high trust, while only a few bad actions to ruin it." When a trustworthy peer acts dishonestly, the forgetting factor (Equation 4.6) causes its trust value to drop down quickly, hence making it difficult for this peer to deceive others or gain back its previous trust within a short time.

Collusion attacks

Collusion attacks happen when a group of malicious peers cooperate together by providing false alert rankings in order to compromise the network. In our system, peers will not be adversely affected by collusion attacks. In our trust model each peer relies on its own knowledge to detect dishonest peers. In addition, we use test messages to uncover malicious peers. Since the test messages are sent in a random manner, it will be difficult for malicious peers to distinguish them from actual requests.

Inconsistency attacks

Inconsistency attacks happen when a malicious peer repeatedly changes its behavior from honest to dishonest in order to degrade the efficiency of the IDN. Inconsistency attacks are harder to succeed in the Dirichlet-based model because of the use of the forgetting factor and the dynamic test message rate, which makes trust values easy to lose and hard to gain. This ensures that the trust values of peers with inconsistent behavior remain low and hence have little impact.

4.5 Simulations and Experimental Results

In this section, we present a set of experiments used to evaluate the efficiency, scalability and robustness of our trust management model in comparison with existing ones [56][45].

Table 4.2: Simulation Parameters

Parameter	Value	Description
R_l	2/day	Low test message rate
R_m	10/day	Medium test message rate
R_h	20/day	High test message rate
λ	0.9	Forgetting factor
th	0.8	Trust threshold for aggregation
c_0	10	Priori Constant
c_1	1.5	Cost rate of low estimate to high estimate
c_2	1	Satisfaction sensitivity factor
s	4	Size of grid region
k	10	Number of satisfaction levels

Experiments are also carried out to demonstrate the desirable properties of our acquaintance management algorithm. Each experimental result presented in this section is derived from the average of a large number of replications with an overall negligible confidence interval.

4.5.1 Simulation Setting

We simulate an IDN environment with n IDS peers randomly distributed over an $s \times s$ grid region. The proximity distance is given by the minimum number of square steps between each two peers. The expertise level of a peer can be low (0.05), medium (0.5) or high (0.95). In the beginning, each peer receives an initial acquaintance list containing neighbour nodes based on proximity. The initial trust value of every peer in the acquaintance list is 0.5. To test the trustworthiness of acquaintances, each peer sends out test messages following a Poisson process with rates according to Table 4.1. The parameters we used are shown in Table 4.2.

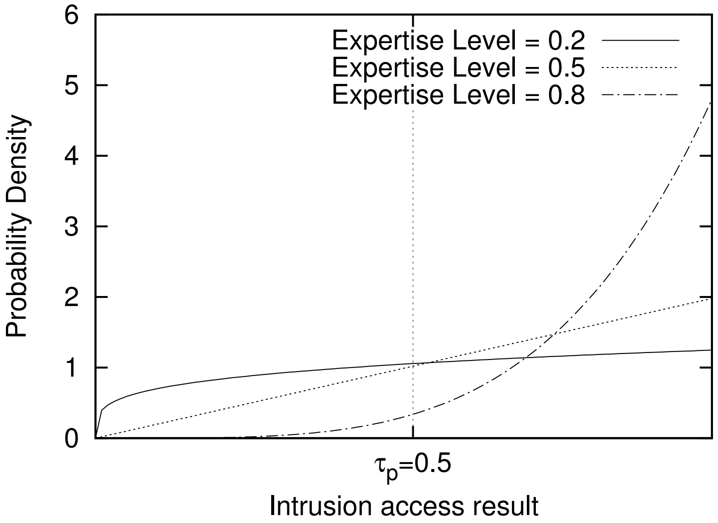


Figure 4.2: Decision Density Function for Expertise Levels

4.5.2 Modeling the Expertise Level of a Peer

To reflect the expertise level of each peer, we use a Beta distribution to simulate the decision model of answering requests. A Beta density function is given by:

$$\begin{aligned} f(p|\alpha,\beta) &= \frac{1}{B(\alpha,\beta)}p^{\alpha-1}(1-p)^{\beta-1} \\ B(\alpha,\beta) &= \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt \end{aligned} \tag{4.15}$$

where $f(p|\alpha,\beta)$ is the probability that a peer with expertise level l answers with a value of $p \in [0,1]$ to an alert of difficulty level $d \in [0,1]$. Higher values for d are associated to attacks that are difficult to detect, i.e. many peers fail to identify them. Higher values of l imply a higher probability of producing correct alert rankings.

Let r be the expected ranking of an alert. We define α and β as follows:

$$
\begin{aligned}
\alpha &= 1 + \frac{l(1-d)}{d(1-l)}\sqrt{\frac{r}{1-r}}\sqrt{\frac{2}{l}-1} \\
\beta &= 1 + \frac{l(1-d)}{d(1-l)}\sqrt{\frac{1-r}{r}}\sqrt{\frac{2}{l}-1} \qquad (4.16)
\end{aligned}
$$

For a fixed difficulty level, the above model has the property of assigning higher probabilities of producing correct rankings to peers with higher levels of expertise. A peer with expertise level l has a lower probability of producing correct rankings for alerts of higher difficulty ($d > l$). $l = 1$ or $d = 0$ represent the extreme cases where the peer can always accurately rank the alert. This is reflected in the Beta distribution by $\alpha, \beta \rightarrow \infty$. Figure 4.2 shows the feedback probability distribution for peers with different expertise levels, where we fix the expected risk level to 0.6 and the difficulty level of test messages to 0.5.

4.5.3 Deception Models

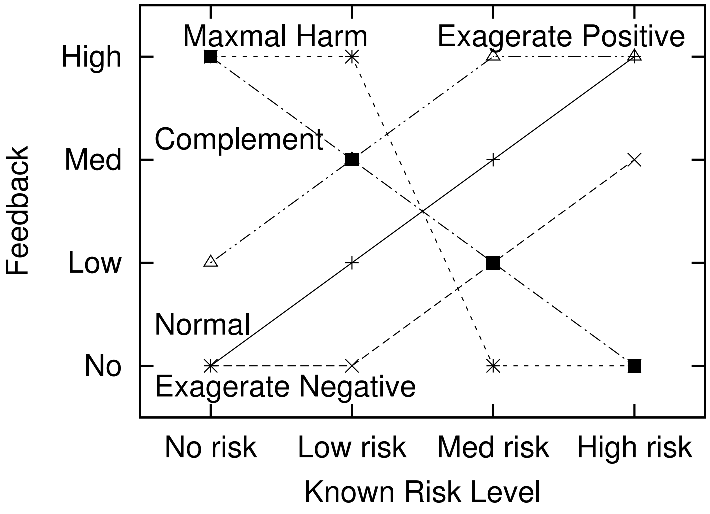


Figure 4.3: Feedback Curves for Different Deception Strategies

A dishonest peer may adopt one of the four deception models: *complementary*, *exaggerate positive*, *exaggerate negative*, and *maximal harm*. The first three deception models are described in [121], where an adversary may choose to send feedback about the risk level of an alert that is respectively opposite to, higher, or lower than the true risk level. We propose a maximal harm model where an adversary always chooses to report false feedback with the intention to bring the most negative impact to the request sender. Figure 4.3 shows the feedback curve for the different deception strategies. For instance, when a deceptive peer using the maximal harm strategy receives a ranking request and detects that the risk level of the request is "medium", it sends feedback "no risk" because this feedback can maximally deviate the aggregated result at the sender side.

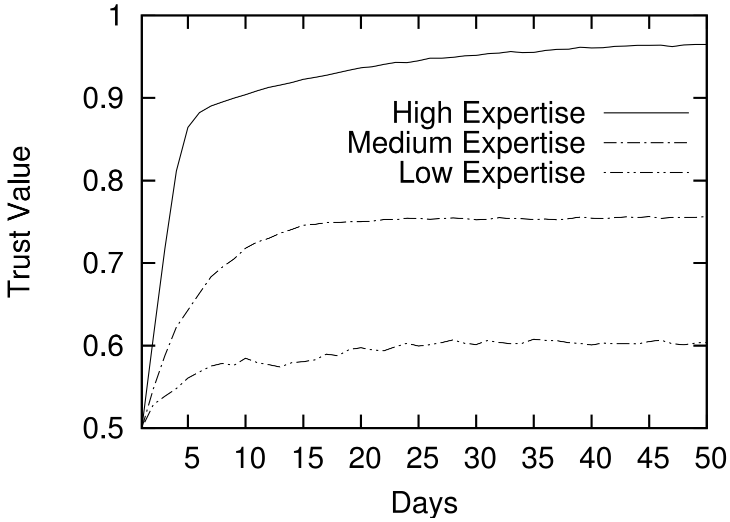


Figure 4.4: Convergence of Trust Values for Different Expertise Levels

4.5.4 Trust Values and Confidence Levels for Honest Peers

We first evaluate the effectiveness of the collaboration and the importance of our trust management. In this experiment, all peers are honest. We simulate the scenario where each peer u has a fixed size N^u of its acquaintance list. The peers are divided into three equally-sized groups of *low*, *medium* and *high* expertise levels respectively. The first phase of the simulation is a learning period (50 days), during which peers learn about each other's expertise levels by sending out test messages. Figure 4.4 shows the resulting average trust values of the 30 acquaintances of peer u. The trust values converge after 30 days of

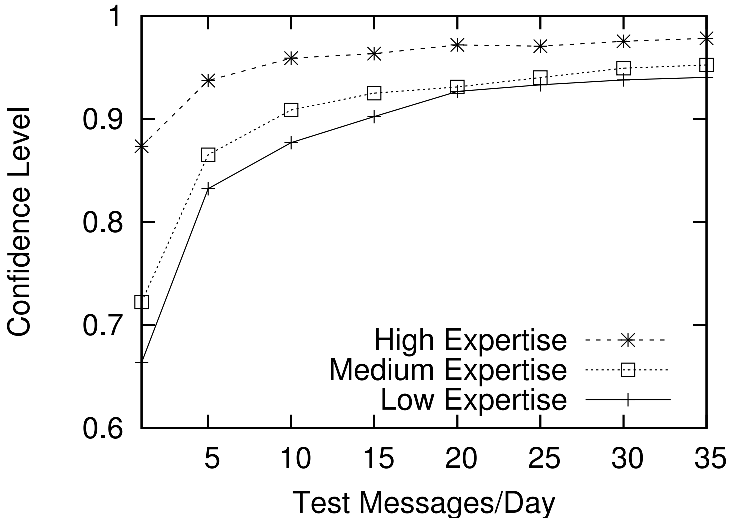


Figure 4.5: Confidence Levels of Estimation for Different Test Message Rates

simulation and the actual expertise levels of the peers are able to be effectively identified by our trust model.

To study the impact of different test message rates on the confidence level of trust estimation (Equation 4.12), we conduct a second experiment to let u use a fixed test message rate in every simulation round. The rate of sending test messages starts with one message per day and increases by five for every simulation round. We plot the confidence level of trust evaluation for each test message rate in Figure 4.5. We can observe that the confidence level increases with the increase of the test message rate. This confirms our argument that sending more test messages improves the confidence of trust estimation. We also observe that the confidence levels increase with the expertise levels. This is because peers with higher expertise levels tend to perform more consistently.

4.5.5 Trust Values for Dishonest Peers

The purpose of this experiment is to study the impact of dishonest peers using the four different deception strategies described in Section 4.5.3. To study the maximum impact of these deception strategies, we only use peers with a *high* expertise level as deceptive

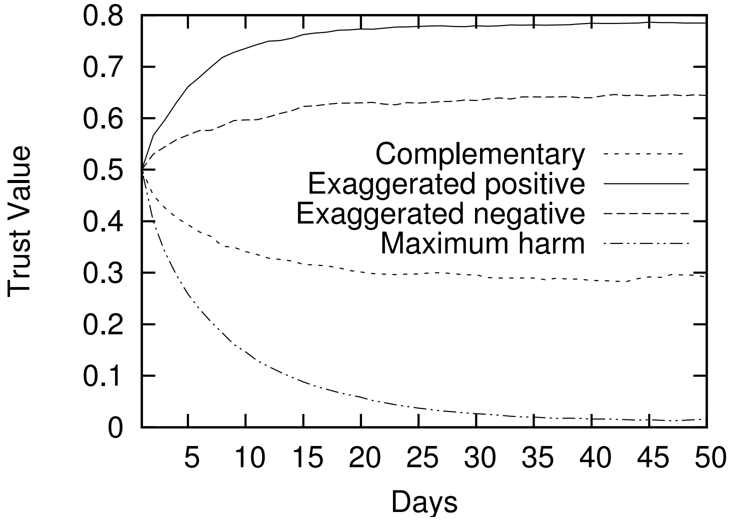


Figure 4.6: Trust Values of Deceptive Peers with Different Deception Strategies

adversaries since they are more likely to know the true answers and can perform the deception strategies more accurately.

In this experiment, we let peer u have an acquaintance list of 40 dishonest peers divided into four groups. Each group uses one of the four deception models: complimentary, exaggerate positive, exaggerate negative, and maximal harm. We use a dynamic test message rate and observe the convergence curve of the average trust value for each group of deceptive peers. Results are plotted in Figure 4.6.

We notice that the trust values of all adversary peers converge to stable values after 30 days of the learning phase. It is not surprising that adversary peers using the maximal harm strategy have the lowest trust values, while adversary peers using the complimentary strategy have the second lowest ones. The converged trust values of adversary peers using exaggerate positives are higher than those using exaggerate negatives. This is because we use an asymmetric penalization mechanism for inaccurate replies ($c_1 > 1$ in Equation 4.1). We penalize more heavily peers that untruthfully report lower risks than those which untruthfully report higher risks.

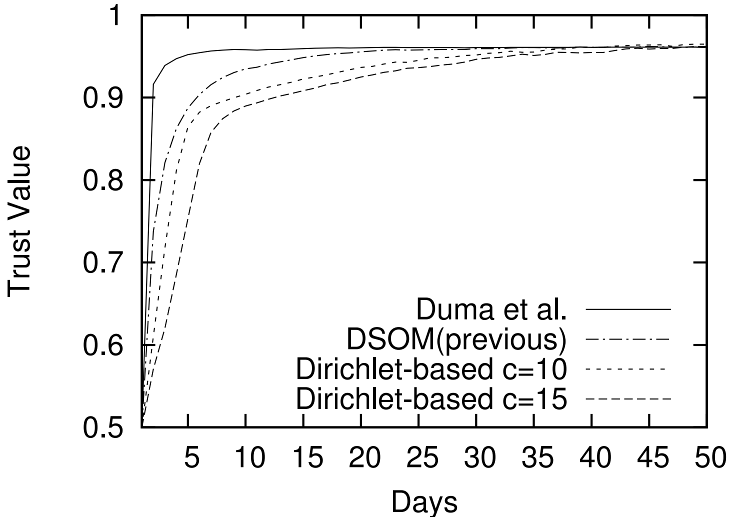


Figure 4.7: Trust Values of Newcomers under Different Trust Models

4.5.6 Robustness of Our Trust Model

The goal of this experiment is to study the robustness of our trust model against various insider attacks. For the newcomer attack, malicious peers white-wash their bad history and re-register as new users to the system. If the trust value of a newcomer can increase quickly based on its short term good behavior, the system is then vulnerable to newcomer attacks. However, a newcomer attack is difficult to succeed in our model. In our model, we use parameter c_0 in Equation 4.6 to control the trust value increasing rate. When c_0 is larger, it takes longer for a newcomer to gain a trust value above the trust threshold.

We compare our Dirichlet-based model with our previous model [56] and the model of Duma et al. [45] in Figure 4.7. We observe that in the Duma et al. model, the trust values of new users increase very fast and reach the aggregation trust threshold (0.8) in the first day, which reveals a high vulnerability to newcomer attacks. The reason for this is that their model does not have an initial trust to new peers and therefore their trust values change fast in the beginning. In the model we developed in [56], the trust values increase in a slower manner and reach the trust threshold after three days. However, that model is not flexible in that it does not offer control over the trust increase speed. In the Dirichlet-based model, the trust increase speed is controlled by the priori constant c_0. For

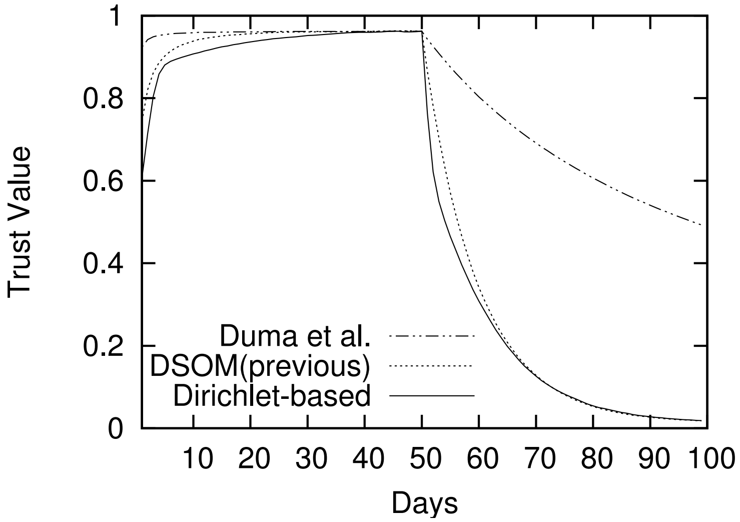


Figure 4.8: Trust of Malicious Peers under Betrayal Attack

$c_0 = 10$, it takes a newcomer four to five days of consistent good behavior to reach the same trust value. Larger values of c_0 make it even slower to reach high trust, hence offering robustness against newcomer attacks.

The second possible threat is the betrayal attack, where a malicious peer first gain a high trust value and then suddenly starts to act dishonestly. This scenario can happen, for example, when a peer is compromised. To demonstrate the robustness of our model against this attack type, we set up a scenario where u has seven peers in its acquaintance list, of which six are honest with an expertise level evenly divided between low, medium, and high. The malicious one has high expertise and behaves honestly in the first 50 days. After that, it launches a betrayal attack by adopting a maximal harm deceptive strategy. We observe the trust value of the betraying peer and the satisfaction levels of aggregated feedback in each day with respect to u.

Figure 4.8 shows the trust value of the betraying peer before and after the launching of the betrayal attack when respectively using Duma et al., our previous and our current trust models. For the Duma et al. model, the trust value of the malicious peer slowly drops after the betrayal attack. This is because their model does not use a forgetting factor, hence providing the previous honest behavior of a malicious peer with a heavy impact on the trust calculation for a considerable amount of time. The trust value of the betraying

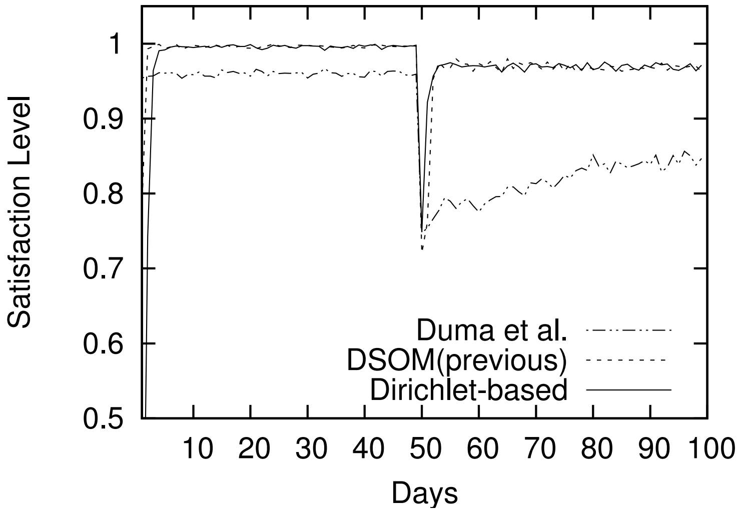


Figure 4.9: Impact on Accuracy of Betrayal Attack

peer drops much faster using our previous model, while the fastest rate is observed when using our Dirichlet-based model. This is because both models use a forgetting factor to pay more attention to the more recent behavior of peers.

We also notice that the Dirichlet-based model has a slight improvement over our previous model. The Dirichlet-based model adopts the dynamic test message rate and can react more swiftly. The rate of sending messages to malicious peers increases as soon as they start behaving dishonestly. Higher rates of test messages help in faster detection of dishonest behavior. However, in our previous model, the test message rate remains the same. This phenomenon can be further observed in Figure 4.10.

The results for the satisfaction levels of aggregated feedback with respect to u before and after the betrayal attack are shown in Figure 4.9. We notice that the satisfaction level of u for the aggregated feedback drops down drastically in the first day following the learning period and recovers after that in all three models. The recovery period is however much shorter for the Dirichlet-based and our previous models. This is again attributed to the use of the forgetting factor. The Dirichlet-based model has a slight improvement in the recovering speed over our previous model. This is because in the Dirichlet-based model, the trust values of betraying peers drop under the aggregation threshold faster than our previous model. Therefore, the impact of betraying peers is eliminated earlier than that

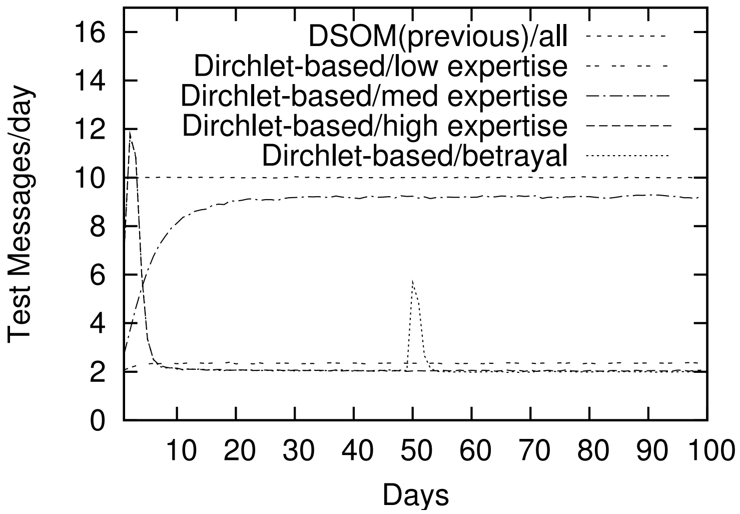


Figure 4.10: Comparison of Average Test Message Rates under Different Models

in the previous model.

4.5.7 Scalability of our Trust Model

The result of test message rates under betrayal attack is shown in Figure 4.10. We notice that in our Dirichlet-based model, the average test message rates for highly trustworthy as well as highly untrustworthy peers are the lowest. The average test message sending rate to peers with the medium expertise level is higher but still below the medium rate (R_m). Compared to our previous model, the average message sending rate is much lower, which demonstrates the improved scalability of our Dirichlet-based model. Note that the spike from the betraying group on around day 50 is caused by the drastic increment of the test message rate. The sudden change of a highly trusted peer behavior will cause the trust confidence level to drop down quickly. The rate of sending messages to this peer then switches to R_h accordingly.

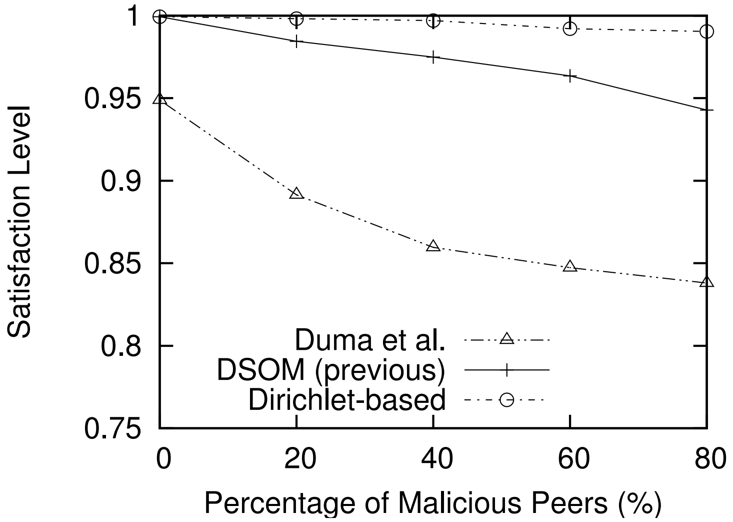


Figure 4.11: Aggregated Feedback under Inconsistency Attack

4.5.8 Efficiency of our Trust Model

To demonstrate the efficiency of our Dirichlet-based trust model, we conduct another experiment to evaluate the intrusion detection accuracy. In this experiment, we let peer u have 15 acquaintances, which are evenly divided into low, medium, and high expertise groups. Among the expert peers, some are malicious and launch inconsistency attacks synchronously to degrade the efficiency of the IDN. More specifically, in each round of behavior changing, these malicious peers adopt the maximal harm deception strategy for two days followed by six days of honest behavior.

In Figure 4.11, we vary the percentages of malicious peers from 0% to 80%. We inject daily intrusions to peer u with medium difficulty (0.5) and random risk levels. We then plot the average satisfaction level for the aggregated feedback. We observe that our Dirichlet-based model outperforms the others. This is because the dynamic test message rate in Dirichlet-based model causes the trust of malicious peers to drop faster and increase more slowly, hence minimizing the impact of dishonest behavior. Among the three models, Duma et al. has the least satisfaction level because of its slow response to sudden changes in peer behavior and its aggregation of all feedback from even untrustworthy peers.

Figure 4.12 shows the success rate of peer u in detecting intrusions. We notice that both

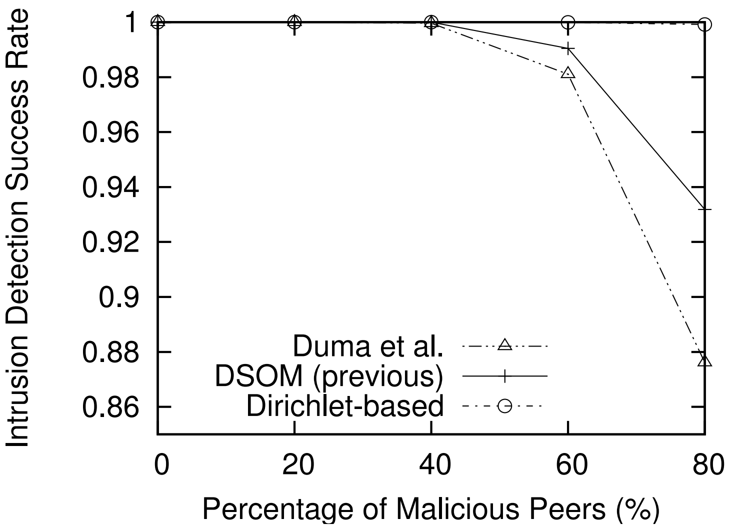


Figure 4.12: Intrusion Detection Success Rate under Inconsistency Attack

our previous model and the Duma et al. model cannot effectively detect intrusions when the majority of peers are malicious. Our Dirichlet-based model shows excellent efficiency in intrusion detection even in the situation of a dishonest majority.

4.6 Related Work

Most of the existing work on distributed collaborative intrusion detection relies on the assumption that all IDSs are trustworthy and faithfully report intrusion events. The Indra system [67] distributes among peers information about attack attempts on different machines so as to proactively react and increase the chance of detecting an attack. This system also allows peer neighbors to share information about intrusion attempts in order to enhance the overall system security. Another example is the distributed intrusion alert fusion system called Cyber Disease Distributed Hash Table (CDDHT) [75]. The CDDHT system provides several load balancing schemes to evenly distribute intrusion alarms among the sensor fusion centers in order to increase the scalability, fault-tolerance and robustness of the system. However, the systems mentioned above are all vulnerable to malicious IDS attacks. False information about intrusion events sent by malicious IDSs may heavily degrade the performance of these IDNs.

To protect an IDN, it is important to evaluate the trustworthiness of participating IDSs. ABDIAS [60] is a community based IDN where IDSs are organized into groups and exchange intrusion information to gain better intrusion detection accuracy. A simple majority-based voting system was proposed to detect compromised nodes. However, such system is vulnerable to colluded voting. Duma et al. [45] propose to address possibly malicious IDSs (peers) by introducing a trust-aware collaboration engine for correlating intrusion alerts. Their trust management scheme uses each peer's past experience to predict others' trustworthiness. However, their trust model is simplistic and does not address security issues within the collaborative network. For instance, in their system, the peer's past experience has the same impact on the final trust values of others, and therefore is vulnerable to betrayal attacks where compromised peers suddenly change their behavior. In our model, we use a forgetting factor when calculating trust, in order to rely more on the peer's recent experience and be robust to the changes of other peers' behavior. Our previous work [56] proposed a robust trust management model that uses test messages to gain personal experience and a forgetting factor to emphasize most recent experiences. However, this model needs to repeatedly aggregate all past experience with a peer when updating its trust, which makes it not scalable over time. It uses a linear model to calculate the average satisfaction levels of past interactions, and lacks a theoretical foundation. Also this approach does not capture trust modeling uncertainties or provide statistical confidence information on intrusion decisions. Our new model uses Dirichlet distributions to model peer trustworthiness. It makes use of dynamic test message rates in order to allow for better scalability. Also, our new model further improves robustness over our previous one through the use of flexible test message rates.

The Dirichlet family has been used in reputation modeling to handle the multi-level rating problem [70, 69]. Researchers in multi-agent systems have also been developing trust models to evaluate the trustworthiness of buying and selling agents in e-marketplaces [122]. One of the earliest trust models developed by Marsh [79] computes the trustworthiness of selling agents by taking into account direct interactions between buying and selling agents. The trust-oriented learning strategy proposed by Tran and Cohen [106] uses reinforcement learning to determine the trustworthiness of selling agents, after the true value of delivered goods is evaluated and compared to the buying agent's expected value for the goods. Selling agents can be classified as untrustworthy if their trust values fall below a certain threshold and buying agents try to select the trustworthy selling agent with the highest expected value for the goods. The Beta Reputation System (BRS) of Whitby et al. [114] and the TRAVOS model of Teacy et al. [105] estimate the trustworthiness of a selling agent by employing a Beta probability density function representing a probability distribution of a continuous variable. The work of Zhang and Cohen [122] focuses on coping with inaccurate reputation information about selling agents shared by malicious buying agents in e-marketplaces. The REGRET model of Sabater et al. [96] offers a multi-dimensional view of trust that includes a social dimension taking into consideration the social relationships among agents. However, it is difficult to clearly determine social relationships among IDSs in IDNs.

Our model is different from the above trust models in several aspects. First, our model is focused on long-term collaboration trust. Repetitive direct interactions between two agents are common in IDN environment. Second, the cost of experience in IDN is much lower than in e-commerce and it allows IDSs to send test messages to better establish trust relationships with others. Third, our model uses fine-grained experience quality rather than a binary measurement such as “good” or “bad”. Instead, it is categorized into multiple levels. Finally, our model uses direct trust modeling rather than reputation models. It is because the latter may suffer from collusion attacks where a group of malicious IDSs cooperate together by providing false reputation information about some IDSs to bad-mouth these targets for example.

Different reputation models were proposed in distributed systems [68, 103, 80]. These reputation models allow peers to get advice when evaluating the trustworthiness of other peers. For example, [68] uses a global reputation management to evaluate distributed trust by aggregating votes from all peers in the network. Sun et al. [103] propose for the communication in distributed networks an entropy-based model and a probability-based one. The models are used to calculate indirect trust, propagation trust and multi-path trust. They however involve a lot of overhead which limits their scalability. Another important concern is that IDSs can be easily compromised and become deceptive when reporting the trustworthiness of others. The reputation models for peer-to-peer networks, such as PowerTrust [91], TrustGuard [101], Malicious detector [82, 81], and Fine-Grained reputation [123] are capable of detecting malicious peers. However, they are purposed to detect deceiving nodes in a P2P network and can not be directly used in IDNs to improve the intrusion detection accuracy. A trust model in IDN should not only detect malicious nodes but also improve the intrusion detection accuracy overall and offer robustness and scalability.

4.7 Conclusions and Future Work

In this chapter, we presented a trust management model for evaluating trustworthiness of Intrusion Detection Systems in a Collaborative Intrusion Detection Network. Our trust management uses Dirichlet density functions as its foundation, and is accordingly able to measure the uncertainty in estimating the likely future behavior of IDSs. The measured uncertainty allows our trust management to employ an adaptive message exchange rate, resulting in good scalability. Equipped with a forgetting factor, it is also robust against some common threats. The effectiveness, robustness and scalability of our trust management have been further validated through experiments carried out in a simulated Collaborative Intrusion Detection Network. IDSs in the conducted experiments have different levels of expertise in detecting intrusions and adopt different deception strategies. The results show that our trust management is more effective compared to existing trust models for intrusion detection networks. This is an important step forward since effective trust management is essential for the deployment of secure IDNs.

One possible direction for future work here is to incorporate a reputation model in our trust management. This will require addressing the important issues of inaccurate reputation information, scalability and collusion attacks.

Chapter 5

Collaborative Decision

5.1 Introduction

Efficient and trustworthy feedback aggregation is a critical component in the design of IDNs. In this chapter, we intruduce FADEX[1], an efficient and trustworthy feedback aggregation mechanism for our IDN. In FADEX, each IDS in the IDN evaluates its peer collaborators based on their false positive and false negative rates, which can be estimated from historical data and test messages. Accordingly assessments received from an incompetent or malicious insider will have less weight in the final decisions. FADEX aggregation is based on data analysis and hypothesis testing methods. Specifically we design optimal decision rules that minimize Bayesian risks of IDSs in the network. In addition, for real-time applications, a host IDS only needs to consult a subset of its acquaintances until desired levels of performance, such as probabilities of detection and false alarm, are achieved. In other words, FADEX provides a data-driven efficiently-distributed sequential algorithm for IDSs to make decisions based on feedback from a subset of their collaborators. The goal is to reduce communication overhead and the computational resources needed to achieve a satisfactory feedback aggregation result when the number of acquaintances of an IDS is large. We consider four possible outcomes of a decision: *false positive* (FP), *false negative* (FN), *true positive* (TP), and *true negative* (TN). Each outcome is associated with a cost. Our proposed sequential-hypothesis testing-based feedback aggregation provides

[1] FADEX stands for "Feedback Aggregation for collaborative intrusion Detection nEtworKS"

improved cost efficiency as compared to other heuristic methods such as the simple average model [93] and the weighted average model [45, 57]. Communication overhead is reduced since the IDS aggregates feedback until a predefined FP and TP goal is reached. An analytical model is used to estimate the number of acquaintances needed for an IDS to reach its predefined intrusion detection goal. Such a result is crucial to the design of an IDS acquaintance list in our IDN.

The remainder of this chapter is organized as follows. In Section 5.2, we survey some IDN trust models and feedback aggregation techniques. The decision aggregation problem is formulated in Section 5.3, where we use hypothesis testing to minimize the cost of decisions, and sequential hypothesis testing to form consultation termination policy for predefined goals is described in Section 5.4. In Section 5.5, we use a simulations to evaluate the effectiveness of FADEX and validate the analytical model. Section 5.6 concludes the chapter and identifies directions for future research.

5.2 Related Work

Recent studies on IDNs [56, 57, 60] have proposed the use of trust models to identify dishonest peers. Intrusion assessments from nodes with different trust values are assigned with different weights to improve intrusion detection accuracy. ABDIAS [60] is a community-based IDN where IDSs are organized into groups and exchange intrusion information in order to gain better intrusion detection accuracy. A simple majority-based voting system is used to detect compromised nodes. However, this voting-based system is vulnerable to colluded voting. Another solution to detect compromised nodes is a trust management system where peers build trust with each other based on personal experience. Existing trust management models for IDN include the linear model [45], [56] and the Bayesian model [57, 53]. However, all these works used heuristic approaches to aggregate consultation results from other collaborators. In this chapter, we propose a Bayesian aggregation model which aims at finding optimal decisions based on collected information.

Bayesian approaches have been used in distributed detection in the past. Existing works, including [107] and [88], use Bayesian hypothesis testing methods to aggregate at a central data fusion center feedback from sensors distributed in a local area network. However, these methods require all participants to engage in every detection case, whereas in our context, IDSs may not be involved in all intrusion detections and the collected responses may come from different groups of IDSs each time.

The trustworthiness of IDNs has been ensured at various levels of the system architecture. In [133] and [132], a communication protocol with the property of reciprocal incentive compatibility has been used to provide IDS nodes incentives to send feedbacks to their peers, and hence to prevent malicious free-riders, denial-of-service attacks and dishonest insiders. However, this approach only ensures the reliability and trustworthiness at the communication overlay of the IDN, and it does not directly deal with the content of the feedback. In [131] and [52], a knowledge sharing mechanism has been proposed to allow expert nodes to disseminate knowledge within the IDN to prevent zero-day attacks. The communication protocols in [131] are implemented at the higher application layers of the collaborative network. In our work, we aim at ensuring security and trustworthiness at the "last mile" of the problem, i.e., feedback aggregation.

5.3 System Model

Consider a set of N nodes, $\mathcal{N} := \{1, 2, \cdots, N\}$, connected in a network, which can be represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. The set $\mathcal{E}$ contains the undirected links between nodes, indicating the acquaintances of IDSs in the network. An IDS node $i \in \mathcal{N}$ has a set of n_i acquaintances, denoted by $\mathcal{N}_i \subseteq \mathcal{N}$, with $n_i = |\mathcal{N}_i|$. When node i observes suspicious activities and does not have enough experience to make an accurate evaluation of potential intrusions, it can send out its observed intrusion information to its acquaintances to ask for diagnosis. The feedback from its acquaintances can be used to make a final decision. The input to the host IDS is the past history of each acquaintance regarding their detection accuracy, as well as their current feedbacks. The output is a decision on whether to raise an alarm or not.

Let $Y_j^i, j \in \mathcal{N}_i$, be a random variable denoting the decision of peer IDS $j, j \in \mathcal{N}_i$, on its acquaintance list $\mathcal{N}_i$ of node i. The random variable Y_j^i takes binary values in $\mathcal{Y} := \{0, 1\}$ for all $j \in \mathcal{N}_i, i \in \mathcal{N}$. In the intrusion detection setting, $Y_j^i = 0$ means that IDS j decides and reports to IDS i that there is no intrusion, while $Y_j^i = 1$ means that IDS j raises an alarm of possible detection of intrusion to IDS i. Each IDS makes its decision based upon its own experience of the previous attacks and its own sophistication of detection. We let

p_j^i as the probability mass function defined on $\mathcal{Y}$ such that $p_j^i(Y_j^i = 0)$ and $p_j^i(Y_j^i = 1)$ denote the probability of reporting no intrusion and the probability of reporting intrusion from IDS j to IDS i, respectively.

We let $\mathbf{Y}^i := [Y_j^i]_{j \in \mathcal{N}_i} \in \mathcal{Y}^{n_i}$ be an observation vector of IDS i that contains feedbacks from its peers in the acquaintance list. Each IDS has two hypotheses H_0 and H_1. H_0 hypothesizes that no intrusion is detected whereas H_1 forwards a hypothesis that intrusion is detected and alarm needs to be raised. Note that we intentionally drop the superscript i on H_0 and H_1 because we assume that each IDS attempts to make the same type of decisions. Denote by π_0^i, π_1^i the apriori probabilities on each hypothesis such that $\pi_0^i = \mathbb{P}[H_0], \pi_1^i = \mathbb{P}[H_1]$ and $\pi_0^i + \pi_1^i = 1$, for all $i \in \mathcal{N}$. Let p^i be the probability measure on $\mathcal{Y}^{n_i}$, for all $i \in \mathcal{N}$. The conditional probabilities $p^i(\mathbf{Y}^i = \mathbf{y}^i | H_l), l = 0, 1$, denote the probabilities of a complete feedback being $\mathbf{y}^i \in \mathcal{Y}^{n_i}$ given the hypothesis H_0, H_1, respectively. Assuming that peers make decisions independently (this is reasonable if acquaintances are appropriately selected), we can rewrite the conditional probability as

$$p^i(\mathbf{Y}^i = \mathbf{y}^i | H_l) = \prod_{j \in \mathcal{N}_i} p_j^i(Y_j^i = y_j^i | H_l), \ i \in \mathcal{N}, l = 0, 1. \tag{5.1}$$

Our goal is to decide whether the system should raise an alarm to the system administrator based on the current received feedbacks. We need to point out that the feedback aggregation does not exclude the local diagnosis of the host IDS itself. If an IDS is capable of making its own diagnosis, this one is aggregated with the feedbacks from its peers in the acquaintances. Table 5.1 summarizes the notations we use in this section.

In the following subsections, we first model the past behavior of acquaintances and then model the decision problem using Bayesian risk function.

5.3.1 Modeling of Acquaintances

The conditional probabilities $p_j^i(Y_j^i | H_l), i \in \mathcal{N}, j \in \mathcal{N}_i, l \in \{0, 1\}$, are often unknown to IDS nodes and they need to be learned from previous data. In this section, we use the beta distribution and its Gaussian approximation to find these probabilities. We let $p_{j,M}^i := p_j^i(Y_j^i = 0 | H_1)$ be the probability of miss of an IDS j's diagnosis to node i's request, also known as the false negative (FN) rate; and let $p_{j,F}^i := p_j^i(Y_j^i = 1 | H_0)$ be the probability of false alarm or false positive (FP) rate. The probability of detection, or true positive (TP) rate, can be expressed as $p^i_{j,D} = 1 - p^i_{j,M}$.

Each IDS in the network maintains a history of data containing the diagnosis data from past consultations. The accuracy of peer diagnosis will be revealed after an intrusion happens. As mentioned in Section 3.1, test messages can also be used to assess the effectiveness of IDSs even though no intrusion history has been collected. IDS i can use these collected data from its peers to assess the distributions over its peer IDS j's probabilities of detection and false alarm using beta functions, denoted by $p^i_{j,D}$ and $p^i_{j,F}$, respectively. The total reported diagnosis data from peer IDS $j, j \in \mathcal{N}_i$, to IDS i is denoted by the set $\mathcal{M}^i_j$, and they are classified into two groups: one is where the result is either false positive or true negative under no intrusion, denoted by the set $\mathcal{M}^i_{j,0}$; and the other is where the result is either false negative or true positive under intrusion, denoted by the set $\mathcal{M}^i_{j,1}$. Both sets are disjoint satisfying $\mathcal{M}^i_{j,0} \cup \mathcal{M}^i_{j,1} = \mathcal{M}^i_j$ and $\mathcal{M}^i_{j,0} \cap \mathcal{M}^i_{j,1} = \emptyset$.

We let the random variables $p^i_{j,F}$ and $p^i_{j,D}$ take the form of beta distributions as follows:

$$p^i_{j,F} \sim \ \mathrm{Beta}(x^i_j|\alpha^i_{j,F}, \beta^i_{j,F}) = \tfrac{\Gamma(\alpha^i_{j,F}+\beta^i_{j,F})}{\Gamma(\alpha^i_{j,F})\Gamma(\beta^i_{j,F})}(x^i_j)^{\alpha^i_{j,F}-1}(1-x^i_j)^{\beta^i_{j,F}-1}, \tag{5.2}$$

$$p^i_{j,D} \sim \ \mathrm{Beta}(y^i_j|\alpha^i_{j,D}, \beta^i_{j,D}) = \tfrac{\Gamma(\alpha^i_{j,D}+\beta^i_{j,D})}{\Gamma(\alpha^i_{j,D})\Gamma(\beta^i_{j,D})}(y^i_j)^{\alpha^i_{j,D}-1}(1-y^i_j)^{\beta^i_{j,D}-1}, \tag{5.3}$$

where $\Gamma(\cdot)$ is the Gamma function; $x^i_j, y^i_j \in [0,1]$; $\alpha^i_{j,F}, \alpha^i_{j,D}$ and $\beta^i_{j,F}, \beta^i_{j,F}$ are beta function parameters that are updated according to historical data as follows.

$$\alpha^i_{j,F} = \sum_{k \in \mathcal{M}^i_{j,0}} (\lambda^i_F)^{t^i_{j,k}} r^i_{j,F,k}, \quad \beta^i_{j,F} = \textstyle\sum_{k \in \mathcal{M}^i_{j,0}} (\lambda^i_F)^{t^i_{j,k}} (1 - r^i_{j,F,k}); \tag{5.4}$$

$$\alpha^i_{j,D} = \sum_{k \in \mathcal{M}^i_{j,1}} (\lambda^i_D)^{t^i_{j,k}} r^i_{j,D,k}, \quad \beta^i_{j,D} = \textstyle\sum_{k \in \mathcal{M}^i_{j,1}} (\lambda^i_D)^{t^i_{j,k}} (1 - r^i_{j,D,k}). \tag{5.5}$$

The introduction of the discount factors $\lambda^i_F, \lambda^i_D \in [0,1]$ above allows more weights on recent data from IDSs while less on the old ones. The discount factors on the data can be different for false negative and false positive rates. The parameter $t^i_{j,k}$ denotes the time when k-th diagnosis data is generated by IDS $j, j \in \mathcal{N}_i$, to its peer IDS i. The parameter $r^i_{j,F,k}, r^i_{j,M,k} \in \{0,1\}$ are the revealed results of the k-th diagnosis data: $r^i_{j,F,k} = 1$ suggests that the k-th diagnosis data from peer j yields an undetected intrusion while $r^i_{j,F,k} = 0$

means otherwise; similarly, $r^i_{j,D,k} = 1$ indicates the data from the peer j results in a correct detection under intrusion, and $r^i_{j,D,k} = 0$ means otherwise.

The parameters $\alpha^i_{j,F}, \beta^i_{j,F}, \alpha^i_{j,D}, \beta^i_{j,D}$ in the distribution above also provide an empirical assessment of the trustworthiness of each peer of IDS i. They can be also seen as the trust values of the collaborators. A peer who is either malicious or incompetent will result in low values of $\alpha^i_{j,D}$ and higher values $\alpha^i_{j,D}$. To make the parametric updates scalable to data storage and memory, we can use the following recursive formulae to update these parameters as follows:

$$\begin{aligned}\alpha^i_{j,e,k} &= (\lambda^i_e)^{t^i_{j,k}-t^i_{j,k-1}}\alpha^i_{j,e,k-1} + r^i_{j,e,k}, \quad k \geq 1, \qquad (5.6)\\ \beta^i_{j,e,k} &= (\lambda^i_e)^{t^i_{j,k}-t^i_{j,k-1}}\beta^i_{j,e,k-1} + r^i_{j,e,k}, \quad k \geq 1, \qquad (5.7)\end{aligned}$$

where $e \in \{F, D\}$; $\alpha^i_{j,D,k}, \beta^i_{j,D,k}$, are parameter values up to k-th data point in their corresponding data set $\mathcal{M}^i_{j,1}$; $\alpha^i_{j,F,k}, \beta^i_{j,F,k}$, are parameter values up to k-th data point in their corresponding data set $\mathcal{M}^i_{j,0}$. We can see that when $\lambda^i_e = 0$, the system becomes memoryless, and when $\lambda^i_e = 1$, all past experiences are taken into account on equal basis. The online iterative calculations also provide a method to assess the trust values with real time data.

When parameters of the beta functions α and β in (5.2) are sufficiently large, i.e., enough data are collected, beta distribution can be approximated by a Gaussian distribution as follows:

$$\mathrm{Beta}(\alpha, \beta) \approx \mathrm{G}\left(\frac{\alpha}{\alpha+\beta}, \sqrt{\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}\right), \qquad (5.8)$$

where the arguments of $G(\cdot, \cdot)$ are the mean value and the standard deviation, respectively. Note that we have dropped the superscripts and subscripts in (5.8) for generality as it can be applied to all i and j in (5.2). Hence, using the Gaussian approximation and (5.4), the expected values for $p^i_{j,D}$ and $p^i_{j,M}$ are given by

$$\mathbb{E}[p^i_{j,F}] = \frac{\alpha^i_{j,F}}{\alpha^i_{j,F} + \beta^i_{j,F}}, \quad \mathbb{E}[p^i_{j,D}] = \frac{\alpha^i_{j,D}}{\alpha^i_{j,D} + \beta^i_{j,D}}. \qquad (5.9)$$

The mean values in (5.9) under large data can be intuitively interpreted as the proportion of results of false alarm and detection in the set $\mathcal{M}^i_{j,0}$ and $\mathcal{M}^i_{j,1}$, respectively. They can thus be used in (5.1) as the assessment of the conditional probabilities.

5.3.2 Feedback Aggregation

The feedback aggregation problem of IDS i can be seen as a hypothesis testing problem in which one finds a decision function $\delta^i(\mathbf{Y}^i) : \mathcal{Y}^{n_i} \to \{0, 1\}$ to minimize the Bayes risk of IDS i

$$R^i(\delta^i) = R_0^i(\delta^i|H_0)\pi_0^i + R_1^i(\delta^i|H_1)\pi_1^i, \tag{5.10}$$

where $R^i(\delta^i|H_0)$ is the cost of false alarm and $R^i(\delta^i|H_1)$ is the cost of missed detection. An optimal decision function partitions the observation space $\mathcal{Y}^{n_i}$ into two disjoint sets $\mathcal{Y}_0^i$ and $\mathcal{Y}_1^i$, where $\mathcal{Y}_0^i = \{\mathbf{y}^i : \delta^i(\mathbf{y}^i) = 0\}$, and $\mathcal{Y}_1^i = \{\mathbf{y}^i : \delta^i(\mathbf{y}^i) = 1\}$.

To find an optimal decision function according to some criterion, we introduce the cost function $C_{ll'}^i, l, l' = 0, 1$, which represents IDS i's cost of deciding that H_l is true when $H_{l'}$ holds. More specifically, C_{01}^i is the cost associated with a missed intrusion or attack and C_{10}^i refers to the cost of false alarm, while C_{00}^i, C_{11}^i are the incurred costs when the decision meets the true situation. Let

$$\begin{aligned} R_0^i(\delta^i|H_0) &= C_{10}^i p^i[\delta^i = 1|H_0] + C_{00}^i p^i[\delta^i = 0|H_0], \qquad (5.11)\\ R_1^i(\delta^i|H_0) &= C_{01}^i p^i[\delta^i = 0|H_1] + C_{11}^i p^i[\delta^i = 1|H_1]. \qquad (5.12)\end{aligned}$$

It can be shown that decision functions can be picked as function of the likelihood ratio given by $L^i(\mathbf{y}^i) = \frac{p^i(\mathbf{y}^i|H_1)}{p^i(\mathbf{y}^i|H_0)}$ (see [88, 107]).

A threshold Bayesian decision rule is expressed in terms of the likelihood ratio and is given by

$$\delta_B^i(\mathbf{y}^i) = \begin{cases} 1 & \text{if } L^i(\mathbf{y}^i) \geq \tau^i \\ 0 & \text{if } L^i(\mathbf{y}^i) < \tau^i \end{cases}, \tag{5.13}$$

where the threshold τ^i is defined by

$$\tau^i = \frac{(C_{10}^i - C_{00}^i)\pi_0^i}{(C_{01}^i - C_{11}^i)\pi_1^i}. \tag{5.14}$$

If the costs are symmetric and the two hypothesis are equal likely, then the rule in (5.13) reduces to the maximum likelihood (ML) decision rule

$$\delta_{ML}^i(\mathbf{y}) = \begin{cases} 1 & \text{if } p^i(\mathbf{y}^i|H_1) \geq p^i(\mathbf{y}^i|H_0) \\ 0 & \text{if } p^i(\mathbf{y}^i|H_1) < p^i(\mathbf{y}^i|H_0) \end{cases}, \tag{5.15}$$

Assume that $C^i_{00}, C^i_{11} = 0$. Using the results in Section 5.3.1, we can obtain the following decision rule for each IDS. The application of the optimal decision rules is summarized in Algorithm 1.

Proposition 5.3.1 *Let* $\bar{\tau}^i := \frac{C^i_{10}}{C^i_{10}+C^i_{01}}$ *and assume that historical data is relatively large. The optimal decision rule of IDS* $i, i \in \mathcal{N}$, *is*

$$\delta^i = \begin{cases} \textit{1 (Alarm)} & \textit{if } \bar{P}^i \geq \bar{\tau}^i, \\ \\ \textit{0 (No alarm)} & \textit{otherwise.} \end{cases} \tag{5.16}$$

where $\bar{P}^i$ *can be obtained by Gaussian approximation as follows:*

$$\bar{P}^i \quad \approx \quad \frac{1}{1 + \frac{\pi^i_0}{\pi^i_1}\prod_{j=1}^{n_i} \frac{\alpha^i_{j,D}+\beta^i_{j,D}}{\alpha^i_{j,F}+\beta^i_{j,F}}(\frac{\alpha^i_{j,F}}{\alpha^i_{j,D}})^{y^i_j}(\frac{\beta^i_{j,F}}{\beta^i_{j,D}})^{1-y^i_j}}.$$

The corresponding Bayes risk for the optimal decision is

$$R^i(\delta^i) = \begin{cases} C^i_{10}(1-\bar{P}^i) & \textit{if } \bar{P}^i \geq \bar{\tau}^i, \\ \\ C^i_{01}\bar{P}^i & \textit{otherwise.} \end{cases} \tag{5.17}$$

Proof The result follows directly from the applications of likelihood ratio test and the Gaussian approximations of beta distributions under the assumption of large data sets.

5.4 Sequential Hypothesis Testing

The optimal decision rule in Section 5.3 requires each IDS to send requests to all the acquaintances. As the number of collaborators increases, it creates a lot of communication overhead and consumes a large amount of computational power to implement Algorithm 1. Instead, it is desirable that IDSs can choose a sufficient number of acquaintances to

Algorithm 1 Optimal Decision Rule for an IDS i

```
Step 1: Send out requests to all acquaintances of IDS $i$ and collect their feedback results.
Step 2: Use (5.16) to decide whether an intrusion occurs or not, and take corresponding actions.
Step 3: Update the data sets $\mathcal{M}^i_{j,0}$, $\mathcal{M}^i_{j,1}$, with the diagnosis results of each peer $j, j \in \mathcal{N}_i$ when the fact has been revealed a posteriori.
Step 4: Calculate beta function parameters $\alpha^i_{j,F}, \alpha^i_{j,D}$ and $\beta^i_{j,F}, \beta^i_{j,F}$ using iterative schemes (5.6) and (5.7).
Step 5: Go to Step 1 when new decisions need to be made or the trustworthiness of new acquaintances need to be evaluated using test messages.
```

guarantee a certain level of confidence in the final feedback aggregation. In this section, we use sequential hypothesis testing to make decisions with minimum number of feedbacks from peer IDSs, [112], [73]. An IDS asks for feedback from its acquaintances until a sufficient number of answers is collected. Let Ω^i denote all possible collections of feedback from the acquaintance list of IDS i and $\omega^i \in \Omega^i$ denotes a particular collection of feedback. Let $N^i(\omega^i)$ be a random variable denoting the number of feedbacks used until a decision is made. A sequential decision rule is formed by a pair (ϕ, δ), where $\phi^i = \{\phi^i_n, n \in \mathbb{N}\}$ is a stopping rule and $\delta^i = \{\delta^i_n, n \in \mathbb{N}\}$ is the terminal decision rule. Introduce a stopping rule with n feedback, $\phi^i_n : \mathcal{Y}^i_n := \prod_{j \in \mathcal{N}_{i,n}} \mathcal{Y} \to \{0, 1\}$, where $\mathcal{N}_{i,n}$ is the set of nodes an IDS i asks up to time n. $\phi^i_n = 0$ indicates that IDS i needs to take more samples after n rounds whereas $\phi^i_n = 1$ means to stop asking for feedback and a decision can be made by the rule δ^i_n. The minimum number of feedbacks is given by

$$N^i(\omega^i) = \min\{n : \phi^i_n = 1, n \in \mathbb{N}\}. \tag{5.18}$$

Note that $N^i(\omega^i)$ is the stopping time of the decision rule. The decision rule δ^i is not used until N. We assume that no cost has incurred when a correct decision is made while the cost of a missed intrusion is denoted by C^i_{01} and the cost of a false alarm is denoted by C^i_{10}. In addition, we assume each feedback incurs a cost D^i. We introduce an optimal sequential rule that minimizes Bayes risk given by

$$R^i(\phi^i, \delta^i) = R(\phi^i, \delta^i | H_0)\pi^i_0 + R(\phi^i, \delta^i | H_1)\pi^i_1, \tag{5.19}$$

where $R(\phi^i, \delta^i | H_l), l = 0, 1$, are the Bayes risks under hypotheses H_0 and H_1, respectively:

$$R^i(\phi^i, \delta^i | H_0) = C_{10}^i p^i [\delta_N(Y_j^i, j \in \mathcal{N}_{i,N}) = 1 | H_0] + D^i \mathbb{E}[N | H_0],$$
$$R^i(\phi^i, \delta^i | H_1) = C_{01}^i p^i [\delta_N(Y_j^i, j \in \mathcal{N}_{i,N}) = 0 | H_1] + D^i \mathbb{E}[N | H_1].$$

Let $V^i(\pi_0^i) = \min_{\phi^i, \delta^i} R^i(\phi^i, \delta^i)$ be the optimal value function. It is clear that when no feedback are obtained from the peers, the Bayes risks reduce to

$$\begin{aligned} R^i(\phi_0^i = 1, \delta_0^i = 1) &= C_{10}^i \pi_0^i, & (5.20) \\ R^i(\phi_0^i = 1, \delta_0^i = 0) &= C_{01}^i \pi_1^i. & (5.21) \end{aligned}$$

Hence, H_1 is chosen when $C_{10}^i \pi_0^i < C_{01}^i \pi_1^i$ or $\pi_0^i < \frac{C_{01}^i}{C_{10}^i + C_{01}^i}$, and H_0 is chosen otherwise. The minimum Bayes risk under no feedback is thus obtained as a function of π_0^i and is denoted by

$$T^i(\pi_0^i) = \begin{cases} C_{10}^i \pi_0^i & \text{if } \pi_0 < \frac{C_{01}^i}{C_{10}^i + C_{01}^i}, \\ C_{01}^i (1 - \pi_0^i) & \text{otherwise.} \end{cases} \tag{5.22}$$

The minimum cost function (5.22) is a piecewise linear function. For ϕ^i such that $\phi_0^i = 0$, i.e., at least one feedback is obtained, let the minimum Bayes risk be denoted by $J^i(\pi_0^i) = \min_{\{(\phi^i, \delta^i) : \phi_0^i = 0\}} R^i(\phi^i, \delta^i)$. Hence, the optimal Bayes risk needs to satisfy

$$V^i(\pi_0^i) = \min\{T^i(\pi_0^i), J^i(\pi_0^i)\}. \tag{5.23}$$

Note that $J^i(\pi_0^i)$ must be greater than the cost of one sample D^i as a sample request incurs D^i and $J^i(\pi_0^i)$ is concave in π_0^i as a consequence of minimizing the linear Bayes risk (5.19). If the cost D^i is high enough so that $J^i(\pi_0^i) > T^i(\pi_0^i)$ for all π_0^i, then no feedback will be requested. In this case, $V^i(\pi_0^i) = T^i(\pi_0^i)$, and the terminal rule is described in (5.22). For other values of $D^i > 0$, due to the piecewise linearity of $T^i(\pi_0^i)$ and concavity of $J^i(\pi_0^i)$, we can see that $J^i(\pi_0^i)$ and $T^i(\pi_0^i)$ have two intersection points π_L^i and π_H^i such that $\pi_L^i \leq \pi_H^i$. It can be shown that for some reasonably low cost D^i and π_0^i such that $\pi_L^i < \pi_0^i < \pi_H^i$, an IDS optimizes its risk by requesting another feedback; otherwise, an IDS should choose to raise an alarm when $\pi_0^i \leq \pi_L^i$ and report no intrusion when $\pi_0^i \leq \pi_L^i$.

Assuming that it takes the same cost D^i for IDS i to acquire a feedback, the problem has the same form after obtaining a feedback from a peer. IDS i can use the feedback to update its apriori probability. After n feedback are obtained, π_0^i can be updated as follows:

$$\pi_0^i(n) = \frac{\pi_0^i}{\pi_0^i + (1 - \pi_0^i) L_n^i}; \tag{5.24}$$

where $L_n^i := \prod_{j \in \mathcal{N}_{i,n}} \frac{p^i(y_j^i|H_1)}{p(y_j^i|H_0)}$. We can thus obtain the optimum Bayesian rule captured by Algorithm 1 below, known as the sequential probability ratio test (SPRT) for a reasonable cost D^i. The SPRT Algorithm 2 can be used to replace step 2 in Algorithm 1. In FADEX, it is important to note that the choice between Algorithm 2 and Algorithm 1 depends on the number of acquaintances of an IDS and its computational and memory resources. For smaller scale IDS networks or new members of the IDN, Algorithm 2 is more desirable because it allows IDSs to collect more data and learn the level of expertise and trustworthiness of their peers. However, Algorithm 2 becomes more efficient when an IDS has a large number of collaborators and limited resources.

Algorithm 2 SPRT Rule for an IDS i

Step 1: Start with $n = 0$. Use (5.25) as a stopping rule until $\phi_n^i = 1$ for some $n \geq 0$.

$$\phi_n^i = \begin{cases} 0 & \text{if } \pi_L^i < \pi_0^i(n) < \pi_H^i, \\ 1 & \text{otherwise.} \end{cases} \tag{5.25}$$

or in terms of the likelihood ratio L_n^i, we can use

$$\phi_n^i = \begin{cases} 0 & \text{if } A^i < L_n^i < B^i \\ 1 & \text{otherwise} \end{cases},$$

where $A^i = \frac{\pi_0^i(1-\pi_H^i)}{(1-\pi_0^i)\pi_H^i}$ and $B^i = \frac{\pi_0^i(1-\pi_L^i)}{(1-\pi_0^i)\pi_L^i}$.

Step 2: Go to Step 3 if $\phi_n^i = 1$ or $n = |\mathcal{N}_i|$; otherwise, choose a new peer from the acquaintance list to request a diagnosis and go to Step 2 with $n = n + 1$.

Step 3: Apply the terminal decision rule as follows to determine whether there is an intrusion or not.

$$\delta_n^i = \begin{cases} 1 & \text{if } \pi_0^i(n) \leq \pi_L^i \\ 0 & \text{if } \pi_0^i(n) > \pi_H^i \end{cases} \quad \text{or } \delta_n^i = \begin{cases} 1 & \text{if } L_n^i \leq A^i \\ 0 & \text{if } L_n^i > B^i \end{cases}$$

5.4.1 Threshold Approximation

In the likelihood sequential ratio test of Algorithm 2, the threshold values A and B need to be calculated by finding π_L^i and π_H^i from $J^i(\pi_0^i)$ and $T^i(\pi_0^i)$ in (5.23). The search for these values can be quite involved using dynamic programming. However, in this subsection, we introduce an approximation method to find the thresholds. The approximation is based on theoretical studies made in [112] and [73] where a random walk or martingale model is used to yield a relation between thresholds and false positive and false negative rates. Let P_D^i, P_F^i be the probability of detection and the probability of false alarm of an IDS i after applying the sequential hypothesis testing for feedback aggregation. We need to point out that these probabilities are different from the probabilities p_D^i, p_F^i discussed in the previous subsection, which are the raw detection probabilities without feedback in the collaborative network. Let $\bar{P}_D^i$ and $\bar{P}_F^i$ be reasonable desired performance bounds such that $P_F^i \leq \bar{P}_F^i,\ P_D^i \geq \bar{P}_D^i$. Then, the thresholds can be chosen such that $A^i = \frac{1-\bar{P}_D^i}{1-\bar{P}_F}^i,\ B^i = \frac{\bar{P}_D^i}{\bar{P}_F^i}$.

The next proposition gives a result on the bound of the users that need to be on the acquaintance list to achieve the desired performances.

Proposition 5.4.1 *Assume that each IDS makes independent diagnosis on its peers' requests and each has the same distribution $p_0^i = \bar{p}_0 := \bar{p}(\cdot|H_0), p_1^i = \bar{p}_1 := \bar{p}(\cdot|H_1)$, $\bar{p}_0(y_i = 0) = \theta_0, \bar{p}_1(y_i = 0) = \theta_1$, for all $i \in \mathcal{N}$.*

Let $D_{KL}(\bar{p}_0||\bar{p}_1)$ be the Kullback-Leibler (KL) divergence defined as follows.

$$\begin{aligned} D_{KL}(\bar{p}_0||\bar{p}_1) &= \sum_{k=0}^{1} \bar{p}_0(k) \ln \frac{\bar{p}_0(k)}{\bar{p}_1(k)}, && (5.26) \\ &= \theta_0 \ln \frac{\theta_0}{\theta_1} + (1-\theta_0) \ln \frac{1-\theta_0}{1-\theta_1} && (5.27) \end{aligned}$$

and likewise introduce the K-L divergence $D_{KL}(\bar{p}_1||\bar{p}_0)$. Then on the average, an IDS needs N_i acquaintances such that

$$N_i \geq \max\left(\left\lceil -\frac{D_M^i}{D_{KL}(\bar{p}_0||\bar{p}_1)} \right\rceil, \left\lceil \frac{D_F^i}{D_{KL}(\bar{p}_1||\bar{p}_0)} \right\rceil \right), \tag{5.28}$$

where $D_M^i = P_F \ln\left(\frac{P_D^i}{P_F^i}\right) + P_D \ln\left(\frac{1-P_D^i}{1-P_F^i}\right)$ and $D_F^i = P_F^i \ln\left(\frac{1-P_D^i}{1-P_F^i}\right) + P_D^i \ln\left(\frac{P_D^i}{P_F^i}\right)$. If $P_F^i \ll 1$ and $P_M^i \ll 1$, we need approximately N_i acquaitances such that

$$N_i \geq \max\left(\left\lceil \frac{P_D^i - 1}{D_{KL}(\bar{p}_0||\bar{p}_1)} \right\rceil, \left\lceil -\frac{P_F^i}{D_{KL}(\bar{p}_1||\bar{p}_0)} \right\rceil \right). \tag{5.29}$$

■

Proof The conditional expected number of feedback needed to reach a decision on the hypothesis in SPRT can be expressed in terms of P_F and P_D, [112], [73].

$$\begin{aligned}\mathbb{E}[N|H_0] = &\ \tfrac{1}{-D_{KL}(\bar{p}_0||\bar{p}_1)}\left[P_F^i \ln\left(\tfrac{P_D^i}{P_F^i}\right) + P_D^i \ln\left(\tfrac{1-P_D^i}{1-P_F^i}\right)\right],\\ \mathbb{E}[N|H_1] = &\ \tfrac{1}{D_{KL}(\bar{p}_1||\bar{p}_0)}\left[P_F^i \ln\left(\tfrac{1-P_D^i}{1-P_F^i}\right) + P_D^i \ln\left(\tfrac{P_D^i}{P_F^i}\right)\right],\end{aligned}$$

Hence, to reach a decision we need to have at least $\max\{\mathbb{E}[N|H_0], \mathbb{E}[N|H_1]\}$ independent acquaintances. Under the assumption that both P_F and P_M^i are much less than 1, we can further approximate

$$\mathbb{E}[N|H_0] \sim -\frac{1-P_D^i}{D_{KL}(\bar{p}_0||\bar{p}_1)}, \mathbb{E}[N|H_1] \sim -\frac{P_F^i}{D_{KL}(\bar{p}_1||\bar{p}_0)}.$$

These lead us to inequalities (5.29) and (5.28).

5.5 Performance Evaluation

In this section, we use a simulation approach to evaluate the efficiency of the FADEX feedback aggregation scheme and compare it with other heuristic approaches, such as the simple average aggregation and the weighted average aggregation (to be explained in more details in this section).

We conduct a set of experiments to evaluate the average cost of the collaborative detection using the FADEX aggregation model in comparison with the simple average and the weighted average models. We validate and confirm our theoretical results on the number of acquaintances needed for consultation. Each experimental result presented in this section is derived from the average of a large number of replications with an overall negligible confidence interval. The parameters we use are shown in Table I.

5.5.1 Simulation Setting

The simulation environment uses an IDN of n peers. Each IDS is represented by two parameters, expertise level l and decision threshold τ_p. Expertise level l represents the ability that the IDS catches suspicious traces from a given observation, and τ_p represents the sensitivity of the IDS (to be elaborated more in Section 5.5.2). At the beginning, each peer receives an initial acquaintance list containing all the other neighbor nodes. In the process of the collaborative intrusion detection, a node sends out intrusion information to its acquaintances to request for an intrusion assessment. The feedbacks collected from others are used to make a final decision, i.e., whether to raise an alarm or not. Different feedback aggregation schemes can be used to make such decisions. We implement three different feedback mechanisms, namely, simple average aggregation, weighted average aggregation, and FADEX aggregation. We compare their efficiency by the average cost of false decisions.

Simple Average Model

If the average of all feedback is larger than a threshold, then raise an alarm.

$$\delta_{SA} = \begin{cases} 1 \text{ (Alarm)} & \text{if } \frac{\sum_{k=1}^{n} \mathbf{y}_k}{n} \geq \tau_{SA}, \\ \\ 0 \text{ (No alarm)} & \text{otherwise,} \end{cases} \tag{5.30}$$

where τ_{SA} is the decision threshold for the simple average algorithm. It is set to be 0.5 if no cost is considered for making false decisions.

Weighed Average Model

Weights are assigned to feedbacks from different acquaintances to distinguish their detection capability. For example, high expertise IDSs are signed with larger weight compared to low expertise IDSs. In [45], [56], and [57], the weights are the trust values of IDSs:

$$\delta_{WA} = \begin{cases} 1 \text{ (Alarm)} & \text{if } \frac{\sum_{k=1}^{n} w_k \mathbf{y}_k}{\sum_{k=1}^{n} w_k} \geq \tau_{WA}, \\ \\ 0 \text{ (No alarm)} & \text{otherwise,} \end{cases} \tag{5.31}$$

where w_k is the weight of the feedback from acquaintance k, which is the trust value of acquaintance k in [45], [56], and [57]. τ_{WA} is the decision threshold for the weighted average algorithm. It is fixed to be 0.5 since no cost is considered for FP and FN. In this simulation, we adopt trust values from [57] to be the weights of feedbacks.

FADEX aggregation Model

As described in section 5.3.2, FADEX aggregation models each IDS with two parameters (FP and TP) instead of a single trust value. It also considers the costs of false positive and false negative decisions. The FADEX decision model investigates the cost of all possible decisions and chooses a decision which leads to a minimal expected cost.

5.5.2 Modeling of a single IDS

To reflect the intrusion detection capability of each peer, we use a Beta distribution to simulate the decision model of an IDS. A Beta density function is given by:

$$\begin{aligned} f(\bar{p}|\bar{\alpha},\bar{\beta}) &= \frac{1}{B(\bar{\alpha},\bar{\beta})}\bar{p}^{\bar{\alpha}-1}(1-\bar{p})^{\bar{\beta}-1}; \\ B(\bar{\alpha},\bar{\beta}) &= \int_0^1 t^{\bar{\alpha}-1}(1-t)^{\bar{\beta}-1}dt, \end{aligned} \tag{5.32}$$

$\bar{\alpha}$ and $\bar{\beta}$ are defined as follows.

$$\begin{aligned} \bar{\alpha} &= 1+\frac{l(1-d)}{d(1-l)}r, \\ \bar{\beta} &= 1+\frac{l(1-d)}{d(1-l)}(1-r). \end{aligned} \tag{5.33}$$

where $\bar{p} \in [0,1]$ is the assessment result from the host IDS about the probability of intrusion, and $f(\bar{p}|\bar{\alpha},\bar{\beta})$ is the distribution of assessment $\bar{p}$ from a peer with expertise level l to an intrusion with difficulty level $d \in [0,1]$. Higher values of d are associated with attacks that are difficult to detect, i.e., many peers may fail to identify them. Higher values of l imply a higher probability of producing correct intrusion assessment. $r \in \{0,1\}$ is the expected result of detection. $r = 1$ indicates that there is an intrusion and $r = 0$ indicates that there is no intrusion.

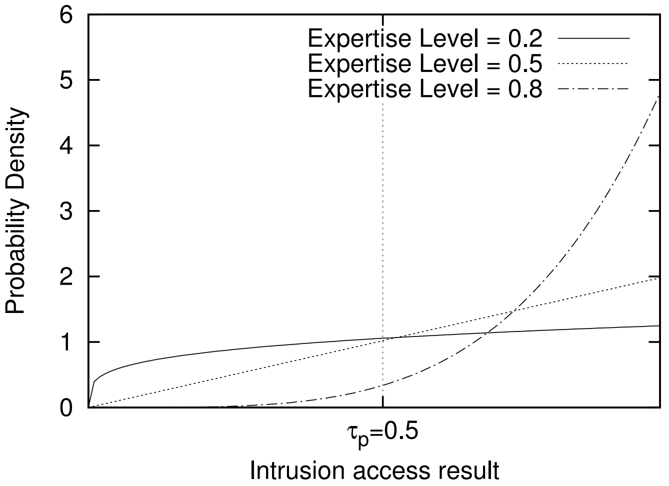


Figure 5.1: Expertise Level and detection rate

For a fixed difficulty level, the above model has the property of assigning higher probabilities of producing correct rankings to peers with higher levels of expertise. A peer with expertise level l has a lower probability of producing correct rankings for alerts of higher difficulty $(d > l)$. $l = 1$ or $d = 0$ represent the extreme cases where the peer can always accurately rank the alert. This is reflected in the Beta distribution by $\alpha, \beta \to \infty$. Figure 5.1 shows the feedback probability distribution for peers with different expertise levels, where we fix $r = 1$ and the difficulty level of test messages to 0.5.

τ_p is the decision threshold of $\bar{p}$. If $\bar{p} > \tau_p$, a peer sends feedback 1 (i.e., under-attack); otherwise, feedback 0(i.e., no-attack) is generated. τ_p indicates the sensitivity of an IDS detector, lower τ value implies a more sensitive detector. i.e., the IDS is more likely to raise alert when suspicious trace is noticed. For a fixed difficulty level, the preceding model assigns higher probabilities of producing correct intrusion diagnosis to peers with higher level of expertise. A peer with expertise level l has a lower probability of producing correct intrusion diagnosis for intrusions of higher detection difficulty $(d > l)$. $l = 1$ or $d = 0$ represent extreme cases where the peer can always accurately detect the intrusion. This is reflected in the Beta distribution by $\bar{\alpha}, \bar{\beta} \to \infty$.

Figure 5.2 shows that both the FP and FN decrease when the expertise level of an IDS increases. We notice that the curves of FP and FN overlap. This is because the IDS detection density distributions are symmetric under $r = 0$ and $r = 1$. Figure 5.3 shows

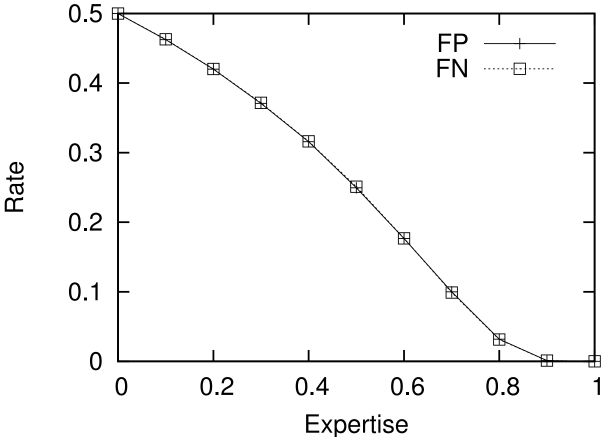


Figure 5.2: FP and FN vs. Expertise Level l

that the FP decreases with the decision threshold while the FN increases with the decision threshold. When the decision threshold is 0, all feedbacks are positive; when the decision threshold is 1, all feedbacks are negative.

5.5.3 Detection Accuracy and Cost

One of the most important metrics to evaluate the efficiency of a feedback aggregation is the average cost of incorrect decisions. We take into consideration the fact that the costs of FP decisions and FN decisions are different. In the following subsections, we evaluate the cost efficiency of the FADEX aggregation algorithm compared with other models under homogeneous and heterogeneous network settings. Then we study the relation between decision cost and the consulted number of acquaintances.

Cost Under Homogeneous Environment

In this experiment, we study the efficiency of the three aggregation models under a homogeneous network setting, i.e., all acquaintances have the same parameters. We fix the expertise levels of all nodes to be 0.5 (i.e., medium expertise) and set $C_{01} = C_{10} = 1$ for the fairness of comparison, since the simple average and the weighted average models do not account for the cost difference between FP and FN. We fix the decision threshold for each

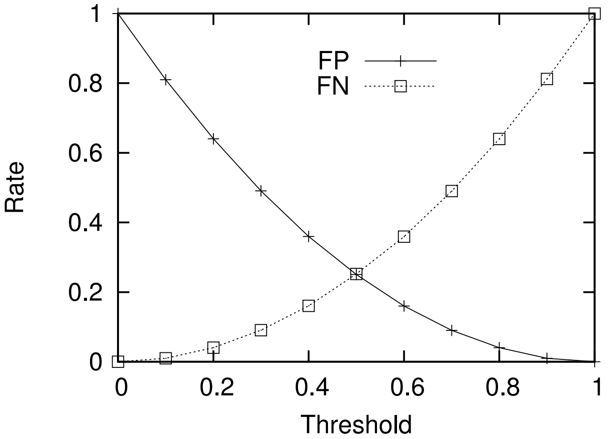


Figure 5.3: FP and FN vs. Threshold τ_p

IDS (τ_p) to 0.1 for the first batch run and then increase it by 0.1 in each following batch run until it reaches 1.0. We measure the average cost of the three aggregation models. As shown in Figure 5.4, the average costs on false decisions yielded by FADEX remains the lowest among the three under all threshold settings. The costs of the weighted average aggregation and the simple average aggregation are close to each other. This is because under such a homogeneous environment, the weights of all IDSs are the same. Therefore, the difference between the weighted average and the simple average is not substantial. We also observe that changing the threshold has a big impact on the costs of the weighted average model and the simple average model, while the cost of the FADEX model changes only slightly with the threshold. All costs reach a minimum when the threshold is 0.5 and increase when it deviates from 0.5.

Cost Under Heterogeneous Environment

In this experiment, we fix the expertise level of all peers to 0.5 and assign decision thresholds ranging from 0.1 to 0.9 to node 1 to 9 respectively with an increment of 0.1. We set false positive cost $C_{10} = 1$ and false negative cost $C_{01} = 5$ to reflect the cost difference between FP and FN. We observe the detection accuracy in terms of FP and FN rates and the average costs of false decisions at node 0 when three different feedback aggregation models are used.

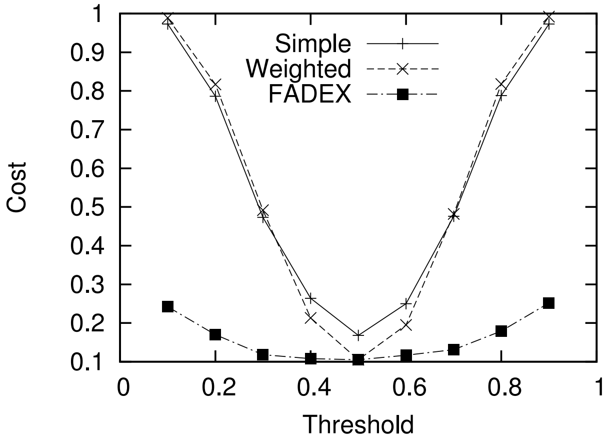


Figure 5.4: Average Cost vs. Threshold τ_p

Figure 5.5 shows that the average costs of the three different models converge after a few days of learning process. The cost of FADEX model starts with a high value and drops drastically in the first 10 days, and finally converges to a stable value on day 30. We then plot in Figure 5.6 the steady state FP, FN, and the cost. We observe that the weighted average model shows significant improvement in the FP and FN rates and cost compared to the simple average model. The FADEX model has a higher FP rate and a lower FN rate compared to the other two models. However, its cost is the lowest among the three. This is because the FADEX model trades some FP with FN to reduce the overall cost of false decisions.

Cost and the Number of Acquaintances

In this experiment, we study the relation between average cost due to false decisions and the number of acquaintances that the host IDS consults. We fix the expertise level of all IDSs in the network to $0.3, 0.5, 0.7, 0.8$ respectively for different batch runs. Every IDS decision threshold is fixed to $\tau_p = 0.5$ in all cases. We observe in Figure 5.7 that, under all cases, the average cost decreases when more acquaintances are consulted. We also notice that for higher expertise acquaintances, fewer consultations are needed to reach the cost goal. For instance, in our experiments, the host IDS only needs to consult 2 acquaintances on average to reach a cost of 0.1, under the case where all acquaintances are with high

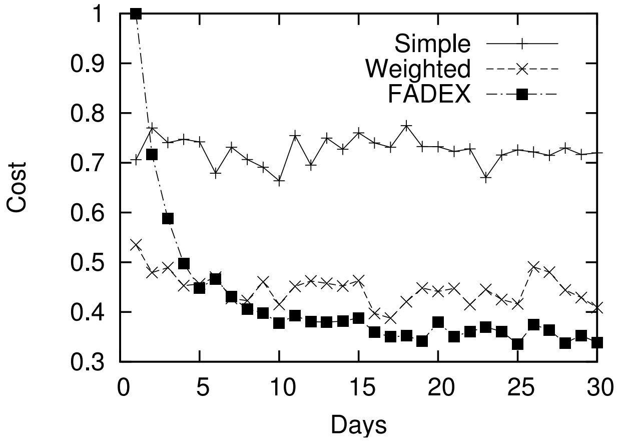


Figure 5.5: Average Costs for Three Different Aggregation Models

expertise level 0.8. Correspondingly, the number of acquaintances needed are 4 and 15 on average when the acquaintance expertise levels are 0.7 and 0.5 respectively. In the case that all acquaintances are 0.3 (i.e., of low expertise), the utility goal can not be reached after consulting a small number (i.e., < 20) of acquaintances.

In the next experiment, the expertise levels of all nodes remain 0.5 and their decision thresholds vary from 0.1 to 0.9. We set $C_{10} = C_{01} = 1$ in the first batch run and increase C_{01} by 1 in every subsequent batch run. We observe the costs under three different models. Figure 5.8 shows that the costs of the simple average model and the weighted average model increase linearly with C_{01} while cost of hypothesis testing model grows the slowest among the three. This is because the hypothesis testing model has a flexible threshold to optimize its cost. The hypothesis testing model has superiority when the cost difference between FP and FN is large.

5.5.4 Sequential Consultation

In this experiment, we study the number of acquaintances needed for consultation to reach a predefined goal. Suppose the TP lower-bound $\bar{P}_D = 0.95$ and FP upper-bound $\bar{P}_F = 0.1$. We observe the change of FP rate and TP rate with the number of acquaintances consulted (n). Figure 5.9 shows that FP rate decreases and TP rate increases with n. Consulting higher expertise nodes leads to a higher TP rate and a lower FP rate. In the

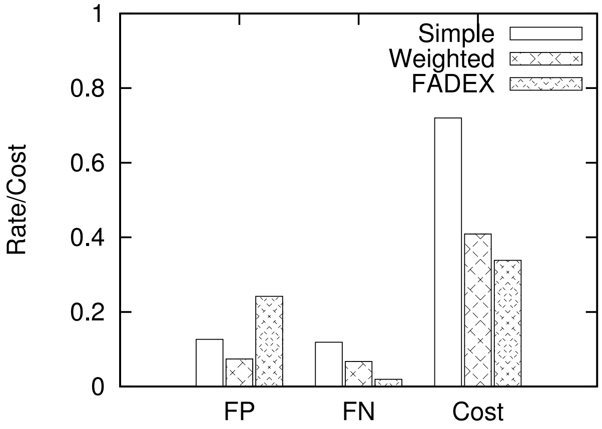


Figure 5.6: Comparison of Three Aggregation Models

next experiment we implement Algorithm 1 on each node and measure the average number of acquaintances needed to reach the predefined TP lower-bound and the FP upper-bound. Figure 5.10 compares the simulation results with the theoretical results (see Equation (5.29)), where the former confirms the latter. In both cases, the number of consultations decreases quickly with the expertise levels of acquaintances. For example, the IDS needs to consult around 50 acquaintances of expertise 0.2, while only 3 acquaintances of expertise 0.7 are needed for the same purpose. This is partly because low expertise nodes are more likely to make conflicting feedbacks and consequently increase the number of consultations. The analytical results can be useful for IDSs to design the size of their acquaintance lists.

5.5.5 Robustness and Scalability of the System

Robustness and scalability are two important features of an IDN. FADEX is robust to malicious insiders since it has an inherent robust trust management model from [57] where malicious insiders can be quickly discovered and removed from the acquaintance list. To verify this, we simulate the scenario of betrayal attack under a homogeneous environment. We fix all 10 IDSs with $l = 0.5$ and $\tau_p = 0.5$. We let one IDS turn malicious at day 20 and start to give opposite diagnosis. We observe the FP and TP rate of a malicious node and its impact on the decision of other nodes. From Figure 5.11 we can see that the FP rate and TP rate of the malicious node raise/drop quickly after day 20. Figure 5.12 shows

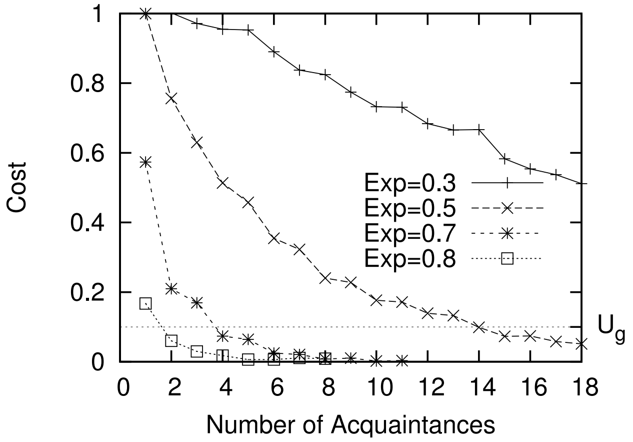


Figure 5.7: Average Cost vs. Number of Acquaintances Consulted (U_g is the cost goal)

that the cost of false decision cost of other normal nodes raises quickly at day 20 and drop back to normal after a few days. Compared with the other two aggregation models, the FADEX model receives the least impact from the malicious node.

This IDN is scalable since the number of acquaintances needed for consultation only depends on the expertise level of those acquaintances rather than the size of the network. Hence the message rate from/to each IDS does not grow with the number of nodes in the network. Furthermore, the dynamic consultation algorithm reduces the number of consultation messages needed for collaborative intrusion detections.

5.6 Conclusion

In this chapter, we have proposed FADEX, a mechanism for trustworthy feedback aggregation. We have obtained optimal decision rules that minimize Bayes risks using hypothesis testing methods, and provided a data-driven mechanism for real-time efficient, distributed and sequential feedback aggregations. In FADEX, an IDS consults sequentially for peer diagnoses until it is capable of making an aggregated decision that meets Bayes optimality. The decision is made based on a threshold rule leveraging the likelihood ratio approximated by beta distribution and thresholds by target rates. Our simulation results have shown that FADEX is superior to other proposed models in the literature in terms of cost effi-

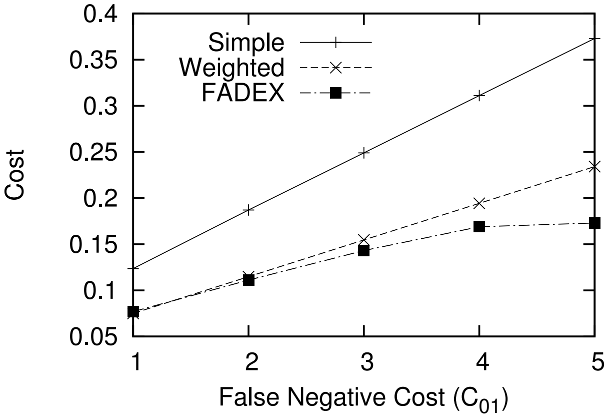


Figure 5.8: Cost vs. C_{01} under three models

ciency. Our simulation results have also corroborated our theoretical results on the average number of acquaintances needed to reach the predefined false positive upper-bound and true positive lower-bound. As future work, we want to investigate large-scale collaborative networks and their topological impact. Another possible research direction is to integrate FADEX with communication networks, and design defense mechanisms against different cyber attacks such as denial of service, man-in-the-middle and insider attacks. Finally, our results can be extended to deal with the case of correlated feedbacks.

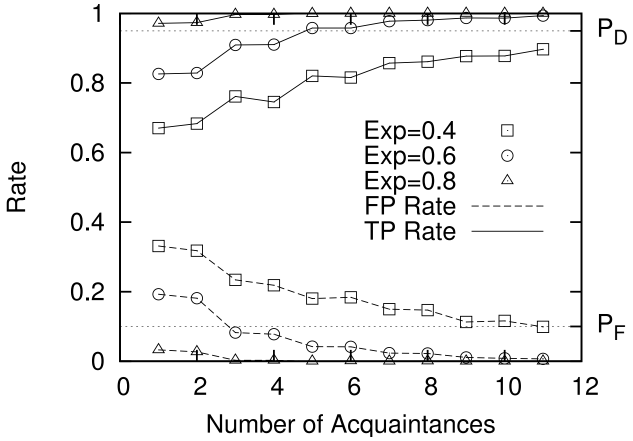


Figure 5.9: FP, TP vs. Number of Acquaintances

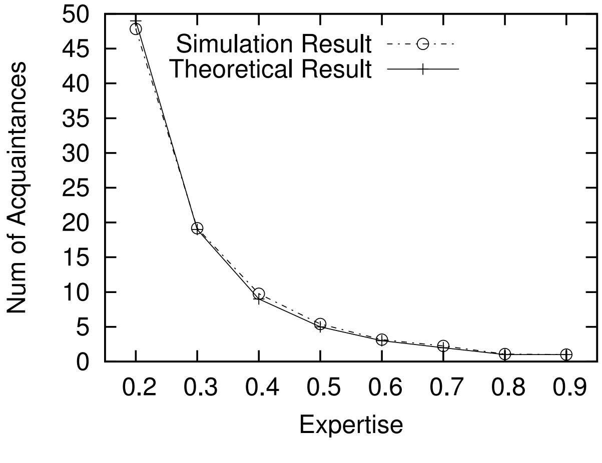


Figure 5.10: Number of Acquaintances vs. Expertise

Table 5.1: Summary of Notations

Symbol	Meaning
$\mathcal{N}$	Set of IDSs in the collaborative network
$\mathcal{N}_i$	Set of acquaintances of IDS $i, i \in \mathcal{N}$
n_i	number of acquaintances of IDS $i, i \in \mathcal{N}$
$Y_j^i,$	Reported decisions from IDS j to IDS i, $i \in \mathcal{N}, j \in \mathcal{N}_i$
$\mathbf{Y}^i$	Vector of complete feedbacks from IDS i's acquaintances
H_0	Hypothesis that there is no intrusion
H_1	Hypothesis that there is an intrusion
$r_{j,F,k}^i$	The diagnosis result at time k from acquaintance j to IDS i given that there is no intrusion
$r_{j,D,k}^i$	The diagnosis result at time k from acquaintance j to IDS j given that there is an intrusion
π_0^i, π_1^i	Prior probability of no-attack and under-attack
$\bar{\tau}^i$	Probability threshold for final decision
L^i	Likelihood ratio for IDS i's decision
L_n^i	Likelihood ratio for IDS i's sequential decision at stage n
R^i	Bayesian risk of IDS i
δ^i	Aggregation decision rule of IDS i
ϕ^i	Stopping decision rule of IDS i
$D_{KL}(p_1\|\|p_2)$	Kullback-Leibler divergence between distributions p_1 and p_2
C_{10}^i, C_{01}^i	Cost of making false positive and false negative decisions for IDS i
C_{00}^i, C_{11}^i	Cost of making correct decisions for IDS i

Table 5.2: Simulation parameters

Parameter	Value	Meaning
τ_{SA}	0.5	decision threshold of the simple average model
τ_{WA}	0.5	decision threshold of the weighted average model
n	10	number of IDSs in the network
d	0.5	difficulty levels of intrusions and test messages
λ	0.9	forgetting factor
π_0, π_1	0.5	probability of no-attack and under-attack
C_{00}, C_{11}	0	cost of correct decisions

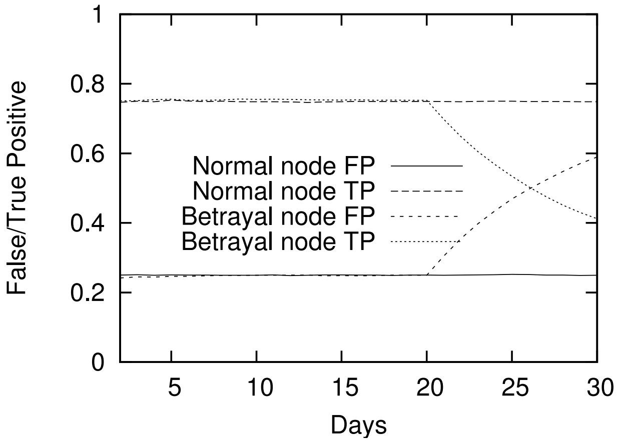


Figure 5.11: False positive and true positive of single IDS under Betrayal attack

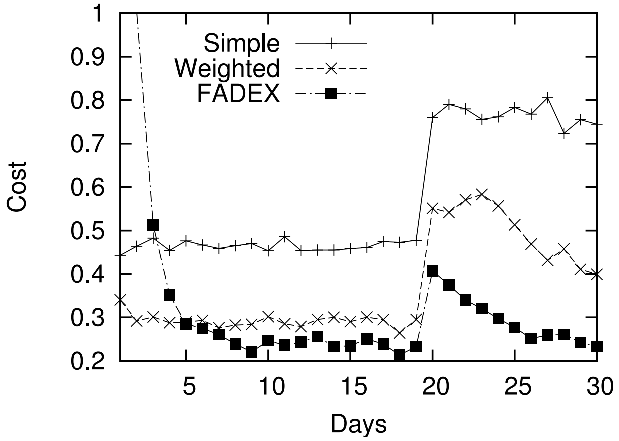


Figure 5.12: False decision cost under Betral attack

Chapter 6

Resource Management

6.1 Introduction

As discussed in the previous chapters, collaborative intrusion detection networks can improve the intrusion detection accuracy of participating IDSs. However, malicious insiders in an IDN may compromise the system by providing false information/feedback or overloading the system with spam. Also, "free riders" [71] can exploit the system by benefiting from others without contributing themselves. This can discourage IDN participants and eventually degrade the overall performance of the collaboration system. To solve the problems of malicious insiders and free-riders, a trust management is necessary to distinguish dishonest or malicious insiders, and an incentive-compatible resource allocation mechanism can help participating IDSs contribute helping resources to collaborators in a fair manner (i.e., more active contributors should receive more helping resources). The resource allocation mechanism itself should be robust against various insider attacks.

In this chapter, we propose a resource allocation mechanism, based on reciprocal incentive design and trust management, where the amount of resources that each IDS allocates to assist its neighbors is proportional to the trustworthiness and the amount of resources allocated by its neighbors to help this IDS. The motivation for reciprocal incentive design is to encourage participants to contribute more in collaboration so as to keep their IDS knowledge up-to-date. This exchange of knowledge is particularly important in order for IDSs to protect the system from new or zero-day attacks. We formulate an $N-$person (or peer) non-cooperative continuous-kernel game model to investigate incentive compatibility of the IDS collaboration system. In our design, each IDS finds an optimal resource allocation to maximize the aggregated satisfaction levels of its neighbors. We show that under certain controllable system conditions, there exists a unique Nash equilibrium. Our experimental results demonstrate that an iterative algorithm which we introduce converges geometrically to the Nash equilibrium, and that the amount of helping resource an IDS receives is proportional to its helpfulness to others. We also demonstrate security features of the system against free riders, dishonest insiders, and DoS attacks.

The main contributions of this work are: 1) A mechanism for optimal resource allocation for each peer to maximize its social welfare with a convex utility function; 2) An N-person non-cooperative game model and an iterative primal/dual algorithm to reach the Nash equilibrium; and 3) Incentive compatibility and robustness that is derived from the resource allocation scheme to tackle the "free riders", dishonest insiders, and DoS attacks.

6.1.1 Related Work

Many IDS collaboration systems have been proposed in literature, such as [119],[116], and [127]. They all assume IDSs cooperate honestly and unselfishly. The lack of trust infrastructure leaves the systems vulnerable to malicious peers.

A few trust-based collaboration systems (e.g. [97] and [56]) and distributed trust management models (e.g. [56], [45], and [57]) have been proposed for effective IDS collaboration. However, none of these proposed models studied incentives for IDS collaboration. Our previous work proposed a trust management system where IDSs exchange test messages to build trust among themselves. The feedback from collaborators is evaluated and a numerical trust value is assigned to reflect the level of truthfulness of collaborators. [56] uses a simple weighted average model to predict the trust value while [57] uses a Bayesian statistics model to estimate the trust value as well as the confidence level of the trust estimation.

A variety of game-theoretic approaches have been applied to network resource allocation in traditional routing networks and peer-to-peer (P2P) networks. In traditional routing networks, non-cooperative game models such as in [64] and [72] have been used in a dynamic resource allocation context; authors of these reference works have considered a network with a general topology where each source has a window-based end-to-end flow control. The available information for a user is the number of packets within the network not yet acknowledged. Each user aims to maximize his own throughput, with bounded delay, and hence faces a constrained optimization problem. The equilibrium obtained is decentralized since each user has only local information on his own unacknowledged packets. The focus has been on the maximal network performance with given resource instead of incentive mechanisms. In peer-to-peer networks, Ma et al. [78] have used a game-theoretical approach to achieve differentiated services allocation based on the peer's contribution to the community. Yan et al. [118] have proposed an optimal resource allocation scheme for file providers. A max-min optimization problem has been constructed to find the optimal solution which achieves fairness in the resource allocation. Both works rely on an independent central reputation system. Reciprocity has not been incorporated. Also the resilience and robustness of the system has not been their focus. Grothoff [62] has proposed a resource allocation economic model to deal with malicious nodes in peer-to-peer networks. It depends solely on the trust values of the peer nodes, and the resource allocation is priority-based on the trust value of the request sender. Grothoff's model can effectively prevent malicious nodes from overusing the network resource since their requests will be dropped due to their low trust. It is also reciprocal altruistic. However, this model may result in unfairness since nodes with the highest trust may take the entire resource. Our model differs from the above ones in that we have made use of the pair-wise nature of the network for designing scalable network algorithms, ensuring secure and resilient properties of the solution, and provide fairness and reciprocal incentive compatibility in resource allocation.

Recently, game-theoretical methods have been used for intrusion detection where in a two-player context, the attacker (intruder) is one player and the intrusion detection system (IDS) is the other player. In [131], and [128], non-cooperative game frameworks have been used to address different aspects of intrusion detection. In [117], Liu et al. use a Bayesian game approach for intrusion detection in ad-hoc networks; a two-person non-zero-sum incomplete information game is formulated to provide a framework for an IDS to minimize its loss based on its own belief. Our previous work [133] provides a game-theoretical model for IDSs to allocate collaboration resource to achieve the goal of fairness and incentive compatibility. This chapter extends our previous work by integrating a complete IDN framework and a robustness evaluation.

The rest of the chapter is organized as follows: In section 6.2, we describe our incentive-based resource allocation scheme for resource management in the IDN. In Section 6.3, we devise a primal/dual algorithm to compute the Nash equilibrium, and in Section 6.4 we evaluate the convergence and incentives of the resource allocation design. Finally, Section 6.5 concludes the chapter.

6.2 Resource Management and Incentive Design

In this section, we first mathematically model resource allocation in an IDN environment as individual optimization problems for its member peers. A game problem (GP) can then be introduced for each peer. We employ a Lagrangian approach to find the Nash equilibrium of the constrained game. Finally, we show that there exists a unique Nash equilibrium in the game and characterize the equilibrium solution in closed form.

6.2.1 Modeling of Resource Allocation

We consider a collaborative intrusion detection network (IDN) with N peers or nodes where all the nodes adopt the same resource allocation scheme. Each IDS user can distribute information to other IDS users in form of messages (in bytes). We denote the set of nodes by $\mathcal{N} = \{1, 2, \cdots, N\}$. The set of neighbor nodes of peer u is denoted by $\mathcal{N}_u$. The communications between IDSs become constrained when the network size is large and the number of collaborators $|\mathcal{N}_u|$ grows. Note that information in the network is symmetric. If u is a neighbor of v, then v is also a neighbor of u. We can represent the topology of an IDN by a graph $\mathcal{G} := (\mathcal{N}, \mathcal{E})$, where $\mathcal{E}$ is the set of (u, v) pairs in the network. We use r_{vu} to denote the units of resource that node u should allocate in order to serve v with full satisfaction. The minimum acceptable resource from u to v is m_{vu}. Note that r_{vu}, m_{vu} are chosen by node v and informed to node u during negotiation. Let $p_{uv} \in \mathbb{R}_+$ be the resource that u allocates to v, for every $u, v \in \mathcal{N}$. The parameter p_{uv} is a decision variable of peer u and is private information between peer u and peer v. To satisfy neighbor v, node u should allocate resource to v over the interval $[m_{vu}, r_{vu}]$.

In this model, we assume that each node has its own mechanism to evaluate the trust of its neighbors, and the trust values have already been determined. The trust evaluation mechanism has been discussed in Chapter 4. Let $T_v^u \in [0,1]$ be the trust value of peer v assessed by peer u, representing how much peer u trusts peer v. The allocated resource p_{uv} from peer u to v is closely related to the trust value T_v^u perceived by u.

Each peer maximizes its effort to help its neighbor nodes under its capacity constraint C_u, which is dependent on its own resource capacity such as bandwidth, CPU, memory, etc. Then, resource allocation should satisfy the following capacity constraint:

$$\sum_{v \in \mathcal{N}_u} p_{uv} \leq C_u, \text{ for all } u \in \mathcal{N}. \tag{6.1}$$

Our system introduces a utility function for each peer to model the satisfaction level of its neighbors. The utility function S_{uv} is given by

$$S_{uv} = \frac{\ln\left(\alpha \frac{p_{uv} - m_{vu}}{r_{vu} - m_{vu}} + 1\right)}{\ln(\alpha + 1)}, \tag{6.2}$$

where $\alpha \in (0, \infty)$ is a system parameter which controls the satisfaction curve and the term $\ln(\alpha + 1)$ in the denominator is the normalization factor. The function S_{uv} is a concave function on its domain under the condition $\alpha > 1$. The choice of logarithmic functions is motivated by the proportional fairness properties as in [84, 100] and has been used in the literature on power control, congestion control and rate control in communication networks [100, 134, 130].

Let $U_u : \mathbb{R}_+^{L(u)} \to \mathbb{R}_+$ be the peer u's aggregated altruistic utility, where $L(u) = \text{card}(\mathcal{N}_u)$, the cardinality of the set $\mathcal{N}_u$. Let the payoff function, U_u, for u be given by:

$$U_u = \sum_{v \in \mathcal{N}_u} w_{uv} S_{uv}, \quad w_{uv} = T_v^u p_{vu}, \tag{6.3}$$

where w_{uv} is the weight on peer v's satisfaction level S_{uv}, which is the product of peer v's trust value and amount of helping resource allocated to u. A higher weight is applied on peer v's satisfaction level S_{uv} if peer v is better trusted and more generous to provide help to u. In this system, each peer $u \in \mathcal{N}$ in the IDN intends to maximize U_u within its resource capacity. A general optimization problem (OP) can then be formulated as follows:

$$\begin{aligned} \max_{\{p_{uv}, v \in \mathcal{N}_u\}} \quad & \textstyle\sum_{v \in \mathcal{N}_u} w_{uv} S_{uv} \\ \text{s.t.} \quad & \textstyle\sum_{v \in \mathcal{N}_u} p_{uv} \leq C_u \\ & m_{vu} \leq p_{uv} \leq r_{vu}, \forall v \in \mathcal{N}_u, \end{aligned} \tag{6.4}$$

where S_{uv} and w_{uv} are given by (6.2) and (6.3), respectively. The upper and lower bounds on resources are imposed by the collaborators. The design of the utility function in OP is built upon the intuition behind how people form collaborations in social networks. With the freedom to choose and design collaborative schemes, we assume that all legitimate agents in the network start with an intent to form collaborations with each other.

Every peer in the network is faced with an optimization problem (OP) to solve. (OP) is a concave problem in which the objective function is a concave function in p_{uv} and the constraint set is an $L(u)$-dimensional simplex, where $L(u) = \text{card}(\mathcal{N}_u)$, the cardinality of the set $\mathcal{N}_u$. Under the assumptions that the size of the network is large and peers can only communicate locally within a distance d, we have N individual optimization problems in the form of (OP) for each node. Hence, we can introduce a corresponding game (GP) by the triplet $\langle \mathcal{N}, A_u, U_u \rangle$, where $\mathcal{N}$ is the set of players or peers, $A_u, u \in \mathcal{N}$, is the action set of each peer, and U_u is the payoff function of peer u, defined in (6.3). An action of a peer here is a decision on the resource allocated to a neighbor peer. The action set of each peer A_u is given by $A_u = A_u^1 \bigcap A_u^2$, where $A_u^1 = \{\mathbf{p}_u \in \mathbb{R}_+^{L(u)} \mid \sum_{v \in \mathcal{N}_u} p_{uv} \leq C_u\}$ and $A_u^2 = \{\mathbf{p}_u \in \mathbb{R}_+^{L(u)} \mid m_{vu} \leq p_{uv} \leq r_{vu}, v \in \mathcal{N}_u\}$. It is not difficult to prove that under the condition $C_u \geq \sum_{v \in \mathcal{N}_u} m_{vu}$, the action set is nonempty.

We note that the decision variable of each peer is a vector $\mathbf{p}_u$ and the action sets of players are not coupled. We thus can use Lagrangian relaxation to penalize the constraints to solve for the Nash equilibrium. Let $\mathcal{L}_u(\mathbf{p}_u, \sigma_u, \mu_u, \lambda_u)$ as follows denote the Lagrangian of peer u's optimization problem:

$$\begin{aligned} \mathcal{L}_u = & \sum_{v \in \mathcal{N}_u} T_v^u p_{vu} S_{uv} - \sum_{v \in \mathcal{N}_u} \mu_{uv}(p_{uv} - r_{vu}) \\ & + \sum_{v \in \mathcal{N}_u} \sigma_{uv}(p_{uv} - m_{vu}) - \lambda_u \left(\sum_{v \in \mathcal{N}_u} p_{uv} - C_u \right), \end{aligned} \tag{6.5}$$

where $\mu_{uv}, \sigma_{uv}, \lambda_u \in \mathbb{R}_+$ are the Lagrange multipliers. Using Lagrangian relaxation, we can transform the game problem to its relaxed counterpart (RGP), where the abbreviation "R" is short for "Relaxed". The triplet of RGP is given by $\langle \mathcal{N}, \bar{A}_u, \mathcal{L}_u \rangle$, where $\bar{A}_u$ is the action set described by the base constraint $p_{uv} \geq 0$, i.e., $\bar{A}_u = \{\mathbf{p}_u \mid p_{uv} \geq 0, v \in \mathcal{N}_u\}$; and the payoff function is replaced by the relaxed Lagrangian function $\mathcal{L}_u$. [1].

[1] In the definition of the relaxed game (RGP), we have chosen to relax simultaneously the two sets of

By formulating the collaborative problem as a game, we use a non-cooperative approach to model altruistic behavior among peers. The non-cooperativeness is appropriate here because there is no centralized control agent in the network, and communications between peers are local and symmetric. The aggregated utility comes from peers' general intention to help other peers. We assume that peers intend to be altruistic when they are introduced into the network. Free-riding peers are penalized via the weighting of the aggregation function. When one peer appears to refuse to help other peers, the other peers will correspondingly decline to assist in return, and as a result free-riding is avoided.

The framework described in this subsection can be potentially applied to a wide range of collaborative networks where reciprocal altruism is desirable. However, many distinct features of IDS networks have been incorporated into the design. Firstly, an attacker can compromise nodes in the network and then start to spread malware to degrade the level of protection provided by the collaborative network. The special structure of the utility function together with the trust values have been used in the model to mitigate malicious and dishonest behaviors of compromised nodes. Secondly, insider threats in IDS networks have been considered by imposing upper and lower bounds on p_{uv}, which can be used to prevent denial-of-service attacks from the insiders.

Remark 6.2.1 *The choice of using the word collaborative networks is to distinguish this approach from its cooperative counterpart. Cooperative networks often refer to a network of nodes that are able to act as a team and then split the team utility among the members. This will require global communications, coordination and bargaining. This appears to be unrealistic for IDN systems. In collaborative networks, nodes behave strategically not because they are selfish agents but because they are unable to coordinate or act as a team. Our work is essentially different from non-cooperative network formation problems, where all agents act selfishly to achieve their individual goals, which can be misaligned with each other. In our IDN design, the players have their goals aligned in a certain way to achieve efficient exchange of knowledge with each other. This is similar to classical strategic games such as Battle of the Sexes and Bach and Stravinsky game [98]. However, the goals become less aligned when agents have low trust values. This flexibility in the model essentially attributes to the reciprocal altruism.*

constraints, capacity constraint and range constraints. Instead, we could have relaxed only the capacity constraint. In that case, the action set $\bar{A}_u$ in the relaxed game would include a range constraint, i.e., $\bar{A}_u = \{\mathbf{p}_u \mid m_{vu} \leq p_{uv} \leq r_{vu}, v \in \mathcal{N}_u\}$.

6.2.2 Characterization of Nash Equilibrium

In this subsection, we solve the GP for its Nash equilibrium. Each peer u has a concave optimization problem as in (6.4). Applying the first-order KKT condition as in [31] and [34] to each peer's concave problem in OP, $\frac{\partial \mathcal{L}_u}{\partial p_{uv}} = 0, \forall v \in \mathcal{N}_u, u \in \mathcal{N}$, we find

$$\frac{\delta_{uv} T_v^u p_{vu}}{1 + \alpha'_{uv} p_{uv} - \alpha'_{uv} m_{vu}} = \xi_{uv}, \forall v \in \mathcal{N}_u, u \in \mathcal{N}, \tag{6.6}$$

where $\delta_{uv} = \frac{\alpha'_{uv}}{\ln(1+\alpha)}$; $\xi_{uv} = -\sigma_{uv} + \mu_{uv} + \lambda_u$, and $\alpha'_{uv} = \frac{\alpha}{r_{vu} - m_{vu}}$. In addition, from the feasibility condition, it is required that an optimal solution satisfies the base constraints in $\bar{A}_u$ and the complimentary slackness conditions for every $u \in \mathcal{N}$:

$$\lambda_u \left(\sum_{v \in \mathcal{N}_u} p_{uv} - C_u \right) = 0. \tag{6.7}$$

$$\sigma_{uv}(p_{uv} - m_{vu}) = 0, \forall v \in \mathcal{N}_u, \tag{6.8}$$

$$\mu_{uv}\,(p_{uv} - r_{vu}) = 0, \forall v \in \mathcal{N}_u\,. \tag{6.9}$$

The variable ξ_{uv} is composed of three Lagrange multipliers. If $\xi_{uv} \neq 0$, we can further simplify the first-order condition into

$$p_{uv} - \frac{T_v^u p_{vu}}{\xi_{uv} \ln(1+\alpha)} = \left(1 + \frac{1}{\alpha}\right) m_{vu} - \frac{1}{\alpha} r_{vu}. \tag{6.10}$$

Definition 6.2.2 *(Başar & Olsder, [29]) A Nash equilibrium $p^*_{uv}, u, v \in \mathcal{N}$ for the game (GP) is a point that satisfies $\mathcal{L}_u(\mathbf{p}^*_u, \mathbf{p}^*_{-u}) \geq \mathcal{L}_u(\mathbf{p}_u, \mathbf{p}^*_{-u})$, $\forall \mathbf{p}_u \in A_u, u \in \mathcal{N}$, and $p_{uv} = p_{vu} = 0$, for $v \in \mathcal{N}_u \backslash \mathcal{N}_u$ and $u \in \mathcal{N}$, where the vector $\mathbf{p}_{-u} = \{\mathbf{p}_i : i \neq u, i \in \mathcal{N}\}$ is comprised of decision vectors of other peers.*

Proposition 6.2.3 *The game (GP) admits a Nash equilibrium in pure strategies.*

Proof The action set A_u is a closed and bounded simplex and U_u is continuous in p_{uv} for all $u \in \mathcal{N}, v \in \mathcal{N}_u$ and concave in $\mathbf{p}_u$. By Theorem 4.4 in [29], there exists a Nash equilibrium to (GP).

With the existence of Nash equilibrium at hand, we can further investigate the solutions to the relaxed game by looking at a pair of nodes u and v. Node u has its decision vector $\mathbf{p}_u$ satisfying (6.10) and similarly, node v has its decision vector $\mathbf{p}_v$ satisfying (6.10) by interchanging indices u and v. Hence, we obtain a pair of equations involving p_{uv} and p_{vu} and they are described by

$$\begin{bmatrix} 1 & \frac{-T_v^u}{\xi_{uv}(\ln(1+\alpha))} \\ \frac{-T_u^v}{\xi_{vu}(\ln(1+\alpha))} & 1 \end{bmatrix} \begin{bmatrix} p_{uv} \\ p_{vu} \end{bmatrix} = \begin{bmatrix} \left(1+\frac{1}{\alpha}\right) m_{vu} - \frac{r_{vu}}{\alpha} \\ \left(1+\frac{1}{\alpha}\right) m_{uv} - \frac{r_{uv}}{\alpha} \end{bmatrix},$$

or in the matrix form, $\mathbf{M}_{uv}\mathbf{q}_{uv} = \mathbf{b}_{uv}$, where $\mathbf{q}_{uv} = [p_{uv}, p_{vu}]^T$, and $\mathbf{b}_{uv}$ is the right-hand side vector and $\mathbf{M}_{uv}$ is the incident matrix.

Definition 6.2.4 *(M-matrix, [30]) An N by N real matrix $\mathbf{A} = [A_{ij}]$ is called an M-matrix if it is of the form $\mathbf{A} = \theta\mathbf{I} - \mathbf{P}$, where $\mathbf{P}$ is entrywise nonnegative and θ is larger than the spectral radius of $\mathbf{P}$, i.e., $\theta > \rho(\mathbf{P})$. An M-matrix A has two key features:*

(F1) the sign patterns $a_{ii} > 0,\ i = 1, ..., N$, and $a_{ij} \leq 0,\ i \neq j$,

(F2) the eigenvalues of $\mathbf{A}$ have all positive real parts.

Theorem 6.2.1 *(Berman and Plemmons, [30]) If $\mathbf{A}$ is an M-matrix, then $\mathbf{A}^{-1} > 0$, i.e. all of its entries are positive.*

Using Theorem 6.2.1, we next state a result on uniqueness of Nash equilibrium for a sufficiently large system parameter α.

Theorem 6.2.2 *Suppose that only capacity constraints are active and $\alpha > \max\limits_{u,v}\{e^{\frac{T_v^u}{\xi_{uv}}}, \frac{r_{vu}}{m_{vu}}\} - 1$. Then, the game admits a unique Nash equilibrium. For each pair of peers u and v, the equilibrium is given by $\mathbf{q}^*_{uv} = \mathbf{M}^{-1}_{uv}\mathbf{b}_{uv}, \forall u, v \in \mathcal{N}$.*

Proof Under the condition that the capacity constraints are active, $\xi_{uv} = k_v\lambda_u > 0$, since the objective function is an increasing function. Firstly, we show that provided that $\alpha > e^{\frac{T_v^u}{\xi_{uv}}} - 1$, we have the inequality $1 > \frac{T_v^u}{\xi_{uv}\ln(1+\alpha)}$. For each pair of nodes u and v, matrix $\mathbf{M}_{uv}$ is an $M-$matrix in (6.10); hence, $\mathbf{M}_{uv}$ are strictly diagonally dominant and thus non-singular; and by Theorem 6.2.1, the entries of the inverse matrix $\mathbf{M}_{uv}^{-1}$ is strictly positive.

Secondly, provided that $\alpha > \frac{r_{uv}}{m_{vu}} - 1$, the vector $\mathbf{b}_{uv}$ is positive, i.e., $\left(1 + \frac{1}{\alpha}\right) m_{vu} > \frac{1}{\alpha} r_{uv}$. Thus, we arrive at a unique solution $\mathbf{q}_{uv}^{*}$, whose entries are all positive, residing in the base constraint action set $\bar{A}_u$ for all u. Since (6.10) holds for any interactive pair, the game admits a unique Nash equilibrium under conditions in Theorem 6.2.2.

Note that Theorem 6.2.2 provides a condition to choose system parameter α. Since the system can determine the value of α, the condition can be met easily.

Remark 6.2.5 *Under general conditions, to have $\xi_{uv} > 0$ requires multipliers μ_{uv}, λ_u, σ_{uv} to satisfy $\mu_{uv} + \lambda_u k_v > \sigma_{uv}$. Since payoff function U_u is increasing in p_{uv}, $\lambda_u > 0$ and only μ_{uv} and σ_{uv} can be zero. To ensure $\xi_{uv} > 0$, we can separate into three cases for general discussion: (1) when $\sigma_{uv} = 0,\ \mu_{uv} \neq 0,$ we require $\mu_{uv} + \lambda_u k_v > 0$; (2) when $\sigma_{uv} = 0,\ \mu_{uv} = 0,$ we require $\lambda_u k_v > 0$; (3) when $\sigma_{uv} \neq 0,\ \mu_{uv} = 0,$ we require $\lambda_u k_v > \sigma_{uv}$. With an assumption as in Theorem 6.2.2 that only capacity constraint is active, it simply leads to $\xi_{uv} > 0$ itself.*

6.2.3 Incentive Properties

We call a network design reciprocal incentive compatible when at the steady state, the helping resource p_{uv} from peer u to v increases as the helping resource p_{vu} from peer v to u also increases. In addition, it is also desirable to have p_{uv} to be proportional to the trust value of v, i.e., the more peer u trusts peer v, the more help u is willing to give. We can further study these properties of the solution obtained in Theorem 6.2.2.

Proposition 6.2.6 *Under the conditions of Theorem 6.2.2, the Nash equilibrium solution of the game (GP) is reciprocal incentive compatible, i.e.,*

1. *The helping resource p_{uv} from u to v increases with helping resource p_{vu} from v to u;*
2. *When the system parameter α increases, the marginal helping resource from u to v decreases for all u and v;*

3. *When peer u trusts v more, i.e., T_v^u increases, the marginal helping resource from u to v increases.*

Proof Using (6.6), we take the derivative with respect to p_{vu} and let $\partial p_{uv}/\partial p_{vu}$ denote the marginal helping rate from u to v.

Since $T_v^u > 0$, $\xi_{uv} > 0$, under the conditions in Theorem 6.2.2, we have $\partial p_{uv}/\partial p_{vu} > 0$, and thus p_{uv} is increasing with p_{vu} at Nash equilibrium. The incentive compatibility results follow.

In the following, we study the incentives of nodes that decide on the lower and upper bounds on desired reply rates. We assume that the lower bound on reply rates are uniformly determined by the system once they join the network, i.e., $m_{vu} = \bar{m}$ for all $v \in \mathcal{N}, u \in \mathcal{N}_v$.

Lemma 6.2.7 *Nodes do not have incentives to overstate their upper bound on the reply rate $r_{vu}, v \in \mathcal{N}, u \in \mathcal{N}_v$.*

Proof From (6.6), we can observe that $\frac{\partial p_{uv}}{\partial r_{vu}} = -1/\alpha < 0$. Hence, a higher level of request results in a lower value of p_{uv}.

Lemma 6.2.7 admits an intuitive interpretation. When a request level is high, it becomes harder for a node to satisfy it and the node will allocate resources to satisfy other ones with lower request levels first. Hence, a higher level of request will result in a lower reply rates.

In the following, we study the effect of understating the upper bound. We first introduce the notion of ϵ-resilience and then derive a condition for achieving it.

Definition 6.2.8 *The Nash equilibrium p^*_{uv} under truthful r^*_{vu} is ϵ-resilient if a deviation r_{vu} from r^*_{vu} results in an equilibrium p_{uv} such that $\|p^*_{uv} - p_{uv}\| \leq \epsilon \|r^*_{vu} - r_{vu}\|$ for all pairs of $(u, v) \in \mathcal{E}$.*

Proposition 6.2.9 *Suppose $\bar{m}$ is sufficiently small and only capacity constraints are active. The Nash equilibrium, if it exists, is ϵ-resilient if $\alpha \geq \frac{1}{\epsilon} \max_{(u,v)\in\mathcal{E}} \left| \frac{T_v^u p_{vu}}{\sum_{v \in \mathcal{N}_u} p_{vu} T_v^u} - 1 \right|$.*

Proof Let r^*_{vu} be the true upper bound, under which the reply rates are

$$\hat{p}^*_{uv} = \min\{\max\{\bar{m}, p^*_{uv}\}, r^*_{vu}\} \leq r^*_{vu},$$

where

$$p^*_{uv} = \left(1 + \frac{1}{\alpha}\right)\bar{m} - \frac{1}{\alpha} r^*_{vu} + \frac{T^u_v p_{vu}}{\xi^*_{uv} \ln(1+\alpha)}.$$

For any other $r_{vu} < r^*_{vu}$, the allocated resource is $\hat{p}_{uv} = \min\{\max\{\bar{m}, p_{uv}\}, r_{vu}\} \leq r_{uv} < r^*_{vu}$, where

$$p_{uv} = \left(1 + \frac{1}{\alpha}\right)\bar{m} - \frac{1}{\alpha} r_{vu} + \frac{T^u_v p_{vu}}{\xi_{uv} \ln(1+\alpha)}.$$

Suppose that $\bar{m}$ is sufficiently small. Due to the assumption that only capacity constraints are active, we only need to study the case where $p_{uv} \leq r_{vu}$. Then, from Lemma 6.2.7, we obtain $p_{uv} > p^*_{uv}$ since $r_{vu} < r^*_{vu}$, and hence $p^*_{uv} < p_{uv} \leq r_{vu} < r^*_{vu}$. Therefore, $\|\hat{p}_{uv} - \hat{p}^*_{uv}\| = \|p_{uv} - p^*_{uv}\|$ and we have

$$\|p_{uv} - p^*_{uv}\| \leq \left\| -\frac{1}{\alpha}(r_{vu} - r^*_{vu}) + \frac{T^u_v p_{vu}}{\ln(1+\alpha)} \left[\frac{1}{\xi_{uv}} - \frac{1}{\xi^*_{uv}}\right] \right\|.$$

Under the relaxed conditions, we can use the closed form expression of Lagrangian multiplier (6.16), which is derived later in Section 6.3, to obtain $\frac{1}{\xi_{uv}} - \frac{1}{\xi^*_{uv}} = \frac{1}{\lambda_u} - \frac{1}{\lambda^*_u} = \frac{\ln(1+\alpha)}{\alpha P_T}(r_{vu} - r^*_{vu})$. Hence combining with the result above, we arrive at

$$\|p_{uv} - p^*_{uv}\| \leq \frac{1}{\alpha} \left\| \frac{T^u_v p_{vu}}{P_T} - 1 \right\| \|r_{vu} - r^*_{vu}\|.$$

Therefore, to ensure ϵ-resiliency, we need $\frac{\|p_{uv} - p^*_{uv}\|}{\|r_{vu} - r^*_{vu}\|} \leq \frac{1}{\alpha} \left\| \frac{T^u_v p_{vu}}{P_T} - 1 \right\| \leq \epsilon$, which leads to the result.

6.3 Primal / Dual Iterative Algorithm

In this section, we introduce a dynamic algorithm to compute the unique Nash equilibrium. Let $p_{uv}(t)$ be the resource from peer u to v at step t. Consider the algorithm:

$$\begin{cases} p_{uv}(t+1) = s_{uv} + t_{uv} p_{vu}(t) \\ p_{vu}(t+1) = s_{vu} + t_{vu} p_{uv}(t) \end{cases}, \tag{6.11}$$

where $s_{uv} = \left(1+\frac{1}{\alpha}\right) m_{vu} - \frac{1}{\alpha} r_{vu},\ t_{uv} = \frac{T_v^u}{\xi_{uv}(\ln(1+\alpha))}$, and $s_{vu},\ t_{vu}$ are defined similarly by interchanging indices u and v, with initial conditions $p_{uv}(0) = \min\left\{\frac{C_u}{\mathcal{N}_u}, r_{uv}.\right\}, \forall u, v \in \mathcal{N}$.

Proposition 6.3.1 *Suppose that capacity constraints are active, and r_{vu} and m_{uv} are chosen such that the associated constraints become inactive constraints, i.e., $\sigma_{uv} = 0, \mu_{uv} = 0$ in (6.8) and (6.9). Given a Lagrange multiplier $\lambda_u^* \neq 0$ and provided that $\alpha > e^{\frac{T_v^u}{\lambda_u}} - 1$, algorithm (6.11) converges to the unique Nash equilibrium in Theorem 6.2.2 at dual optimal λ_u^*.*

The algorithm described in (6.11) depends on the Lagrange multiplier λ_u. We can exploit duality to devise an iterative algorithm for the Lagrange multiplier. Let $D_u(\lambda_u)$ be the dual functional given by $D_u(\lambda_u) = \max_{\mathbf{p}_u} \mathcal{L}_u(\mathbf{p}_u, \lambda_u)$. The dual function $D_u(\lambda_u)$ is a convex function and a dual optimal λ_u^* solves the dual optimization problem (DOP)[2]

$$\min_{\lambda_u > 0} D_u(\lambda_u). \tag{6.12}$$

Using the solution from Theorem 6.2.2, we can obtain $D_u(\lambda_u)$ as follows.

$$D_u = \lambda_u \left(C_u + \frac{K_R}{\alpha} + \left(1 + \frac{1}{\alpha}\right) K_M \right) + \frac{\overline{P_T} - P_T}{\ln(\alpha + 1)},$$

and its first-order derivative as follows:

$$D_u' = C_u - \frac{\sum_{v \in \mathcal{N}_u} p_{vu} T_v^u}{\lambda_u \ln(1+\alpha)} + \frac{1}{\alpha} \sum_{v \in \mathcal{N}_u} r_{vu} - \frac{\alpha + 1}{\alpha} \sum_{v \in \mathcal{N}_u} m_{vu},$$

where $P_T = \sum_{v \in \mathcal{N}_u} p_{vu} T_v^u$ is the sum of the weights; $K_M = \sum_{v \in \mathcal{N}_u} m_{vu}$; $K_R = \sum_{v \in \mathcal{N}_u} r_{vu}$. K_M and K_R can be interpreted as the total request weighted by marginal costs; and

$$\overline{P_T} = \sum_{v \in \mathcal{N}_u} p_{vu} T_v^u \ln \left(\frac{\alpha}{\ln(\alpha + 1)} \frac{p_{vu} T_v^u}{\lambda_u (r_{vu} - m_{vu})} \right). \tag{6.13}$$

[2] Peer u's dual function is expressed in terms of λ_u and $\mathbf{p}_{-u}$, and the decision variable for peer u changes from a multi-dimensional vector $\mathbf{p}_u$ to a scalar variable λ_u. Using the dual function, we can reduce the dimension of the game and turn a constrained game into an unconstrained one.

The gradient of the dual function is dependent on the local capacity of node u and the information sent by the neighbor node v of peer u such as the helping resource p_{vu}, and the maximum (minimum) requested resources r_{vu} (m_{vu}) from v. All the information is available to peer u to calculate the gradient locally at each λ_u.

By taking the second-order derivative of the dual function, we obtain

$$D''_u(\lambda_u) = \frac{\sum_{v\in\mathcal{N}_u} p_{vu}T^u_v}{\lambda_u^2 \ln(1+\alpha)}. \tag{6.14}$$

The dual function in (6.12) is not only a convex function but also a strong convex function, whose Hessian is bounded uniformly as in $L_1 \leq \nabla^2 D_u(\lambda_u)$, for some L_1 [34]. In addition, provided that the sum of weights w_{uv} is bounded from above, i.e.,

$$\sum_{v\in\mathcal{N}_u} p_{vu}T^u_v \leq M, \tag{6.15}$$

for some $M \in \mathbb{R}_{++}$, then $\nabla^2 D_u(\lambda_u) \leq L_2$, for some constant L_2.

Proposition 6.3.2 *Suppose that the sum of weights is bounded as in (6.15). The dual function D_u is strongly convex and its Hessian is bounded from above and below uniformly.*

Proof Firstly, λ_u is bounded from above by some constant $\bar{\lambda}_u$ since the dual problem is feasible. Thus, $\epsilon_1 \leq \lambda_u \leq \bar{\lambda}_u, \epsilon_1 > 0$. In addition, $\sum_{v\in\mathcal{N}_u} w_{uv} \neq 0$; otherwise, the primal problem is trivial because $w_{uv} = 0$, for all v. Therefore, $\epsilon_2 \leq \sum_{v\in\mathcal{N}_u} w_{uv} \leq M, \epsilon_2 > 0$. Hence, the statement is true.

Strong duality ensures a unique optimal solution. The unique dual optimal λ^*_u can be found explicitly by applying the unconstrained optimality condition, i.e., $D'_u(\lambda_u) = 0$. As a result, we obtain

$$\lambda^*_u = \frac{P_T}{\left(C_u - K_M + \frac{1}{\alpha}(K_R - K_M)\right)\ln(1+\alpha)}. \tag{6.16}$$

To find the dual optimal, we can also devise a dynamic algorithm that can be used in conjunction with Algorithm (6.11). An iterative algorithm based on gradient methods to find λ_u is given by

$$\lambda_u(t+1) = \lambda_u(t) - \beta_u D'_u(\lambda_u(t)), \forall u \in \mathcal{N}, \tag{6.17}$$

where $\beta_u \in (0, 1)$ is the step size. The gradient algorithm in (6.17) is distributed over the network. Each peer needs to collect openly accessible information from its neighboring peers to evaluate K_M, K_R and P_T. With the property of strong convexity, we can show in the following the fast convergence of the algorithm to (6.16).

Proposition 6.3.3 *Suppose that $D'_u(\lambda_u)$ is Lipschitz with Lipschitz constant L_3 and $D_u(\lambda_u)$ is strongly convex with $D''_u(\lambda_u) \geq L_1$. The dual algorithm (6.17) converges geometrically to dual optimal λ^*_u in (6.16) with step size $\beta_u < \frac{\min(2,L_1)}{L_3}$.*

Proof We can use the technique in [34] to prove the proposition. Using the property of strong convexity and Lipschitz property, we obtain

$$\begin{aligned}
&\|\lambda_u(t+1) - \lambda^*_u\|^2 \\
&\qquad = \|\lambda_u(t) - \lambda^*_u\|^2 - 2\beta_u D'_u(\lambda_u(t))(\lambda_u(t) - \lambda^*_u) \\
&\qquad\quad + \beta_u^2 \|D'_u(\lambda_u(t))\|^2 \\
&\qquad \leq \|\lambda_u(t) - \lambda^*_u\|^2 - 2\beta_u (D_u(\lambda_u(t)) - D_u(\lambda^*_u)) \\
&\qquad\quad + \beta_u^2 L_3 \|\lambda_u(t) - \lambda^*_u\|^2 \\
&\qquad \leq \|\lambda_u(t) - \lambda^*_u\|^2 - \beta_u L_1 \|\lambda_u(t) - \lambda^*_u\|^2 \\
&\qquad\quad + \beta_u^2 L_3 \|\lambda_u(t) - \lambda^*_u\|^2 \\
&\qquad = (1 - \beta_u L_1 + \beta_u^2 L_3) \|\lambda_u(t) - \lambda^*_u\|^2.
\end{aligned}$$

Hence, when $\beta_u < \frac{\min(2,L_1)}{L_3}$, we have a contraction. In addition, $\|\lambda_u(t+1) - \lambda^*_u\|^2 \leq (1 - \beta_u L_1 + \beta_u^2 L_3)^{t+1} \|\lambda_u(0) - \lambda^*_u\|^2$. Hence, the convergence rate is geometric.

Note that the condition of strong convexity can be easily satisfied from (6.14) if we eliminate trivial cases that all trust values of neighbors or p_{vu} are zeros.

6.4 Experiments and Evaluation

In this section, we perform numerical experiments and evaluate the trust and resource management capabilities of the resource allocation system as described in Sections 6.2 and 6.3. We follow two different approaches to evaluate the Nash equilibrium of the

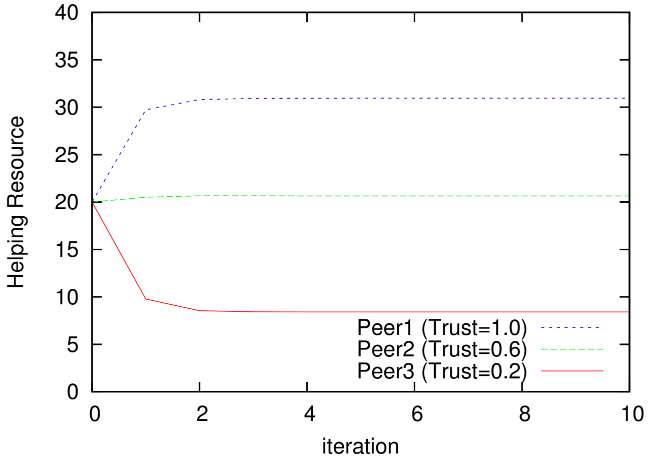


Figure 6.1: Helping Resources v.s. Time - First Approach

collaborative system. In the first experiment, we implement the dynamic algorithm in Section 6.3 to find the Nash equilibrium. We show that the algorithm yields the Nash equilibrium of the game at the steady state and the system is incentive compatible under the equilibrium. In the second experiment, we use a stochastic discrete-event based simulation to model an IDS network. In the simulation, peers estimate the resources received from the other peers and adjust their allocations of resources to the others accordingly. We are interested in finding the Nash equilibrium and verifying the incentives in the collaborative system at the equilibrium.

6.4.1 Nash Equilibrium Computation

In this section, we implement the dynamic algorithm described in Section 6.3 to calculate the Nash equilibrium centrally. We simulate a three-node network with initial trust values $0.2, 0.6, 1.0$, respectively. For the ease of demonstration, we assume that the trust between peer nodes is homogeneous. i.e., the trust value of node i is the same to all other nodes. We set the minimum demand of resource to 1 unit and the maximum to 20 units for all nodes. Every node has an equal capacity of 20 units and the system parameter $\alpha = 100$. We find that, if all peers have the same trust values, then the resource is fairly and evenly distributed among all peers. When the trust values are different, peers with higher trust

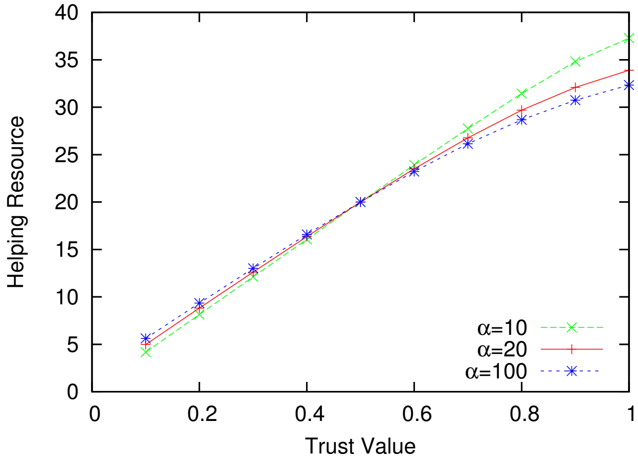


Figure 6.2: Helping Resource Received Varies with Trust Value - First Approach

values receive more resources. Fig. 6.1 shows that the resources received by three peers with different trust values converge fast within two or three iterations. A peer with higher trust value receives more help than a peer with lower trust value.

Fixing the resource capacity of all peers to 20 units and the trust values of two of the nodes to 0.5, we vary the trust value of the third peer from 0.1 to 1.0. In Fig. 6.2, we observe that the resource received by the third peer increases with its trust value under different α values. We also see that all curves cross at trust value 0.5 and resource 20 units. This is because all peers should receive equal amount of resources when they are identically configured, regardless of the α value we choose. By fixing the trust values of all nodes to 1.0 and varying the resource capacity of the third peer from 3 to 30, we observe in Fig. 6.3 that the amount of resources a peer receives is roughly linearly proportional to the resources it provides to the others. Similarly, all curves intersect at capacity 20 and resource 20. These results further confirm our theoretical analysis in Section 6.2. Figs. 6.2 and 6.3 also reveal that a larger α value leads to a lower marginal helping resource. A smaller α value provides stronger incentive to the participants.

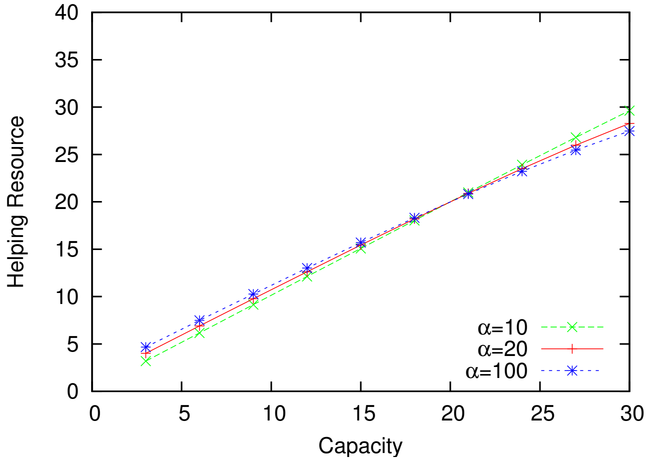


Figure 6.3: Helping Resource Received Varies with Resource Contribution - First Approach

6.4.2 Nash Equilibrium using Distributed Computation

In this experiment, we use a stochastic discrete-event based simulation to model the IDN. Discrete-event simulation is commonly used to aid strategic decision making since it has the capability of emulating complex real world problems. It concerns the modeling of a system as it evolves over time by representing the changes as separate events. It bridges over our model and a real-life IDS network. In this simulation, each node collaborates with others by sending out requests and waits for their responses. At the beginning of each day, nodes send resource upper-bound/lower-bound to all their neighbors and wait for the resource quota from them. The resource quota allocation is determined through optimizing (6.4). The consultation requests are generated randomly following a Poisson process with an average arrival rate equal to the resource quota they receive. Upon the arrival of a request at its destination queue, it will be replied by the corresponding peer on a first-come-first-serve basis. Each peer estimates the resource it receives from other peers by calculating the average number of consultation requests answered by each peer. In this experiment, all peers initialize with an unbiased allocation, and then apply the resource allocation scheme.

For the purpose of comparing with the numerical experiment, we use the same experiment configuration as in Section 6.4.1, i.e., we simulate a network of 3 nodes; we set the minimum resource requirement to 1 request/day and the maximum to 20 requests/day for

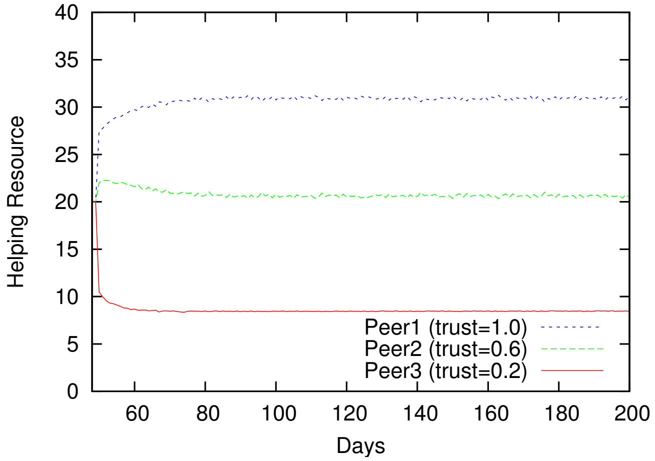


Figure 6.4: Helping Resources v.s. Time - Second Approach

all peers; each peer has a capacity of 20 requests; we set $\alpha = 100$ and the trust values of nodes to be 0.2, 0.6, and 1.0, respectively.

Fig. 6.4 illustrates the received resources for all three nodes with respect to time. We note that the helping resource converges to the Nash equilibrium at steady state, and nodes with higher trust values obtain more resource. This confirms that our resource allocation scheme provides incentives in the collaborative network.

By fixing the resource capacity of all peers to 20, the trust values of two of the peers to 0.5, and varying the trust values of the third peer from 0.1 to 1.0, we obtain in Fig. 6.5 that the received resource of the third peer increases with its trust value under different α values. Fixing the resource capacity of the first two peers to 20 requests/day and trust values to 1.0 for all peers, we vary the capacity of the third peer from 3 requests/day to 30 requests/day and observe that the resource received by the third node also increases with its resource capacity under different α values, as shown in Fig. 6.6. The simulation results are consistent with the theoretical results obtained in Section 6.2 and the ones in Section 6.4.1.

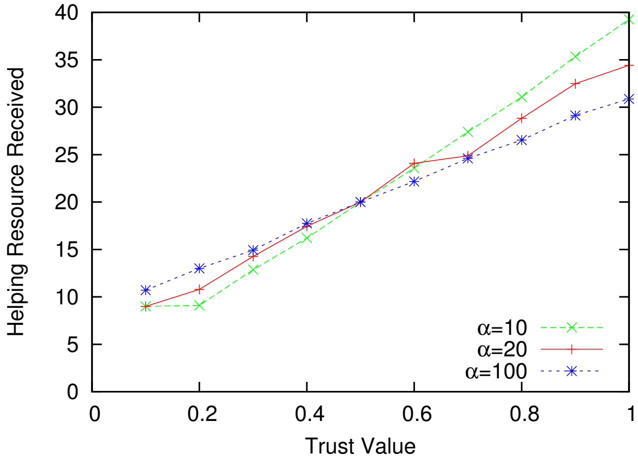


Figure 6.5: Helping Resource Received Varies with Trust Value - Second Approach

6.4.3 Robustness Evaluation

Robustness is a required and important feature for the design of an IDN. In this subsection, we discuss a few common insider threats against the incentive-based resource allocation mechanism, and we show how our design is robust to these attacks. Note that all participants in the IDN have to abide by the protocols with a given flexibility in parameters tuning. However, due to the reciprocity of the mechanism, IDSs with selfish or dishonest behaviors will be penalized and eventually removed from the network. This execution process is an integrated part of the IDN.

Free Riding

Free riders are nodes that enjoy resources from others while not contributing themselves [48, 61]. A free rider in the IDN may collaborate with a large number of IDSs, aiming at receiving a good amount of accumulated resources $\bar{m}$ from the large number of collaborators. However, our IDN design is not beneficial to free riders. First, the amount of help that a node receives is proportional to the resources it allocates to others. Second, the larger the number of collaborators a node has, the more demanding it is for the node to maintain the collaboration since each collaborator needs minimum resource $\bar{m}$ to be satisfied. Therefore, a node that does not contribute to the collaboration will end up receiving bare

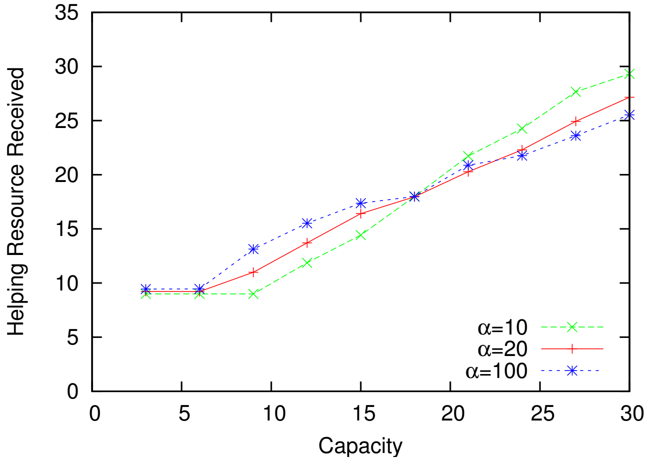


Figure 6.6: Helping Resource Received Varies with Resource Contribution - Second Approach

minimum helping resources from others. We simulate a scenario where a free rider with initial trust value 1.0 switches to a free riding mode at day 200 (Fig. 6.8). We notice that the amount of helping resources received by the free rider drops quickly and converges to a low level. This is because the collaborators of the free rider can notice the drop of contributed resources from the free rider and adjust their resource allocation according to (6.4). The result corroborates that free riding is not practical in the IDN with such a resource allocation design.

Denial-of-Service (DoS) Attacks

DoS attacks happen when malicious nodes send a large amount of information to overload the victim [85]. In our IDN, the amount of information exchanged between participant nodes is negotiated beforehand. A quota is calculated and sent to all nodes. If a node sends more data than the given quota, then it is considered malicious, and hence will be removed from the collaboration network.

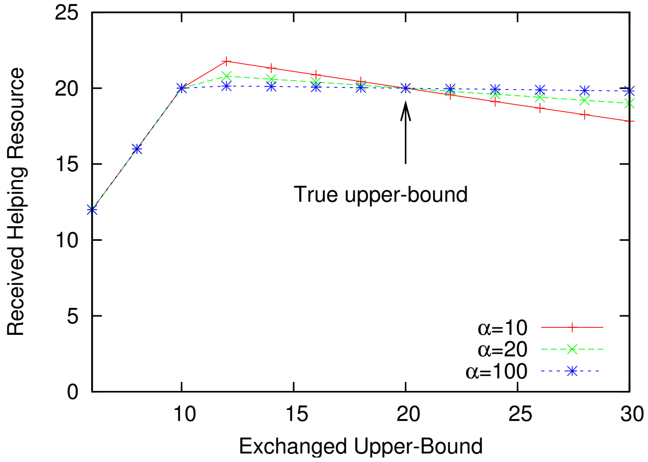


Figure 6.7: Resource received vs. exchanged upper-bound.

Dishonest Insiders

In the IDN, dishonest nodes can report false information to gain advantages. For example, a dishonest node can misinform about its upper-bound and lower-bound requests for gaining more resources from its collaborators. We imposed a maximum lower-bound $\bar{m}$ for all nodes. In addition, experimental results in Fig. 6.7 show that claiming a higher upper-bound than the true value lowers received resource, while claiming a lower upper-bound may lead to a bounded gain that is controllable by system parameter α. A lower upper-bound will not lead to full satisfaction of the node when resource constraints are inactive.

6.4.4 Large-Scale Simulation

Previous experiments are based on a small-scale network. In this subsection, we design numerical experiments to study the resource allocation in a large-scale intrusion detection network. We set up a network of 100 nodes, which are randomly scattered in a 100×100 square. Each node shares its resources with the other nodes in the vicinity at a distance of 5. The trust values are generated according to a uniform distribution from 0 to 1.0. The lower bound and the upper bound on the requests are 1 and 20, respectively, for each node. We separate nodes into two groups: one group with a capacity of 20 units and the other with 40. In Fig. 6.9, we can see that, in both groups, nodes with higher trust values

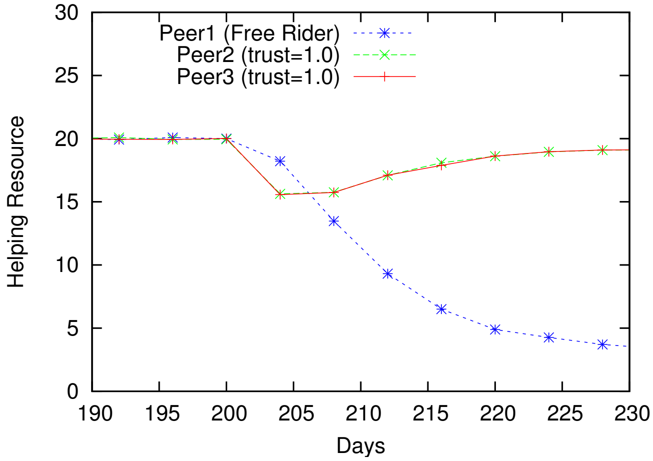


Figure 6.8: Resource received after free riding attack

tend to receive more assistance. The response to trust value appears to be more prominent for the group with capacity of 40 units. It can be explained by the fact that when the resource capacity is low, most of the resource is used to satisfy the lower bound of all the neighbors and little is left to allocate based on incentives. In the second experiment, we fix trust values of all nodes to 1.0 and randomly choose the resource capacity of each node between 0 and 30. Fig. 6.10 shows the resource received by nodes with different resource capacities. We note that, on the average, nodes with higher resource capacities receive more resources. This confirms the incentives under a large collaboration group.

6.5 Conclusion

In this chapter, we have proposed incentive-based resource allocation mechanism based on trust management in the context of an IDN. By formulating an associated continuous-kernel non-cooperative game, we have shown that a Nash equilibrium exists and is unique under certain system conditions. We have also shown that the unique Nash equilibrium possesses features that allow peers to communicate in a conducive environment in which peers endeavor to contribute knowledge and resource to assist neighbor nodes. Any selfish or free-riding behavior will receive a tit-for-tat response from the neighbors as a consequence. The dynamic algorithm proposed in the chapter is used to compute the Nash

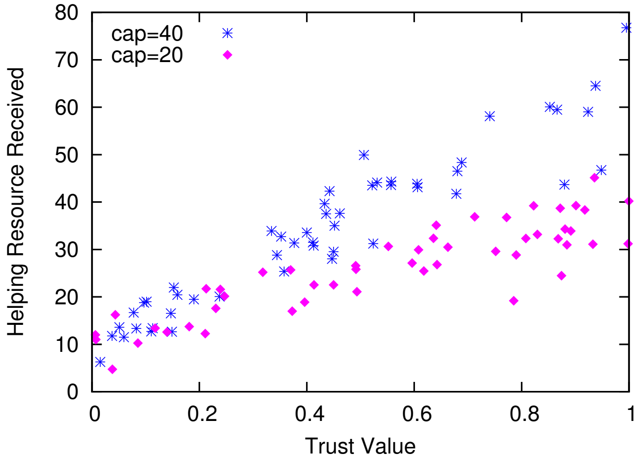


Figure 6.9: Resource received for peers with different trust values

equilibrium. Experimental results showed that the algorithm converges to the Nash equilibrium at a geometric rate, further corroborating the theoretical results. We have also discussed the resistance of our IDN design to common insider attacks, such as free-riding, dishonest insiders, and DoS attacks. As a future work, one can study other potential attacks to the IDN system, for example, the collusion attacks.

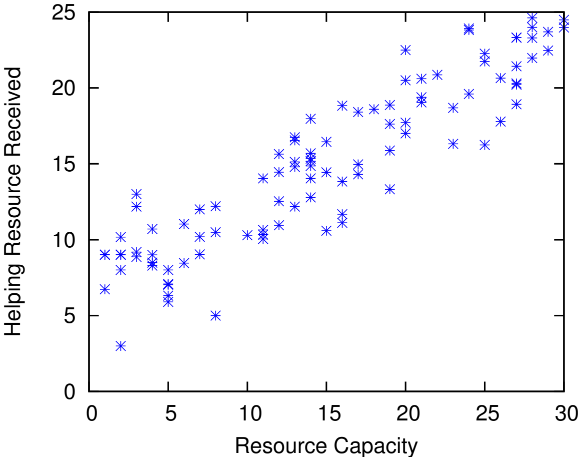


Figure 6.10: Resource received for peers with different resource capacities

Chapter 7

Collaborators Selection and Management

7.1 Introduction

As discussed in precious chapters, malicious insiders in an IDN may send false information to mislead other IDSs into making incorrect intrusion decisions. This may render the collaboration system ineffective. Furthermore, IDSs in the collaboration network may have different intrusion detection expertise levels and capabilities. An effective trust management model should be capable of distinguishing honest participants from malicious ones, and low-expertise IDSs from high-expertise IDSs. Chapter 4 describes a Bayesian learning model for IDSs to evaluate the trustworthiness of their collaborators. However, a collaboration relationship is a mutual agreement between both participants, and it should only occur when both parties agree to collaborate with each other. As we discussed in Chapter 5, the expected cost of false decisions decreases when receiving feedback from more collaborators. However, it takes more computing resources to maintain a collaboration relationship; for example, sending test messages and responding to consultation requests from other collaborators requires CPU/memory and bandwidth to proceed. The extra cost of recruiting a new collaborator may exceed the benefit from that collaborator. How IDSs select collaborators to achieve optimal cost efficiency is an important problem to solve for an IDN. We define an IDN *acquaintance management* as the process of identifying, selecting, and maintaining collaborators for each IDS. An effective acquaintance management model is crucial to the design of an IDN.

The purpose of the work in this chapter is to seek an effective acquaintance management mechanism with which IDSs can selectively recruit collaborators that can bring maximal benefit, taking into account both the false decision cost and maintenance cost. We propose a Bayesian learning technique that helps each IDS identify expert nodes and novice nodes based on past experience with them, specifically, the false positive (FP) rate and false negative (FN) rate of each collaborator. Dishonest collaborators are identified and removed from the collaborator list. We define *feedback aggregation* in the IDN as a decision-making method as to whether or not to raise an alarm based on the collected opinions (feedback) from collaborator IDSs. We propose a Bayesian decision model for feedback aggregation. Bayes theory is used to estimate the conditional probability of intrusions based on feedback from collaborators. A cost function is modeled to include the false positive decision cost and false negative decision cost. A decision as to whether to raise an alarm or not is made in order to achieve the minimum cost of false decisions.

For collaborator selection, an IDS may add all honest IDSs to its collaborator list to achieve maximized detection accuracy. However, including a large list of collaborators may result in a high maintenance cost. We define an *acquaintance selection* as the process of finding the optimal list of collaborators to minimize false decision and maintenance costs. Existing approaches for acquaintance management often set a fixed number of collaborators [121] or a fixed accuracy threshold to filter out less honest or low-expertise collaborators [122, 57, 53]. These static approaches lack flexibility, and the fixed acquaintance length or accuracy threshold may not be optimal when the context changes (e.g., some nodes leave the network and some new nodes join the network). Our proposed acquaintance management algorithm can dynamically select collaborators in any context setting to obtain high efficiency at minimum cost.

For collaborator maintenance, the IDSs in our system periodically update their collaborator lists to guarantee an optimal cost. A probation list is used to explore and learn the quality of new potential collaborators. New collaborators stay in the probation list for a certain period before their feedback is considered for intrusion decision.

We evaluate our system using a simulated collaboration network using a Java-based discrete-event simulation framework. The results show that the proposed Bayesian decision model outperforms the threshold-based model [89], which only counts the number of intrusion reports, in terms of false decision cost. The results also show that our dynamic acquaintance management algorithm outperforms the static approaches of setting a fixed acquaintance length or accuracy threshold. Finally, our approach also achieves several desired properties, such as efficiency, stability, robustness, and incentive-compatibility.

Major contributions of our work are summarized as follows:

1. An acquaintance selection algorithm is devised to optimally select collaborators, which leads to minimal overall cost, including false decision cost and maintenance cost;

2. A dynamic acquaintance management algorithm is proposed to integrate the concept of probation period and consensus negotiation;

The rest of the chapter is organized as follows. In Section 7.2, we discuss some related work; Section 7.3 describes the formalization of our IDS learning model and feedback aggregation. Acquaintance selection and management algorithms are presented in Section 7.4. We then present evaluation results demonstrating the effectiveness of our acquaintance management and its desired properties in Section 7.5. We conclude this chapter in Section 7.6.

7.2 Related Work

Various approaches have been proposed to evaluate IDSs, and all have used a single trust value to measure whether an IDS will provide good feedback about intrusions based on past experience with that IDS. For example, Duma et al. [45] introduced a trust-aware collaboration engine for correlating intrusion alerts. Their trust management scheme uses each peer's past experience to predict others' trustworthiness. Our previous work [57, 53] uses Dirichlet distributions to model peer trust, but it does not investigate conditional detection accuracy such as false positives and false negatives. In this work, we use both false positive and true positive rates to represent the detection accuracy of an IDS, based on a Bayesian learning approach. The methods for aggregating feedback provided by Duma et al. [45] and our previous work [57, 53] are also simplistic. They both use a weighted average approach to aggregate feedback. Another broadly accepted decision model in IDNs is the threshold-based, which is used in AVCloud [89]. In this model, when the total number of collaborators raising alarms exceeds a fixed threshold, an alarm will be raised. In this chapter, we apply the well established Bayes' theorem for feedback aggregation which achieves better performance. Our previous work [57, 53] focuses on the trust evaluation with a simple threshold-based acquaintance selection. This work focuses on the optimal collaboration decision and optimal acquaintance selection.

Most previous approaches set a fixed length of the acquaintance list, such as in [121]. Others use a trust threshold to filter out less honest acquaintances [122, 57]. The advantage of the threshold based decision is its simplicity and ease of implementation. However, it is only effective in a static environment where collaborators do not change, such as that presented in [89]. In a dynamic environment, nodes join and leave the network and the acquaintance list changes over time. Therefore, finding an optimal threshold is a difficult task. Our Bayesian decision model is efficient and flexible. It can be used in both static and dynamic collaboration environments. Equipped with this Bayesian decision model, our acquaintance selection algorithm can find the smallest set of best acquaintances that can maximize the accuracy of intrusion detection. Based on this acquaintance selection algorithm, our acquaintance management method uses a probation list to explore potential candidates for acquaintances and balances the cost of exploration and the speed of updating the acquaintance list.

7.3 IDS Detection Accuracy Evaluation and Feedback Aggregation

To select collaborators, an IDS should first learn the qualification of all candidate IDSs. In this section, we first introduce a Bayesian learning model to evaluate the detection accuracy of the candidates. A Bayesian decision model is then used to optimally aggregate feedback from acquaintances.

7.3.1 Detection Accuracy for a Single IDS

To better capture the qualification of an IDS, we use both *false positive* (FP) and *true positive* (TP) rates to represent the detection accuracy of an IDS. Let $\mathcal{A}$ denote the set of

Table 7.1: Summary of Notations

Symbol	Meaning
$X \in \{0,1\}$	Random variable denoting whether there is an attack or not
$Y \in \{0,1\}$	Random variable of positive or negative diagnose from an IDS
$\mathbf{y}$	A feedback instance vector from acquaintances
$\mathbf{Y}$	Feedback vector from acquaintances
$\mathcal{C}$	Set of acquaintance candidates
$\mathcal{A}$	Set of Acquaintances
l	The acquaintance list length
δ	The decision of raising alarm or not
$R(.)$	The risk cost of false alarms and miss intrusions
$M(.)$	The maintenance cost of acquaintances
C_{fp}, C_{fn}	Unit cost of false alarm and miss intrusion
C_a	Unit cost of maintaining each acquaintance
π_0, π_1	Priory probability of no-intrusion and with-intrusion
T_i, F_i	True positive rate and false positive rate of IDS i
λ	Forgetting factor of the past experience

acquaintances and random variables F_k and T_k denote the FP and TP rates of acquaintance $k \in \mathcal{A}$ respectively. FP is the probability that the IDS gives a positive diagnosis (under-attack) under the condition of no-attack, and TP is the probability that the IDS gives a correct positive diagnosis under the condition of under-attack. Let random variable $X \in \{0,1\}$ represent the random event on whether there is an attack or not, and let random variable $Y \in \{0,1\}$ denote whether the IDS makes a positive diagnosis or not. Then FP and TP can be written as $\mathbb{P}[Y = 1|X = 0]$ and $\mathbb{P}[Y = 1|X = 1]$, respectively. The list of notations is summarized in Table 7.1.

Let $\mathcal{F}_k$ and $\mathcal{T}_k$ be the probability density functions of F_k and T_k whose support is $[0,1]$. We use the notation $Z_0 : Y_k = 1|X = 0$ and $Z_1 : Y_k = 1|X = 1$ to represent the conditional variables that acquaintance k gives positive decision under the conditions where there is no attack and there is an attack respectively. They can be seen as two independent random variables satisfying Bernoulli distribution with successful rates F_k and T_k, respectively. The past experience with acquaintance k can be seen as the samples from the Bernoulli distributions. According to the Bayesian probability theory [59], the posterior distribution of $\mathcal{F}_k$ and $\mathcal{T}_k$ given a set of observed samples can be represented using a Beta function, written as follows:

$$\mathcal{F}_k \sim \quad \mathrm{Beta}(x_k|\alpha_k^0, \beta_k^0) = \tfrac{\Gamma(\alpha_k^0+\beta_k^0)}{\Gamma(\alpha_k^0)\Gamma(\beta_i^0)} x_k^{\alpha_k^0-1}(1-x_k)^{\beta_k^0-1}, \tag{7.1}$$

$$\mathcal{T}_k \sim \quad \mathrm{Beta}(y_k|\alpha_k^1, \beta_k^1) = \tfrac{\Gamma(\alpha_k^1+\beta_k^1)}{\Gamma(\alpha_k^1)\Gamma(\beta_i^1)} y_k^{\alpha_k^1-1}(1-y_k)^{\beta_k^1-1}, \tag{7.2}$$

where $\Gamma(\cdot)$ is the gamma function [70], and its parameters α_k^0, α_k^1 and β_k^0, β_k^1 are given by

$$\begin{aligned} \alpha_k^0 &= \sum_{j=1}^{u} \lambda^{t_{k,j}^0} r_{k,j}^0 \qquad & \beta_k^0 &= \sum_{j=1}^{u} \lambda^{t_{k,j}^0} (1 - r_{k,j}^0); \\ \alpha_k^1 &= \sum_{j=1}^{v} \lambda^{t_{k,j}^1} r_{k,j}^1 \qquad & \beta_k^1 &= \sum_{j=1}^{v} \lambda^{t_{k,j}^1} (1 - r_{k,j}^1), \end{aligned} \tag{7.3}$$

where $\alpha_k^0, \beta_k^0, \alpha_k^1, \beta_k^1$ are the cumulated instances of false positive, true negative, true positive, and false negative, respectively, from acquaintance k. $r_{k,j}^0 \in \{0,1\}$ is the jth diagnosis result from acquaintance k under no-attack. $r_{k,j}^0 = 1$ means the diagnosis from k is positive while there is actually no attack happening. $r_{k,j}^0 = 0$ means otherwise. Similarly, $r_{k,j}^1 \in \{0,1\}$ is the jth diagnosis data from acquaintance k under attack where $r_{k,0}^1 = 1$ means that the diagnosis from k is positive under attack, and $r_{k,0}^1 = 0$ means otherwise. Parameters $t_{k,j}^0$ and $t_{k,j}^1$ denote the time elapsed since the jth feedback is received. $\lambda \in [0,1]$ is the forgetting factor on the past experience. A small λ makes old observations quickly forgettable. We use exponential moving average to accumulate past experience so that old experience takes less weight than new experience. u is the total number of no-attack cases among the past records and v is the total number of attack cases.

To make the parametric updates scalable to data storage and memory, we can use the following recursive formula to update α_k^0, α_k^1 and β_k^0, β_k^1:

$$\begin{aligned} \alpha_k^m(t_j) &= \lambda^{(t_{k,j}^m - t_{k,j-1}^m)} \alpha_k^m(t_{k,j-1}^m) + r_{k,j}^m; \\ \beta_k^m(t_j) &= \lambda^{(t_{k,j}^m - t_{k,j-1}^m)} \beta_k^m(t_{k,j-1}^m) + r_{k,j}^m, \end{aligned} \tag{7.4}$$

where $l = 0, 1$ and $j - 1$ indexes the previous data point used for updating α_k^m or β_k^m. Through this way, only the previous state and the current state are required to be recorded, which is efficient in terms of storage compared to when all states are recorded in Equation 7.3.

7.3.2 Feedback Aggregation

When an IDS detects suspicious activities and is not confident about its decision, it sends out the description of the suspicious activities or the related executable files to its collaborators for consultation. The node receives diagnosis results from its collaborators, denoted by vector $\mathbf{y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{|\mathcal{A}|}\}$, where $\mathbf{y}_i \in \{0, 1\}$, for $0 < i < |\mathcal{A}|$, is the feedback from acquaintance i. We use $X \in \{0, 1\}$ to denote the scenario of "no-attack" or "under-attack", and $\mathbf{Y} \in \{0, 1\}^{|\mathcal{A}|}$ to denote all possible feedback from acquaintances. The conditional probability of an IDS being "under-attack" given the diagnosis results from all acquaintances can be written as $\mathbb{P}[X = 1|\mathbf{Y} = \mathbf{y}]$. Using Bayes' Theorem [95] and assuming that the acquaintances provide diagnoses independently and their FP rate and TP rate are known, we have

$$\begin{aligned}
\mathbb{P}[X{=}1|\mathbf{Y}{=}\mathbf{y}] &= \frac{\mathbb{P}[\mathbf{Y}{=}\mathbf{y}|X{=}1]\mathbb{P}[X{=}1]}{\mathbb{P}[\mathbf{Y}{=}\mathbf{y}|X{=}1]\mathbb{P}[X{=}1]{+}\mathbb{P}[\mathbf{Y}{=}\mathbf{y}|X{=}0]\mathbb{P}[X{=}0]} \\
&= \frac{\pi_1 \prod_{k=1}^{|\mathcal{A}|} T_k^{\mathbf{y}_k}(1 - T_k)^{1-\mathbf{y}_k}}{\pi_1 \prod_{k=1}^{|\mathcal{A}|} T_k^{\mathbf{y}_k}(1 - T_k)^{1-\mathbf{y}_k} + \pi_0 \prod_{k=1}^{|\mathcal{A}|} F_k^{\mathbf{y}_k}(1 - F_k)^{1-\mathbf{y}_k}},
\end{aligned}$$

where $\pi_0 = \mathbb{P}[X = 0]$ and $\pi_1 = \mathbb{P}[X = 1]$, such that $\pi_0 + \pi_1 = 1$, are the prior probabilities of the scenarios of "no-attack" and "under-attack", respectively. $\mathbf{y}_k \in \{0, 1\}$ is the kth element of vector $\mathbf{y}$.

Since T_k and F_k are both random variables with distributions as in Equations (7.1) and (7.2), we can see that the conditional probability $\mathbb{P}[X = 1|\mathbf{Y} = \mathbf{y}]$ is also a random variable. We use a random variable P to denote $\mathbb{P}[X = 1|\mathbf{Y} = \mathbf{y}]$. Then P takes a continuous value over domain $[0, 1]$. We use $f_P(p)$ to denote the probability density function of P.

When α and β are sufficiently large, a Beta distribution can be approximated by Gaussian distribution according to $\text{Beta}(\alpha, \beta) \approx N\left(\frac{\alpha}{\alpha+\beta}, \sqrt{\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}\right)$. Then the density function of P can be also approximated using Gaussian distribution. By Gauss's approximation formula, we have,

$$\begin{aligned}\mathbb{E}[P] &\approx \frac{1}{1+\frac{\pi_0 \prod_{k=1}^{|\mathcal{A}|} \mathbb{E}[F_k]^{\mathbf{y}_k}(1-\mathbb{E}[F_k])^{1-\mathbf{y}_k}}{\pi_1 \prod_{k=1}^{|\mathcal{A}|} \mathbb{E}[T_k]^{\mathbf{y}_k}(1-\mathbb{E}[T_k])^{1-\mathbf{y}_k}}} \\ &= \frac{1}{1+\frac{\pi_0}{\pi_1}\prod_{k=1}^{|\mathcal{A}|} \frac{\alpha_k^1+\beta_k^1}{\alpha_k^0+\beta_k^0}(\frac{\alpha_k^0}{\alpha_k^1})^{\mathbf{y}_k}(\frac{\beta_k^0}{\beta_k^1})^{1-\mathbf{y}_k}}. \end{aligned} \quad (7.5)$$

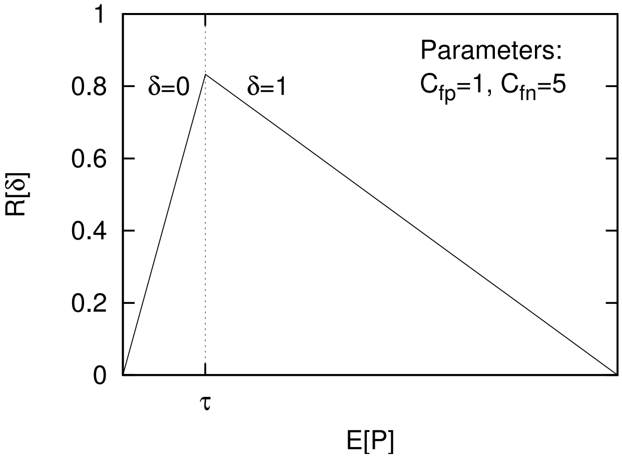


Figure 7.1: Bayes Risk for Optimal Decisions when $C_{fp} = 1$ and $C_{fn} = 5$

Let C_{fp} and C_{fn} denote the marginal cost of a FP decision and a FN decision. We assume there is no cost when a correct decision is made. We use marginal cost because the cost of a FP may change in time depending on the current state. C_{fn} largely depends on the potential damage level of the attack. For example, an intruder intending to track a user's browsing history may have lower C_{fn} than an intruder intending to modify a system file. We define a decision function $\delta(\mathbf{y}) \in \{0, 1\}$, where $\delta = 1$ means raising an alarm and

$\delta = 0$ means no alarm. Then, the Bayes risk can be written as:

$$\begin{aligned} R(\delta) &= \int_0^1 (C_{fp}(1-x)\delta + C_{fn}x(1-\delta))f_P(x)dx \\ &= \delta C_{fp}\int_0^1 (1-p)f_P(p)dp + (1-\delta)C_{fn}\int_0^1 pf_P(p)dp \\ &= \int_0^1 C_{fn}xf_P(x)dx + \delta\left(C_{fp} - (C_{fp}+C_{fn})\int_0^1 xf_P(x)dx\right) \\ &= C_{fn}\mathbb{E}[P] + \delta(C_{fp} - (C_{fp}+C_{fn})\mathbb{E}[P]), \end{aligned} \tag{7.6}$$

where $f_P(p)$ is the density function of P. To minimize the risk $R(\delta)$, we need to minimize $\delta(C_{fp} - (C_{fp}+C_{fn})\mathbb{E}[P])$. Therefore, we raise an alarm (i.e. $\delta = 1$) if

$$\mathbb{E}[P] \geq \frac{C_{fp}}{C_{fp}+C_{fn}}. \tag{7.7}$$

Let $\tau = \frac{C_{fp}}{C_{fp}+C_{fn}}$ be the threshold. If $\mathbb{E}[P] \geq \tau$, we raise an alarm, otherwise no alarm is raised. The corresponding Bayes risk for the optimal decision is:

$$R(\delta) = \begin{cases} C_{fp}(1-\mathbb{E}[P]) & \text{if } \mathbb{E}[P] \geq \tau, \\ \\ C_{fn}\mathbb{E}[P] & \text{otherwise.} \end{cases} \tag{7.8}$$

An example of the Bayes risk for optimal decisions when $C_{fp} = 1$ and $C_{fn} = 5$ is illustrated in Figure 7.1.

7.4 Acquaintance Management

Intuitively when an IDS consults a larger number of acquaintances, it can achieve higher detection accuracy and lower risk of being compromised. However, having more acquaintances causes higher maintenance cost since the IDS needs to allocate resources for each node in its acquaintance list. When an IDS makes a decision about how many acquaintances to recruit, both the intrusion risk cost and the maintenance cost should be taken into account. When adding a node as an acquaintance does not lower the total cost, then the node shall not be added into the acquaintance list. However, how to select acquaintances and how many acquaintances to include are crucial to build an efficient IDN. In this section, we first define the acquaintance selection problem, then a corresponding solution is devised to find the optimal set of acquaintances. Finally, we propose an acquaintance management algorithm for IDSs to learn, recruit, update, or remove their acquaintances dynamically.

7.4.1 Problem Statement

Let $\mathcal{A}_i$ denote the set of acquaintances of IDS i. Let $M_i(\mathcal{A}_i)$ be the cost for IDS i to maintain the acquaintance set $\mathcal{A}_i$. We use $R_i(\mathcal{A}_i)$ to denote the risk cost of missing intrusions and/or false alarms for IDS i, given the feedback of acquaintance set $\mathcal{A}_i$. In the rest of this section, we drop subscript i from our notations for the convenience of presentation.

Our goal is to select a set of acquaintances from a list of candidates so that the overall cost $R(\mathcal{A}) + M(\mathcal{A})$ is minimized. We define the problem as follows:

Given a list of acquaintance candidates $\mathcal{C}$, we need to find a subset of acquaintances $\mathcal{A} \subseteq \mathcal{C}$, such that the overall cost $R(\mathcal{A}) + M(\mathcal{A})$ is minimized.

In practice, maintenance cost of acquaintances may not be negligible since acquaintances send test messages/consultations periodically to ask for diagnosis. It takes resources (CPU and memory) for the IDS to receive, analyze the requests, and reply with corresponding answers. The selection of $M_i(.)$ can be user defined on each host. For example, a simple maximum acquaintance length restriction can be mapped to $M(\mathcal{A}) = C\max(|\mathcal{A}| - L, 0)$, where $L \in \mathcal{N}^+$ is the acquaintance length upper-bound and $C \in [0, \infty)$ is the penalty of exceeding the bound.

The risk cost can be expressed as:

$$\begin{aligned} R(\mathcal{A}) = {} & C_{fn}P[\delta = 0|X = 1]P[X = 1] \\ & + C_{fp}P[\delta = 1|X = 0]P[X = 0] \end{aligned}$$

where C_{fn}, C_{fp} denote the marginal cost of missing an intrusion and raising a false alarm, respectively. $P[X = 1] = \pi_1, P[X = 0] = \pi_0$ are the prior probabilities of under-attack and no-attack, where $\pi_0 + \pi_1 = 1$. Note that in practice π_1 can be learned from the history and be updated whenever a new threat is found. A moving average method can be used to update the estimated value.

The above equation can be further written as:

$$\begin{aligned} R(\mathcal{A}) &= C_{fn}\pi_1 \sum_{\forall \mathbf{y}\in\{0,1\}^{|\mathcal{A}|}|\delta(\mathbf{y})=0} P[\mathbf{Y}=\mathbf{y}|X=1] \\ &+ C_{fp}\pi_0 \sum_{\forall \mathbf{y}\in\{0,1\}^{|\mathcal{A}|}|\delta(\mathbf{y})=1} P[\mathbf{Y}=\mathbf{y}|X=0] \\ &= C_{fn}\pi_1 \sum_{\forall \mathbf{y}\in\{0,1\}^{|\mathcal{A}|}|\delta(\mathbf{y})=0} \prod_{i=1}^{|\mathcal{A}|} (T_i)^{\mathbf{y}_i}(1-T_i)^{1-\mathbf{y}_i} \\ &+ C_{fp}\pi_0 \sum_{\forall \mathbf{y}\in\{0,1\}^{|\mathcal{A}|}|\delta(\mathbf{y})=1} \prod_{i=1}^{|\mathcal{A}|} (F_i)^{\mathbf{y}_i}(1-F_i)^{1-\mathbf{y}_i} \\ &= C_{fn}\pi_1 \sum_{\forall \mathbf{y}\in\{0,1\}^{|\mathcal{A}|}|f(\mathbf{y})<1} \prod_{i=1}^{|\mathcal{A}|} (T_i)^{\mathbf{y}_i}(1-T_i)^{1-\mathbf{y}_i} \\ &+ C_{fp}\pi_0 \sum_{\forall \mathbf{y}\in\{0,1\}^{|\mathcal{A}|}|f(\mathbf{y})\geq 1} \prod_{i=1}^{|\mathcal{A}|} (F_i)^{\mathbf{y}_i}(1-F_i)^{1-\mathbf{y}_i} \\ &= \sum_{\mathbf{y}\in\{0,1\}^{|\mathcal{A}|}} min\{C_{fn}\pi_1 \prod_i T_i^{\mathbf{y}_i}(1-T_i)^{1-\mathbf{y}_i}, \\ & C_{fp}\pi_0 \prod_i F_i^{\mathbf{y}_i}(1-F_i)^{1-\mathbf{y}_i}\} \end{aligned} \tag{7.9}$$

where T_i, F_i are the TP rate and FP rate of acquaintance i respectively.

$$f(\mathbf{y}) = \frac{C_{fn}\pi_1 \prod_{i=1}^{|\mathcal{A}|} (T_i)^{\mathbf{y}_i}(1-T_i)^{1-\mathbf{y}_i}}{C_{fp}\pi_0 \prod_{i=1}^{|\mathcal{A}|} (F_i)^{\mathbf{y}_i}(1-F_i)^{1-\mathbf{y}_i}}.$$

$\forall \mathbf{y} \in \{0,1\}^l|\delta(y) = 1$ refers to the combination of decisions which causes the system to raise an alarm and vice versa.

7.4.2 Acquaintance Selection Algorithm

To solve such a subset optimization problem, the brute force method is to examine all possible combinations of acquaintances and select the one which has the least overall cost.

However, the computation complexity is $O(2^n)$. It is not hard to see that the order of selecting acquaintances does not affect the overall cost. We propose an acquaintance selection algorithm based on a heuristic approach to find an acquaintance set which achieves satisfactory overall cost. In this algorithm, the system always selects the nodes which bring the lowest overall cost.

For the ease of demonstration, We assume the maintenance cost can be written as follow:

$$M(\mathcal{A}) = C_a l = C_a|\mathcal{A}| \tag{7.10}$$

where C_a is the unit maintenance cost of each acquaintance, which includes the cost of communication, detection assistance, and test messages. Note that any other form of maintenance cost can be easily included into the algorithm.

As shown in Algorithm 1, in the beginning, the acquaintance list is empty. The initial cost is the minimum cost of the decision based only on the prior information (line 3). For each loop, the system selects a node from the acquaintance candidate list which brings the lowest overall cost and stores it into e_{max} (lines 7-14), where $U - R(\mathcal{A}) - M(\mathcal{A})$ is the amount of cost reduced by adding a node into the acquaintance list. When such a node is found, it is then moved to the acquaintance list if the current acquaintance length is less than L_{min} or the cost is reduced by adding the new node and the acquaintance length does not exceed L_{max}. The loop stops when no node can be added into $\mathcal{A}$ any further.

7.4.3 Acquaintance Management Algorithm

In the previous section, we devised an algorithm to select acquaintances from a list of candidates. However, collaboration is usually based on mutual consensus. If node A selects B as an acquaintance but B does not select A (non-symmetric selection), then the collaboration is not established.

We propose a distributed approach for an IDS in the IDN to select and manage acquaintances and a consensus protocol to allow an IDS to deal with the non-symmetric selection problem. To improve the stability of the acquaintance list, we propose to use a probation period on each new node for the IDS to learn about the new node before considering it as an acquaintance. For this purpose, each IDS maintains a *probation list*, where all new

Algorithm 3 Acquaintance Selection ($\mathcal{C}, L_{min}, L_{max}$)

```
Require: A set of acquaintance candidates $\mathcal{C}$
Ensure: A set of selected acquaintances $\mathcal{A}$ with minimum length $L_{min}$ and max length
    $L_{max}$ which brings the minimum overall cost
$Quit = false$ //quit the loop if $Quit = true$
$\mathcal{A} \Leftarrow \emptyset$
$U = min(\pi_0 C_{fp}, \pi_1 C_{fn})$ //initialize the overall cost while there is no acquaintance.
    $min(\pi_0 C_{fp}, \pi_1 C_{fn})$ is the cost when a node makes a decision without feedback from
    collaborators
while $Quit = false$ do
    //select the node that reduces cost most in each iteration
    $D_{max} = -MAXNUM$ //initialize the maximum cost reduction to the lowest possible
    for all $e \in \mathcal{C}$ do
        $\mathcal{A} = \mathcal{A} \cup e$
        if $U - R(\mathcal{A}) - M(\mathcal{A}) > D_{max}$ //see Equation (7.9) and Equation (7.10) for $R(\mathcal{A})$
        and $M(\mathcal{A})$ then
            $D_{max} = U - R(\mathcal{A}) - M(\mathcal{A})$
            $e_{max} = e$
        end if
        $\mathcal{A} = \mathcal{A} \setminus e$ //remove $e$ from $\mathcal{A}$
    end for
    if ($D_{max} > 0$ and $|\mathcal{A}| < L_{max}$) or $|\mathcal{A}| < L_{min}$ then
        $\mathcal{A} = \mathcal{A} \cup e_{max}$
        $\mathcal{C} = \mathcal{C} \setminus e_{max}$ //remove $e_{max}$ from $\mathcal{C}$
        $U = U - D_{max}$
    else
        $Quit = true$
    end if
end while
```

Algorithm 4 Managing Acquaintance & Probation Lists

```
**Initialization** :
$\mathcal{A} \Leftarrow \emptyset$ //Acquaintance list.
$\mathcal{P} \Leftarrow \emptyset$ //Probation list.
$l^p = l^{ini}$ //initial Probation length
//Fill $\mathcal{P}$ with randomly selected nodes
**while** $|\mathcal{P}| < l^p$ **do**
    $e \Leftarrow$ select a random node
    $\mathcal{P} \Leftarrow \mathcal{P} \cup e$
**end while**
**set** new timer event($t_u$, "**SpUpdate**")
**Periodic Maintenance:**
**at timer event** ev of type "**SpUpdate**" **do**
//Merge the first mature node into the acquaintance list.
$e \Leftarrow selectOldestNode(\mathcal{P})$
$\mathcal{C} \Leftarrow \mathcal{A}$ //$\mathcal{C}$ is the temporary candidate list
**if** $t_e > t_p$ //$t_e$ is the age of node $e$ in the probation list **then**
    $\mathcal{P} \Leftarrow \mathcal{P} \setminus e$
    **if** $T_e > T_{min}$ and $F_e < F_{max}$ //$T_e$ and $F_e$ are the true positive rate and false positive rate of the node $e$ **then**
        $\mathcal{C} \Leftarrow \mathcal{C} \cup e$
    **end if**
**end if**
//Consensus protocol
$\mathcal{S}$ =Acquaintance Selection($\mathcal{C}, l^{min}, \max(l^{min}, \frac{q}{q+1} l^{max})$)
//Send requests for collaboration and receive responses
$S_{accp} \Leftarrow RequestandReceiveCollaboration(S, t_{timeout})$
$\mathcal{A} \Leftarrow S_{accp}$ //Only nodes that accept the collaboration invitations are moved into the acquaintance list
//Refill $\mathcal{P}$ with randomly selected nodes
**while** $|\mathcal{P}| < max(q|\mathcal{A}|, l^{min})$ **do**
    $e \Leftarrow$ Select a random node not in $\mathcal{A}$
    $\mathcal{P} \Leftarrow \mathcal{P} \cup e$
**end while**
**set** new timer event($t_u$, "**SpUpdate**")
**end timer event**
```

nodes remain during their probation periods. A node also communicates with nodes in its probation list periodically to evaluate their detection accuracy. The purpose of the probation list is thus to explore potential collaborators and keep introducing new qualified nodes to the acquaintance list.

Suppose that node i has two sets $\mathcal{A}_i$ and $\mathcal{P}_i$, which are the acquaintance list and the probation list respectively. The corresponding false positive rate and true positive rate of both sets are $F_i^{\mathcal{A}}, T_i^{\mathcal{A}}$ and $F_i^{\mathcal{P}}, T_i^{\mathcal{P}}$. To keep learning the detection accuracy of the acquaintances, a node sends test messages to nodes in both the acquaintance list and the probation list periodically, and keeps updating their estimated false positive rates and true positive rates. Let l^{max} be the maximum number of IDSs in both the acquaintance and the probation list. We set this upper-bound because the amount of resources used for collaboration is proportional to the number of acquaintances it manages. l^{max} is determined by the resource capacity of each IDS. Let l^{min} be the minimum length of a probation list and q be the parameter that controls the length of the probation list l^p compared to the length of acquaintance list l^a, such that $l^{min} \leq l^p \leq ql^a$. The parameters l^{min} and q are used to tune the trade-off between the adaptability to the situation where nodes join or leave the network frequently ("high churn rate"), and the overhead of resources used for testing new nodes.

The acquaintance management procedure for each node is shown in Algorithm 2. The acquaintance list $\mathcal{A}$ is initially empty and the probation list $\mathcal{P}$ is filled by l^{ini} random nodes to utilize the resources in exploring new nodes. An acquaintance list updating event is triggered every t_u time units. $\mathcal{A}$ is updated by including new trusted nodes from $\mathcal{P}$. A node that stays at least t_p time units in probation is called a *mature node*. Only mature nodes are allowed to join the acquaintance list (lines 15-21). Mature nodes with bad qualification will be abandoned right away. After that the acquaintance selection algorithm is used to find the optimal candidate list. Collaboration requests are sent out for nodes which are selected in the optimal list. If an acceptance is received before expiration time then the collaboration is confirmed, otherwise the node is abandoned (lines 22-26). Then, $\mathcal{P}$ is refilled with new randomly chosen nodes (lines 28-31).

Several properties are desirable for an effective acquaintance management algorithm, including convergence, stability, robustness, and incentive-compatibility for collaboration. When our acquaintance management is in place, we are interested to know with whom the IDS nodes end up collaborating with and how often they change their collaborators. We also expect to see cooperative nodes are rewarded and dishonest nodes penalized.

In Section 7.5 we evaluate our acquaintance management algorithm, to determine whether it achieves the above properties.

7.5 Evaluation

In this section, we describe the conducted simulation to demonstrate the desirable properties of our acquaintance management algorithm. We evaluate the cost efficiency of our Bayesian decision model, cost and time efficiency of the acquaintance selection algorithm, and several desired properties of the acquaintance management algorithm. Each simulation result presented in this section is derived from the average of a large number of replications with an overall negligible confidence interval.

Table 7.2: Simulation Parameters

Parameter	Range	Value	Description
R	$[0, \infty)$	10/day	Test message rate
λ	[0, 1]	0.95	Forgetting factor
C_{fp}/C_{fn}	$[0, \infty)$	20/100	Unit cost of false positive/negative decisions
C_a	$[0, \infty)$	0.01	Maintenance cost of one acquaintance
t_p	$[0, \infty)$	10 days	Probation period
t_u	$[0, \infty)$	1 day	Acquaintance list update interval
l^{ini}	$\mathbb{N}^+$	10	Initial probation length
l^{max}	$\mathbb{N}^+$	20	Maximum total number of acquaintances
l^{min}	$\mathbb{N}^+$	2	Minimum probation list length
T^{min}	[0,1]	0.5	Minimum acceptable true positive rate
F^{max}	[0,1]	0.2	Maximum acceptable false positive rate
q	$[0, \infty)$	0.5	Length ratio of probation to acquaintance list
π_1	[0, 1]	0.1	Prior probability of intrusions

7.5.1 Simulation Setting

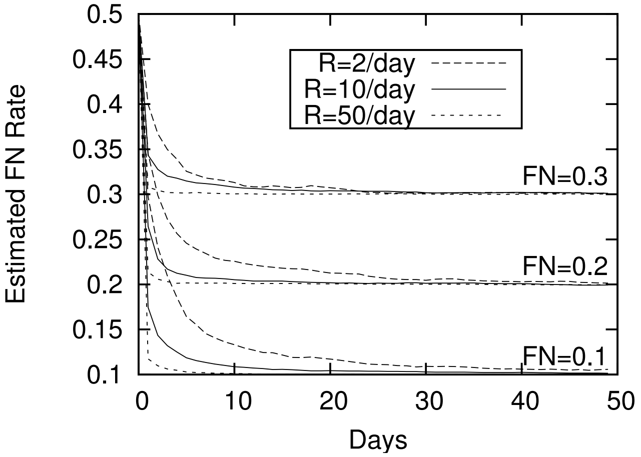


Figure 7.2: The Convergence of Learning Speed and the Test Message Rate

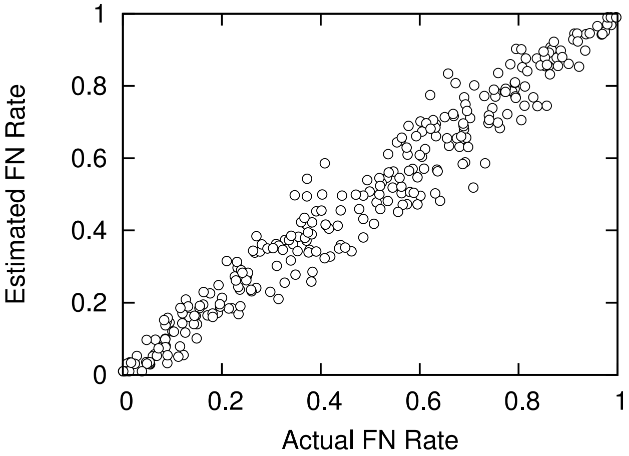


Figure 7.3: The distribution of estimated FN rate (R=10/day)

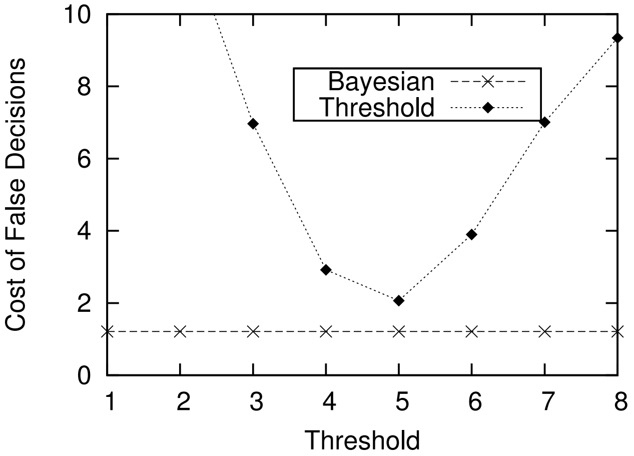


Figure 7.4: Comparison of Cost using Threshold Decision and Bayesian Decision

We simulate an environment of n IDS peers collaborating together by adding each other as acquaintances. We adopt two parameters to model the detection accuracy of each IDS, namely, false positive rate (FP) and false negative rate (FN). Notice that in reality most IDSs have low FP (< 0.1) and FN is normally in the range of $[0.1, 0.5]$ [89]. This is because false positives can severely damage the reputation of the product, so vendors strive to control their FP rate at a low level. In our experiment, we select parameters which reflect real world properties. To test the detection accuracy of acquaintances, each peer sends test messages where their correct answers are known beforehand. Test messages are sent following a Poisson process with average arrival rate R. R will be determined in the next subsection. We use a simulation day as the time unit in our experiments. The diagnosis results given by an IDS are simulated following a Bernoulli random process. If a test message represents a benign activity, the IDS i raises alarm with a probability of FP_i. Similarly, if the test message represents intrusions, an alarm will be raised with a probability of 1-FN_i. All parameter settings are summarized in Table 7.2.

7.5.2 Determining the Test Message Rate

The goal of our first experiment is to study the relationship between test message rates and FP, FN learning speed. We simulate two IDSs A and B. A sends B test messages to ask for diagnosis, and learns the FP and FN of B based on the quality of B's feedback. The learning procedure follows Equations (7.1), (7.2), and (7.3). We fix the FN of B to 0.1, 0.2, and 0.3 respectively. Under each case, we run the learning process under different test message rates, 2/day, 10/day, and 50/day respectively. We observe the change of estimated FN over time, plotted in Figure 7.2. We see that when R is 2/day, the estimated FN converges after around 30 days in the case of FN=0.2. The converging time is slightly longer and shorter in the cases of FN=0.3 and FN=0.1, respectively. When R is increased to 10/day, the converging time decreases to around 10 days. In the case of R=50/day, the corresponding converging time is the shortest (around 3 days) among the three cases. Increasing the test message rate R to 50/day does not reduce much learning process time. Based on the above observation, we choose R=10/day and the probation period t_p to be 10 days as our system parameters. In this way, the test message rate is kept low and the learned FN and FP values converge after the probation period.

The second experiment is to study the efficiency of learning results after our chosen probation period. We fix R=10/day, t_p=10/day, and randomly choose FN of node B uniformly among [0, 1]. We repeat the experiments 100 times with different FNs. The FNs estimated using our learning process till the end of probation period are plotted in Figure 7.3. We can see that in all different settings of FNs, the estimated FN rates are close to the actual FN rates after the probation period.

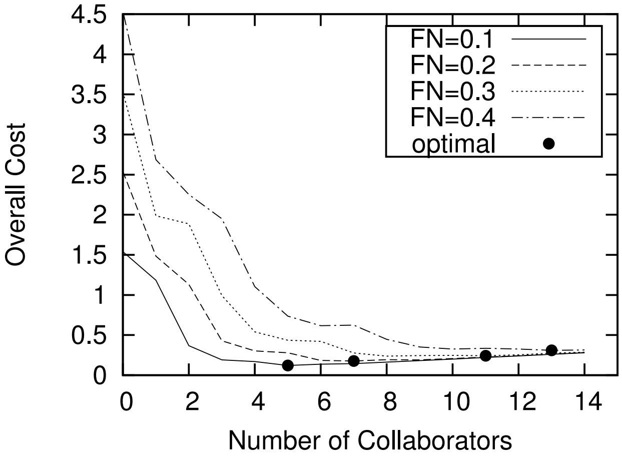


Figure 7.5: The Overall Cost under Different Collaborator Quality

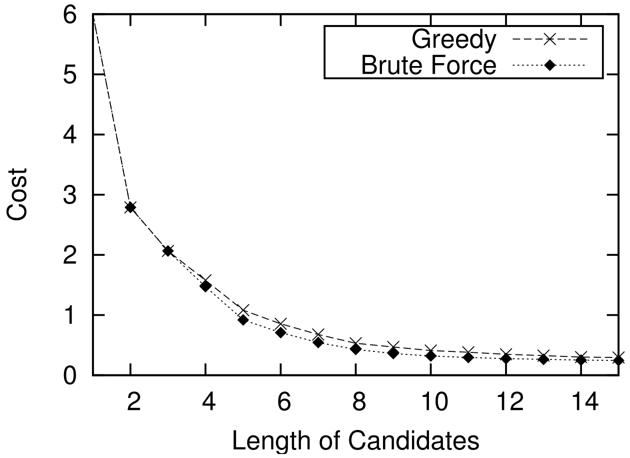


Figure 7.6: The Cost using Different Acquaintance Selection Algorithms

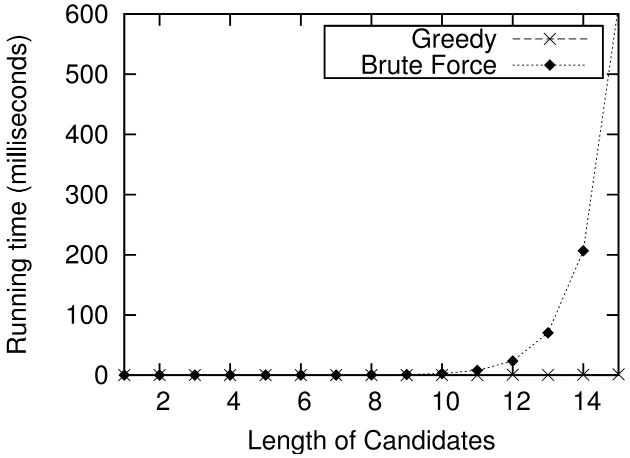


Figure 7.7: The Running Time using Different Acquaintance Selection Algorithms

7.5.3 Efficiency of our Feedback Aggregation

In this experiment, we evaluate the effectiveness of our Bayesian decision based feedback aggregation by comparing it with a threshold based aggregation. We have described our Bayesian decision model in Section 7.3.2. In a simple threshold based feedback aggregation method, if the number of IDSs reporting intrusions is larger than a predefined threshold, then the system raises an alarm. The threshold-based decision is used in N-version cloud anti-virus systems [89].

We set up eight IDSs $\{\mathrm{IDS}_0, \mathrm{IDS}_1, ..., \mathrm{IDS}_7\}$ with their FP and FN rates randomly chosen from the range [0.1, 0.5]. IDS_0 sends consultations to all other IDSs, collects and aggregates feedback to make intrusion decisions. The costs of false positive and false negative decisions are C_{fp}=20 and C_{fn}=100 respectively. We compare the average false detection cost using the Bayesian decision model and the simple threshold-based approach. Figure 7.4 shows that the cost of threshold decision largely depends on the chosen threshold value. An appropriate threshold can significantly decrease the cost of false decisions. In contrast, the Bayesian decision model does not depend on any threshold setting and prevails over the threshold decision under all threshold settings. This is because the threshold decision treats all participants equally, while the Bayesian decision method recognizes different detection capabilities of IDSs and takes them into account in the decision process. For example, if an IDS asserts that there is intrusion, our Bayesian model may raise an alarm if the IDS has a low FP rate and ignores the warning if the IDS has a high FP rate. However, the threshold based decision model will either raise an alarm or not based on the total number of IDSs which raise warnings and compare it with a predefined threshold, irrespective of the individual that issued the warning.

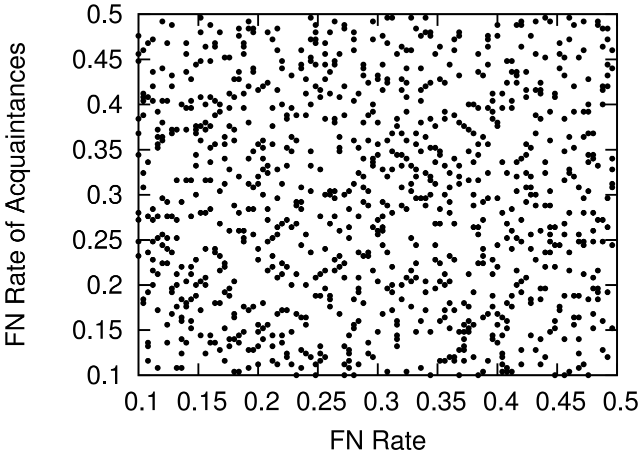


Figure 7.8: Acquaintances Distribution on Day 25

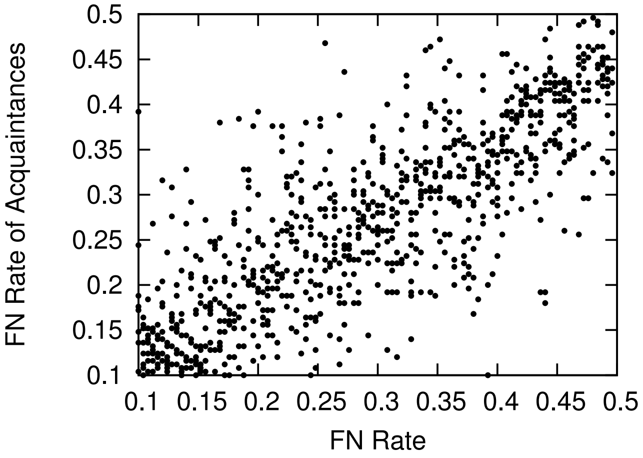


Figure 7.9: Acquaintances Distribution on Day 200

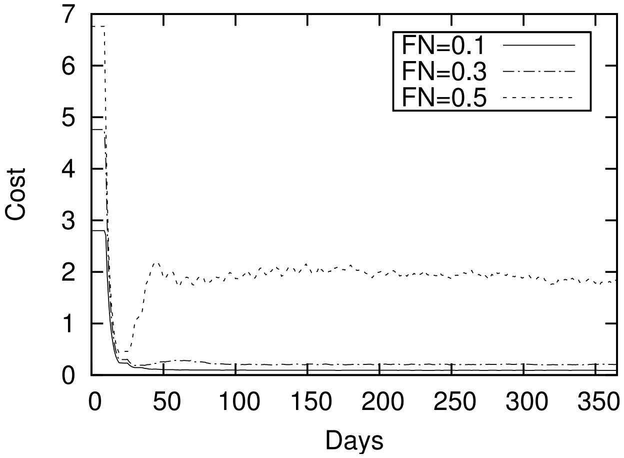


Figure 7.10: The Average Cost for Collaboration

7.5.4 Cost and the Number of Collaborators

We define *risk cost* to be the expected cost from false decisions such as raising false alarms (FP) and missing the detection of an intrusion (FN). We show that introducing more collaborators can decrease the risk cost. In this experiment, we study the impact of the number of collaborators on the risk cost. We set up four groups with an equal number of IDSs. Nodes in all groups have the same FP rate of 0.03, but their FN rates vary from 0.1 to 0.4, depending on the group they are in. Inside each group every node collaborates with every other node. We are interested in the risk cost as well as the maintenance cost. The *maintenance cost* is the cost associated with the amount of resource that is used to maintain the collaboration with other nodes, such as answering diagnosis requests from other IDSs. Since our purpose is to capture the concept of maintenance cost but not to study how much it is, we assume the maintenance cost to be linearly proportional to the number of collaborators with a unit rate C_a=0.01 (see Table 7.2).

We increase the size of all groups and observe the overall cost of nodes in each group. From Figure 7.5, we can see that in all groups, the costs drop down fast in the beginning and slow down as the groups' sizes increase. After an optimal point (marked by large solid circles), the costs slowly increase. This is because when the number of collaborators is large enough, the cost saving by adding more collaborators becomes small, and the increment of maintenance cost becomes significant. We find that groups with higher detection accuracy have lower optimal costs. Also they need a smaller number of collaborators to reach the optimal costs. For example, in the case of $FN = 0.4$, 13 collaborators are needed to reach the optimal cost, while the number of collaborators required is 5 in the case of $FN = 0.1$.

7.5.5 Efficiency of Acquaintance Selection Algorithms

We learned in the previous section that when the number of collaborators is large enough, adding more collaborators does not decrease the overall cost because of the associated maintenance cost. An acquaintance selection algorithm is proposed in Algorithm 3. In this section, we compare the efficiency of acquaintance selection using the brute force algorithm and our acquaintance selection algorithm. We create 15 IDSs as candidate acquaintances with FP and FN rates randomly chosen from intervals $[0.01, 0.1]$ and $[0.1, 0.5]$, respectively. Both algorithms are implemented in Java and run on a PC with AMD Athlon dual core processor 2.61GHZ, and with 1.93 GB RAM. We start the candidate set size from 1 and gradually increase the size. We observe the cost efficiency and running time efficiency of both algorithms.

Figure 7.6 shows that the brute force algorithm performs slightly better with respect to acquaintance list quality since the overall cost using its selected list is slightly lower.

However, Figure 7.7 shows that the running time of the brute force method increases significantly when the candidate set size exceeds 11, and continues to increase exponentially, while our algorithm shows much better running time efficiency. These experiments suggest to use the brute force method only when the size of candidates list is small (≤ 11). When the candidates list is large, our greedy algorithm should be used to select acquaintances.

7.5.6 Evaluation of Acquaintance Management Algorithm

In this experiment, we study the effectiveness of our acquaintance management algorithm (Algorithm 4). We set up a simulation environment of 100 nodes. For the convenience of observation, all nodes have fixed FP rate 0.1 and their FN rates are uniformly distributed in the range of $[0.1, 0.5]$. All nodes update their acquaintance list once a day (t_u=1). We observe several properties: convergence, stability, robustness, and incentive-compatibility.

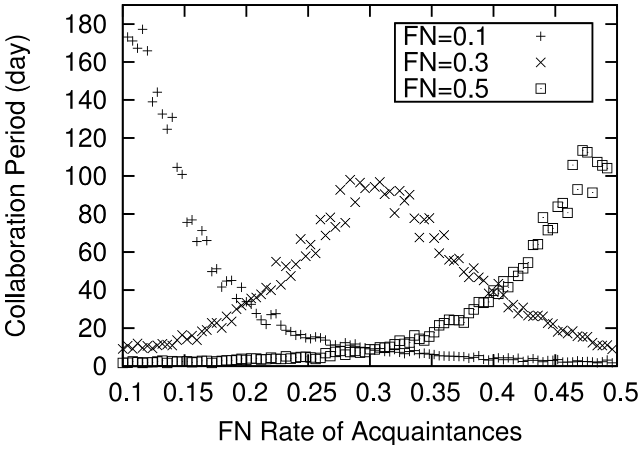


Figure 7.11: The Collaboration Time Span

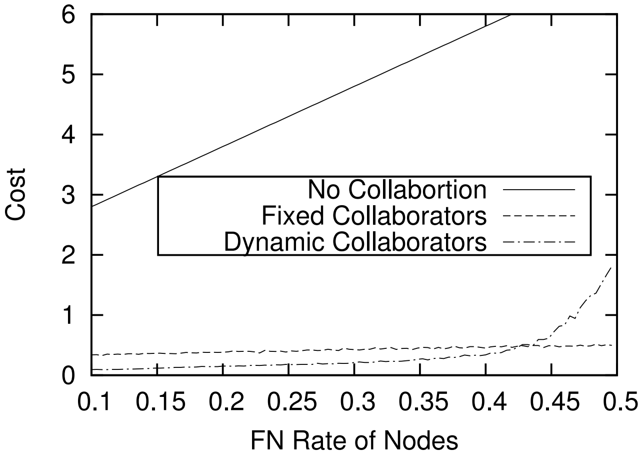


Figure 7.12: The Converged Cost Distribution

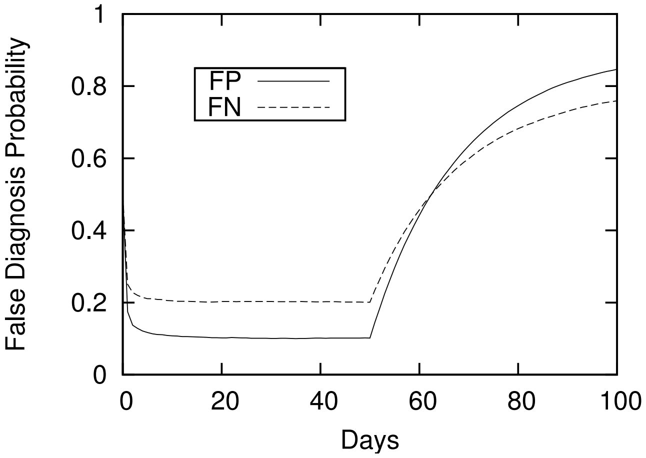


Figure 7.13: The FP and FN of Betrayal Node

Convergence

Our first finding about our acquaintance management algorithm is that IDSs converge to collaborating with other IDSs with similar detection accuracy levels. We observed through experiments that IDSs collaborate with random other nodes in the network in the beginning (Figure 7.8). After a longer period of time (200 days), all IDSs collaborate with others with similar detection accuracy, as shown in Figure 7.9. Our explanation is that the collaboration between pairs with high qualification discrepancy is relatively not stable since our collaboration algorithm is based on mutual consensus and consensus is hard to reach between those pairs.

Figure 7.10 plots the average overall cost in the first 365 days of collaboration for three nodes with FN values $0.1, 0.3$, and 0.5 respectively. In the first 10 days, the costs for all nodes are high. This is because all collaborators are still in probation period. After day 10, all cost values drop down significantly. This is because collaborators pass the probation period and start to contribute to intrusion decisions. The cost for high expertise nodes continues to drop while the cost for low expertise nodes increases partially after around day 20, and stabilizes after day 50. This is because the acquaintance management algorithm selects better collaborators to replace the initial random ones. We can see that the collaboration cost of nodes converges with time and becomes stable after the initial phase.

Stability

Collaboration stability is an important property since the collaboration between IDSs is expected to be long term. Frequently changing collaborators is costly because IDSs need to spend considerable amount of time to learn about new collaborators. In this experiment, we record the average time span of all acquaintances from the time they pass the probation period till they are replaced by other acquaintances. The result is shown in Figure 7.11, where the average collaboration time spans for three selected nodes are shown with different point shapes. We can see that collaboration among nodes with similar expertise levels is more stable than that between nodes with different expertise levels. For example, nodes with low $FN = 0.1$ form stable collaboration connections with other nodes with low FN (around 180 days in average), while the collaboration with IDSs with high FN is short (close to 0 day in average).

Incentive-compatibility

Collaboration among IDSs is expected to be a long term relationship. Incentive is important for the long term sustainability of collaborations since it provides motivation for peers to contribute [47, 38]. We compare the average overall cost of all nodes with different FN rates under three different conditions, namely, no collaboration, fixed acquaintances collaboration (acquaintance length (acqlen)=8), and dynamic acquaintance management collaboration. Figure 7.12 shows the distribution of the converged cost of all nodes. We can observe that the cost of all IDSs is much higher when no collaboration is performed in the network. On the other hand, collaborating with random fixed acquaintances can significantly reduce the cost of false decisions, however, the cost of high expertise nodes and low expertise nodes are very close. With our dynamic acquaintance management, high expertise nodes achieve much lower cost than nodes with low expertise, which reflects an incentive design of the collaboration system. Therefore, the system provides motivation for nodes to update their knowledge base and behave truthfully in cooperation.

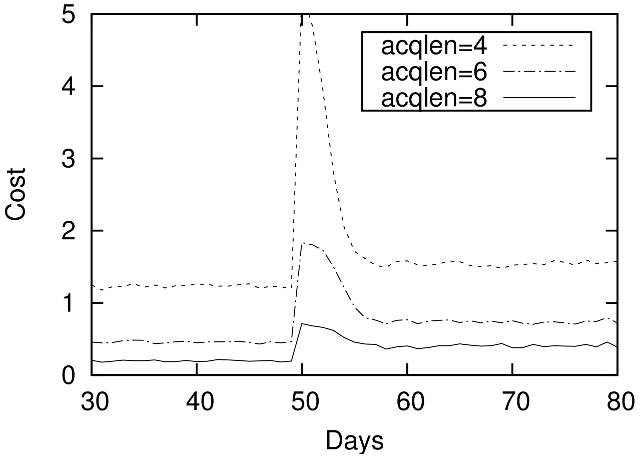


Figure 7.14: The Cost of an IDS under a Betrayal Attack

Robustness

Robustness is a desired property of an IDN since malicious users may try to attack the collaboration mechanism to render it ineffective. In this experiment we focus on the Betrayal attack. To study the impact from one malicious node, we set up a collaboration scenario where IDS$_0$ is collaborating with a group of other IDSs with $FP = 0.1$ and $FN = 0.2$. Among the group, one IDS turns to be dishonest after day 50 and gives false diagnoses. We observe the FP rate and FN rate of this malicious node perceived by IDS$_0$, and the impact on the risk cost of IDS$_0$ under various collaborator group sizes. Figure 7.13 shows the perceived FP and FN rate of the malicious node during each simulation day. We can see that the perceived FP and FN increase fast after day 50. The malicious node is then removed from the acquaintance list of IDS$_0$ when its perceived FP and FN are higher than a predefined threshold. The cost of IDS$_0$ under betrayal attack is depicted in Figure 7.14; we notice that the betrayal behavior introduces a spike of cost increment under all group sizes, but the magnitude of increment decreases when the number of collaborators increases. However, the system can efficiently learn the malicious behavior and recover to normal by excluding malicious nodes from the acquaintance list.

7.6 Conclusion and Future Work

We proposed a statistical model to evaluate the tradeoff between the maintenance cost and intrusion cost, and an effective acquaintance management method to minimize the overall cost for each IDS in an IDN. Specifically, we adopted a Bayesian learning approach to evaluate the accuracy of each IDS in terms of its false positive and true positive rates in detecting intrusions. The Bayes' theorem is applied for the aggregation of feedback provided by the collaborating IDSs. Our acquaintance management explores a list of candidate IDSs and selects acquaintances using an acquaintance selection algorithm. This algorithm is based on a greedy approach to find the smallest number of best acquaintances and minimize the cost of false intrusion decisions and maintenance. The acquaintances list is updated periodically by introducing new candidates which pass the probation period.

Through a simulated IDN environment, we evaluated our Bayesian decision model against threshold-based decision models, and acquaintance selection algorithm against a brute force approach. Compared to the threshold-based model, our Bayesian decision model performs better in terms of cost of false decisions. Compared to the brute force approach, our algorithm achieves similar performance but requires much less computation time. Our acquaintance management is also shown to achieve the desirable properties of convergence, stability, robustness, and incentive-compatibility.

As future work, we plan investigate other more sophisticated attack models on the collaboration mechanism and integrate corresponding defense techniques. Robustness of the acquaintance management system is particularly critical if extended to support IDS peer recommendations. In this case, malicious IDSs may provide untruthful recommendations about other IDSs [110, 122, 83], or worse collude to collaboratively bring the system down.

Chapter 8

Conclusion

Intrusion detection networks (IDNs) are collaboration networks used by intrusion detection systems (IDSs) to exchange information and knowledge, in order to collectively achieve higher intrusion detection accuracy. However, building an IDN is a challenging task. Intrusion detection efficiency, robustness against malicious insiders, incentive compatibility, and scalability are four desired features of IDNs. In this thesis, we have proposed a distributed IDN design (Chapter 3), where IDSs are connected to their collaborators in a peer-to-peer overlay. IDSs in the IDN send consultation messages to their collaborators when they do not have enough information to make a confident intrusion decision, and aggregate the feedback from collaborators to make a final intrusion decision. The proposed IDN architecture includes a number of components essential for IDS collaboration, four of which were developed in detail throughout this thesis, namely, trust management, collaborative intrusion decision, resource management, and acquaintance management.

As part of the trust management component (Chapter 4), we proposed a Dirichlet-based Bayesian learning model to calculate the trust values of collaborators based on past experiences. We showed that this model not only provides an efficient way to estimation trust values, but also provides the confidence levels in trust estimations. In Chapter 5, we modeled the collaborative intrusion decision problem as a Bayes optimization problem. We obtained optimal decision rules that minimize Bayes risks using hypothesis testing methods and provided a data-driven mechanism for real-time efficient, distributed, and sequential feedback aggregation. As part of the resource management component (Chapter 6), a continuous-kernel non-cooperative game model was introduced to solve the problem of fair and incentive-compatible resource allocation. Finally, for the acquaintance management component (Chapter 7), we proposed a statistical model to evaluate the trade-off between the maintenance cost and intrusion cost, and an effective acquaintance management method to minimize the overall cost for each IDS in the network by appropriately selecting acquaintances.

We evaluated the thesis contribution primarily using a simulated IDN considering the desired properties of effective collaborative IDNs previously mentioned. Specifically, we identified a set of metrics to evaluate the performance of the collaboration networks, namely, intrusion detection accuracy, robustness against malicious insiders, incentive-compatibility in resource allocation, and scalability in network size.

The obtained results showed that the proposed IDN design and the IDN architecture components developed in this thesis are indeed efficient, incentive-compatible, scalable and robust. In particular and to evaluate collaborative IDN robustness, we have studied various attack models and corresponding defence mechanisms. Finally though the collaboration management mechanism were developed for intrusion detection networks in this thesis, we believe they can be useful for other types of collaboration networks with untrusted node such as social networks, mobile ad hoc networks, vehicular networks, and sensor networks.

The novelty of this work can be summarized as follows: First, we applied trust evaluation, Bayesian decision, and game theoretical modelling to solve intrusion detection problems; second, we introduced the concept of test messages to evaluate the trustworthiness and expertise levels of participating IDSs; third, we defined a resource-aware collaboration model, where the communication overhead and helping resources between IDSs are constrained and controlled.

As a future work, we plan to contribute to the design of the architecture components not addressed in this thesis, namely the communication overlay component and the mediator. For an extension of the trust management component, it will be interesting to tune some of the model's parameters such as c_1, c_2 (Equaltion 4.1) to evaluate their impact on the performance. A possible extension of the feedback aggregation component is to the optimal aggregation method for correlated feedbacks. For the resource management component, we plan to investigate other types of attacks that malicious insiders can do to game the system, for instance when a node does not follow the predefined resource allocation rules. A possible extension of the acquaintance management component is to study on how to select proper length of the probation period to achieve optimal effect. Moreover, we also would like to investigate more sophisticated collusion attacks and their corresponding defence strategies. As an application of the work presented in this thesis, we are currently working on malware detection using collaborative antiviruses. In this project, file scanning results from different antivirus programs are aggregated to make a more accurate malware detection within acceptable time frame. The ultimate goal of this thesis is to provide guidelines for the deployment of a secure and scalable IDN where effective collaboration can be established between IDSs.

APPENDICES

Appendix A

An Example of IDMEF format

```
<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message xmlns:idmef="http://iana.org/idmef"
version="1.0">
  <idmef:Alert messageid="123456789abc">
    <idmef:Analyzer analyzerid="sensor01">
      <idmef:Node category="dns">
        <idmef:name>sensor.abc.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime
ntpstamp="0xbc89f6f9.0xef669437">2012-09-09T10:01:25.93464Z</
idmef:CreateTime>
    <idmef:Source ident="a0b2" spoofed="yes">
      <idmef:Node ident="a0b2-1">
        <idmef:Address ident="a1a2-2" category="ipv4-addr">
          <idmef:address>192.0.1.100</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Source>
    <idmef:Target ident="b5b6">
      <idmef:Node>
        <idmef:Address ident="b5b6-1" category="ipv4-addr">
          <idmef:address>192.0.1.10</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Target>
    <idmef:Target ident="c7c8">
      <idmef:Node ident="c7c8-1" category="nisplus">
        <idmef:name>hipo</idmef:name>
      </idmef:Node>
    </idmef:Target>
    <idmef:Target ident="d1d2">
      <idmef:Node ident="d1d2-1">
        <idmef:location>Waterloo B10</idmef:location>
        <idmef:name>Cisco.router.b10</idmef:name>
      </idmef:Node>
    </idmef:Target>
    <idmef:Classification text="Ping-of-death detected">
      <idmef:Reference origin="cve">
        <idmef:name>CVE-1999-128</idmef:name>
        <idmef:url>http://www.cve.mitre.org/cgi-bin/cvename.cgi?
name=CVE-1999-128</idmef:url>
      </idmef:Reference>
    </idmef:Classification>
  </idmef:Alert>
</idmef:IDMEF-Message>
```

Figure A.1: Example of intrusion alert in IDMEF format

References

[1] Avast. http://www.avast.com[Last accessed in Feb 15, 2013].

[2] Avira. http://www.avira.com [Last accessed in Feb 15, 2013].

[3] Bots and botnetsa growing threat. http://us.norton.com/botnet/promo [Last accessed in Feb 15, 2013].

[4] Bro. http://www.bro-ids.org/[Last accessed in Feb 15, 2013].

[5] CAIDA: The Cooperative Association for Internet Data Analysis. http://www.caida.org [Last accessed in Feb 15, 2013].

[6] Common intrusion detection signatures standard. http://tools.ietf.org/html/draft-wierzbicki-cidss-04 [Last accessed in Feb 15, 2013].

[7] Faster Actions Needed Against Phishing Domains. http://news.netcraft.com /archives/2009/06/22/faster_actions_needed_against_phishing_domains.html [Last accessed in Feb 15, 2013].

[8] Fksensor. "http://www.keyfocus.net/kfsensor/download" [Last accessed in Feb 15, 2013].

[9] Honeyd. "http://www.honeyd.org" [Last accessed in Feb 15, 2013].

[10] Intrusion detection message exchange format. http://www.ietf.org/rfc/rfc4765.txt [Last accessed in Feb 15, 2013].

[11] myNetWatchman. http://www.mynetwatchman.com [Last accessed in Feb 15, 2013].

[12] National vulnerability Database. http://nvd.nist.gov [Last accessed in Feb 15, 2013].

[13] OSSEC. http://www.ossec.net/[Last accessed in Feb 15, 2013].

[14] Protecting Against the Rampant Conficker Worm. http://www.pcworld.com/article/157876/protecting_against_the_rampant _conficker_worm.html [Last accessed in Feb 15, 2013].

[15] Request for comments. http://newrfc.itms.pl/?mod=yes&range=4765 [Last accessed in Feb 15, 2013].

[16] SANS Internet Storm Center (ISC). http://isc.sans.org/ [Last accessed in Feb 15, 2013].

[17] Snort. http://www.snort.org/[Last accessed in Feb 15, 2013].

[18] Spector. "http://www.specter.com" [Last accessed in Feb 15, 2013].

[19] Symantec. http://www.symantec.com/ [Last accessed in Feb 15, 2013].

[20] The Honeynet Project. http://www.honeynet.org/[Last accessed in Feb 15, 2013].

[21] TripWire. http://www.tripwire.com/[Last accessed in Feb 15, 2013].

[22] US-CERT. http://www.kb.cert.org [Last accessed in Feb 15, 2013].

[23] What is SmartScreen Filter? http://www.microsoft.com/security/filters/smartscreen.aspx [Last accessed in Feb 15, 2013].

[24] ZDnet. http://www.zdnet.com/blog/security/confickers-estimated-economic-cost-91-billion/3207 [Last accessed in Feb 15, 2013].

[25] The honeynet project. know your enemy: Fast-flux service networks, 13 July, 2007. http://www.honeynet.org/book/export/html/130[Last accessed in Feb 15, 2013].

[26] Apples app store downloads top three billion, 2010. http://www.apple.com/pr/library/2010/01/05Apples-App-Store-Downloads-Top-Three-Billion.html [Last accessed in Feb 15, 2013].

[27] M. Ahamad, D. Amster, M. Barrett, T. Cross, G. Heron, D. Jackson, J. King, W. Lee, R. Naraine, G. Ollmann, et al. Emerging cyber threats report for 2009. 2008.

[28] Ehab Al-Shaer, Hazem Hamed, Raouf Boutaba, and M. Hasan. Conflict classification and analysis of distributed firewall policies. *IEEE Journal on Selected Areas in Communications*, 23(10):2069–2084, 2005.

[29] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. SIAM, Philadelphia, 2nd edition, 1999.

[30] A. Berman and R.J. Plemmons. *Nonnegative Matrices in Mathematical Sciences*. SIAM, 1994.

[31] D. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[32] Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2003.

[33] K. Boudaoud, H. Labiod, R. Boutaba, and Z. Guessoum. Network security management with intelligent agents. In *Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP*, pages 579 –592, 2000.

[34] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[35] M. Cai, K. Hwang, Y.K. Kwok, S. Song, and Y. Chen. Collaborative internet worm containment. *IEEE Security & Privacy*, 3(3):25–33, 2005.

[36] F. Cuppens and A. Miege. Alert correlation in a cooperative intrusion detection framework. In *2002 IEEE Symposium on Security and Privacy, 2002. Proceedings*, 2002.

[37] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen. Honeystat: Local worm detection using honeypots. *Lecture Notes in Computer Science*, pages 39–58, 2004.

[38] A. Dal Forno and U. Merlone. Incentives and Individual Motivation in Supervised Work Groups. *European Journal of Operational Research*, 207(2):878–885, 2010.

[39] D. Dash, B. Kveton, J.M. Agosta, E. Schooler, J. Chandrashekar, A. Bachrach, and A. Newman. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In *Proceedings of the national conference on Artificial Intelligence*, volume 21, page 1115. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[40] N. Daswani, C. Kern, and A. Kesavan. *Foundations of Security: What Every Programmer Needs to Know*. Dreamtech Press, 2007.

[41] Chris Davies. iphone spyware debated as app library phones home, 2009. "http://offerpia.com/won/link/?item_no=23887" [Last accessed in Feb 15, 2013].

[42] H. Debar, M. Becker, and D. Siboni. A neural network component for an intrusion detection system. In *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*, pages 240–250. IEEE, 1992.

[43] D.E. Denning. An intrusion-detection model. *Software Engineering, IEEE Transactions on*, (2):222–232, 1987.

[44] J.R. Douceur. The sybil attack. *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, 2002.

[45] C. Duma, M. Karresand, N. Shahmehri, and G. Caronni. A trust-aware, p2p-based overlay for intrusion detection. In *International Conference on Database and Expert Systems Applications*, 2006.

[46] Adel El-Atawy, Ehab Al-Shaer, Tung Tran, and Raouf Boutaba. Adaptive early packet filtering for defending firewalls against dos attacks. In *INFOCOM 2009, IEEE*, pages 2437–2445. IEEE Press, 2009.

[47] E. Fehr and H. Gintis. Human Motivation and Social Cooperation: Experimental and Analytical Foundations. *Annu. Rev. Sociol.*, 33:43–64, 2007.

[48] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. *Selected Areas in Communications, IEEE Journal on*, 24(5):1010–1019, 2006.

[49] Michael Fitzpatrick. Mobile that allows bosses to snoop on staff developed, 2010. ”http://news.bbc.co.uk/2/hi/8559683.stm” [Last accessed in Feb 15, 2013].

[50] M. Fossi, G. Egan, K. Haley, E. Johnson, T. Mack, T. Adams, J. Blackbird, M.K. Low, D. Mazurek, D. McKinney, et al. Symantec internet security threat report trends for 2010. *Volume XVI*, 2011.

[51] M. Fossi, E. Johnson, D. Turner, T. Mack, J. Blackbird, D. McKinney, M.K. Low, T. Adams, M.P. Laucht, and J. Gough. Symantec report on the underground economy: July 2007 to june 2008. Technical report, Technical Report, Symantec Corporation, 2008.

[52] C. Fung, Q. Zhu, R. Boutaba, and T. Başar. Poster: SMURFEN: A rule sharing collaborative intrusion detection network. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pages 761–764, 2011.

[53] Carol J Fung, Jie Zhang, Issam Aib, and Raouf Boutaba. Dirichlet-based trust management for effective collaborative intrusion detection networks. *Network and Service Management, IEEE Transactions on*, 8(2):79 –91, june 2011.

[54] Carol J Fung, Jie Zhang, and Raouf Boutaba. Effective Acquaintance Management for Collaborative Intrusion Detection Networks. In *16th International Conference on Network and Service Management (CNSM 2010)*, 2010.

[55] Carol J Fung, Jie Zhang, and Raouf Boutaba. Effective acquaintance management based on bayesian learning for distributed intrusion detection networks. *Network and Service Management, IEEE Transactions on*, 9(3):320–332, Sept. 2012.

[56] C.J. Fung, O. Baysal, J. Zhang, I. Aib, and R. Boutaba. Trust management for host-based collaborative intrusion detection. In *19th IFIP/IEEE International Workshop on Distributed Systems*, 2008.

[57] C.J. Fung, J. Zhang, I. Aib, and R. Boutaba. Robust and scalable trust management for collaborative intrusion detection. In *Proceedings of the Eleventh IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2009.

[58] C.J. Fung, Q. Zhu, R. Boutaba, and T. Barsar. Bayesian Decision Aggregation in Collaborative Intrusion Detection Networks. In *12th IEEE/IFIP Network Operations and Management Symposium (NOMS10)*, 2010.

[59] A. Gelman. *Bayesian data analysis*. CRC press, 2004.

[60] A. Ghosh and S. Sen. Agent-based distributed intrusion alert system. In *Proceedings of the 6th International Workshop on Distributed Computing (IWDC04)*. Springer, 2004.

[61] S.J. Grossman and O.D. Hart. Takeover bids, the free-rider problem, and the theory of the corporation. *The Bell Journal of Economics*, pages 42–64, 1980.

[62] C. Grothoff. An excess-based economic model for resource allocation in peer-to-peer networks. *Wirtschaftsinformatik*, 45(3):285–292, 2003.

[63] Graham Hamed, E., A. J., Elmaghraby, and D. Ingram. Doorknob rattling attack detection using a distributed agent-based intrusion detection system. In *Third Annual International Systems Security Engineering Conference*, pages 7:1–7:11, 2002.

[64] M.T.T. Hsiao and A.A. Lazar. Optimal decentralized flow control of Markovian queueing networks with multiple controllers. *Performance Evaluation*, 13(3):181–204, 1991.

[65] F. IAO. ”lloveyou” virus lessons learned report. 2003.

[66] Gartner Inc. Gartner survey shows phishing attacks escalated in 2007; more than 3 billion lost to these attacks. Press release, 2007.

[67] R.W. Janakiraman and M.Q. Zhang. Indra: a peer-to-peer approach to network intrusion detection and prevention. *WET ICE 2003. Proceedings of the 12th IEEE International Workshops on Enabling Technologies*, 2003.

[68] T. Jiang and J.S. Baras. Trust evaluation in anarchy: A case study on autonomous networks. In *INFOCOM*. IEEE, 2006.

[69] Audun Josang and Jochen Haller. Dirichlet reputation systems. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 112–119. IEEE, 2007.

[70] Audun Jøsang and Roslan Ismail. The Beta Reputation System. In *Proceedings of the Fifteenth Bled Electronic Commerce Conference*, 2002.

[71] J.H. Keppler and H. Mountford. *Handbook of Incentive Measures for Biodiversity: Design and Implementation.* OECD, 1999.

[72] Y.A. Korilis and A.A. Lazar. On the existence of equilibria in noncooperative optimal flow control. *Journal of the ACM (JACM)*, 42(3):584–613, 1995.

[73] B. C. Levy. *Principles of Signal Detection and Parameter Estimation.* Springer-Verlag, 2008.

[74] Z. Li, Y. Chen, and A. Beach. Towards scalable and robust distributed intrusion alert fusion with good load balancing. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, pages 115–122. ACM New York, NY, USA, 2006.

[75] Z. Li, Y. Chen, and A. Beach. Towards scalable and robust distributed intrusion alert fusion with good load balancing. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense.* ACM, 2006.

[76] ME Locasto, JJ Parekh, AD Keromytis, and SJ Stolfo. Towards collaborative security and P2P intrusion detection. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 333–339, 2005.

[77] T.F. Lunt, D.E. Denning, R.R. Schell, M. Heckman, and W.R. Shockley. The seaview security model. *Software Engineering, IEEE Transactions on*, 16(6):593–607, 1990.

[78] R.T.B. Ma, S.C.M. Lee, J.C.S. Lui, and D.K.Y. Yau. A game theoretic approach to provide incentive and service differentiation in P2P networks. In *Sigmetrics/Performance*, 2004.

[79] Stephen Marsh. *Formalising Trust as a Computational Concept.* Ph.D. thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[80] L. Mekouar, Y. Iraqi, and R. Boutaba. A reputation management and selection advisor schemes for peer-to-peer systems. In *In Proceedings of the 15th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM)*, 2004.

[81] L. Mekouar, Y. Iraqi, and R. Boutaba. Detecting malicious peers in a reputation-based peer-to-peer system. In *Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE*, pages 37 – 42, 2005.

[82] L. Mekouar, Y. Iraqi, and R. Boutaba. Peer-to-peers most wanted: malicious peers. *Computer Networks*, 50(4):545–562, 2006.

[83] L. Mekouar, Y. Iraqi, and R. Boutaba. A Recommender Scheme for Peer-to-Peer Systems. In *International Symposium on Applications and the Internet (SAINT)*. IEEE, 2008.

[84] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking (ToN)*, 8(5):556–567, 2000.

[85] D. Moore, C. Shannon, D.J. Brown, G.M. Voelker, and S. Savage. Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, 2006.

[86] Dan Moren. Retrievable iphone numbers mean potential privacy issues, September 2009. http://www.macworld.com/article/1143047/phone_hole.html [Last accessed in Feb 15, 2013].

[87] Atlas Arbor Networks, 2008. http://atlas.arbor.net/ [Last accessed in Feb 15, 2013].

[88] K.C. Nguyen, T. Alpcan, and T. Başar. A decentralized Bayesian attack detection algorithm for network security. In *Proceedings of the 23rd International Information Security Conference*, 2005.

[89] J. Oberheide, E. Cooke, and F. Jahanian. Cloudav: N-version antivirus in the network cloud. In *Proceedings of the 17th USENIX Security Symposium*, 2008.

[90] Pandalabs. Annual report Panda labs 2010. http://press.pandasecurity.com/wp-content/uploads/2010/05/PandaLabs-Annual-Report-2010.pdf.

[91] AGP Rahbar and O. Yang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4):460–473, 2007.

[92] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Commun. ACM*, 43(12):45–48, 2000.

[93] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood. The value of reputation on eBay: A controlled experiment. *Experimental Economics*, 9(2):79–101, 2006.

[94] R. Richardson. 15th annual 2010/2011 computer crime and security survey. *Computer Security Institute, New York, NY*, 2011.

[95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Second Edition, Prentice Hall, Englewood Cliffs, New Jersey, 2002.

[96] Jordi Sabater and Carles Sierra. Regret: A reputation model for gregarious societies. In *Proceedings of the Fifth International Conference on Autonomous Agents Workshop on Deception, Fraud and Trust in Agent Societies*, 2001.

[97] P. Sen, N. Chaki, and R. Chaki. HIDS: Honesty-rate based collaborative intrusion detection system for mobile ad-hoc networks. *Computer Information Systems and Industrial Management Applications. CISIM'08.*, pages 121–126, 2008.

[98] T.C. Shelling. *The Strategy of Conflict.* Harvard University Press, 1980.

[99] L. Spitzner. Honeypots: definitions and value of honeypots. *Available from: www. tracking-hackers. com/papers/honeypots. html*, 2003.

[100] R. Srikant. *The Mathematics of Internet Congestion Control.* Birkhäuser, 2004.

[101] 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*, 2005.

[102] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[103] Y.L. Sun, Z. Han, W. Yu, and K.J.R. Liu. A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks. In *INFOCOM.* IEEE, 2006.

[104] Matthew Tanase. Ip spoofing: an introduction. *Security Focus*, 11, 2003.

[105] W. T. Luke Teacy, Jigar Patel, Nicholas R. Jennings, and Michael Luck. Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In *Proceedings of Fourth International Autonomous Agents and Multiagent Systems (AAMAS)*, 2005.

[106] Thomas Tran and Robin Cohen. Improving user satisfaction in agent-based electronic marketplaces by reputation modeling and adjustable product quality. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2004.

[107] J.N. Tsitsiklis. Decentralized detection. *Advances in Statistical Signal Processing*, pages 297–344, 1993.

[108] D. Turner, M. Fossi, E. Johnson, T. Mack, J. Blackbird, S. Entwisle, M.K. Low, D. McKinney, and C. Wueest. Symantec global internet security threat report–trends for july-december 07. *Symantec Enterprise Security*, 13:1–36, 2008.

[109] J. Ullrich. DShield. http://www.dshield.org/indexd.html [Last accessed in Feb 15, 2013].

[110] Pedro B. Velloso, Rafael P. Laufer, Daniel de O. Cunha, Otto Carlos M. B. Duarte, and Guy Pujolle. Trust Management in Mobile Ad Hoc Networks Using a Scalable Maturity-Based Model. *IEEE Transactions on Network and Service Management (TNSM)*, 7(3):172–185, 2010.

[111] R. Vogt, J. Aycock, and M. Jacobson. Army of botnets. In *ISOC Symp. on Network and Distributed Systems Security*, 2007.

[112] A. Wald. *Sequential Analysis.* John Wiley and Sons, 1947.

[113] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A taxonomy of computer worms. In *Proceedings of the 2003 ACM workshop on Rapid Malcode*, pages 11–18. ACM New York, NY, USA, 2003.

[114] Andrew Whitby, Audun Jøsang, and Jadwiga Indulska. Filtering out unfair ratings in bayesian reputation systems. *The Icfain Journal of Management Research*, pages 48–64, February 2005.

[115] Paul Wood, Mathew Nisbet, Gerry Egan, Nicholas Johnston, Kevin Haley, Bhaskar Krishnappa, Tuan-Khanh Tran, Irfan Asrar, Orla Cox, Sean Hittel, et al. Symantec internet security threat report trends for 2011. *Volume XVII*, 2012.

[116] Y.S. Wu, B. Foo, Y. Mei, and S. Bagchi. Collaborative intrusion detection system (CIDS): a framework for accurate and efficient IDS. In *Proc. of 19th Annual Computer Security Applications Conference*, 2003.

[117] H. Man Y. Liu, C. Comaniciu. A Bayesian game approach for intrusion detection in wireless ad hoc networks. *Valuetools*, October 2006.

[118] Y. Yan, A. El-Atawy, and E. Al-Shaer. Ranking-based optimal resource allocation in peer-to-peer networks. In *Proc. of the 26th annual IEEE conference on computer communications (IEEE INFOCOM 2007), May*, 2007.

[119] V. Yegneswaran, P. Barford, and S. Jha. Global Intrusion Detection in the DOMINO Overlay System. In *Proc. of Network and Distributed System Security Symposium (NDSS04)*, 2004.

[120] Vinod Yegneswaran, Paul Barford, and Somesh Jha. Global intrusion detection in the domino overlay system. In *In Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2004.

[121] B. Yu and M.P. Singh. Detecting deception in reputation management. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003.

[122] Jie Zhang and Robin Cohen. Trusting advice from other buyers in e-marketplaces: the problem of unfair ratings. In *Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet*, 2006.

[123] Y. Zhang and Y. Fang. A fine-grained reputation system for reliable service selection in peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 1134–1145, 2007.

[124] Z. Zhong, L. Ramaswamy, and K. Li. ALPACAS: A Large-scale Privacy-Aware Collaborative Anti-spam System. In *Proc. IEEE INFOCOM*, 2008.

[125] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. Collaborative detection of fast flux phishing domains. *Journal of Networks*, 4:75–84, February 2009.

[126] Chenfeng Vincent Zhou, Christopher Leckie, Shanika Karunasekera, and Tao Peng. A self-healing, self-protecting collaborative intrusion detection architecture to trace-back fast-flux phishing domains. In *The 2nd IEEE Workshop on Autonomic Communication and Network Management (ACNM 2008)*, April 2008.

[127] C.V. Zhou, S. Karunasekera, and C. Leckie. A peer-to-peer collaborative intrusion detection system. In *Proceedings of the IEEE International Conference on Networks*, pages 118–123, November 2005.

[128] Q. Zhu and T. Başar. Indices of power in optimal ids default configuration: theory and examples. In *Proc. of 2nd Conference on Decision and Game Theory for Security (GameSec 2011), College Park, MD, USA.*, November 2011.

[129] Q. Zhu, C.J. Fung, R. Boutaba, and T. Barsar. A Distributed Sequential Algorithm for Collaborative Intrusion Detection Networks. In *IEEE International Conference on Communications (ICC2010)*, 2009.

[130] Q. Zhu and L. Pavel. Enabling osnr service differentiation using generalized model in optical networks. *IEEE Transactions on Communications*, 57(9):2570–2575, September 2009.

[131] Quanyan Zhu, Carol Fung, Raouf Boutaba, and Tamer Başar. A game-theoretic approach to knowledge sharing in distributed collaborative intrusion detection networks: Fairness, incentives and security. In *Proc. of the 50th IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC), Orlando, USA*, December 2011.

[132] Quanyan Zhu, Carol Fung, Raouf Boutaba, and Tamer Başar. GUIDEX: A game-theoretic incentive-based mechanism for intrusion detection networks. *IEEE Journal on Selected Areas in Communications (JSAC) Special Issue on Economics of Communication Networks & Systems, to appear*, 2012.

[133] Quanyan Zhu, Carol Fung, Raouf Boutaba, and Tamer Başar. A game-theoretical approach to incentive design in collaborative intrusion detection networks. In *Proceedings of the International Symposium on Game Theory for Networks (GameNets)*, May, 2009.

[134] Quanyan Zhu and Lacra Pavel. End-to-end DWDM optical link power-control via a Stackelberg revenue-maximizing model. *Int. J. Netw. Manag.*, 18(6):505–520, November 2008.