Toward Email Archive Intrusion Detection

By

Yiru Li

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfilment of

the requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario, Canada

December 2005

The undersigned hereby recommend to

the Faculty of Graduate Studies and Research

acceptance of the thesis,

Toward Email Archive Intrusion Detection

submitted by

Yiru Li

Dr. Douglas Howe

(Director, School of Computer Science)

Dr. Anil Somayaji

(Thesis Supervisor)

Carleton University

December 2005

Abstract

Online email archives can store years worth of sensitive personal and business information. However, the standard authentication mechanism used by most email archives, reusable text passwords, is weak and can easily be compromised. To protect such archives, I propose a novel user-specific design for an anomaly-based email archive intrusion detection system. The design contains two parts—user-tailored modelling and user-involved alarm response. As a first step towards building such a system, I have developed a simple probabilistic model of user email behavior that correlates email senders and users' dispositions of email messages. In tests using data gathered from three months of observed user behavior and synthetic models of attacker behavior, this model exhibits a low rate of false positives (generally one false alarm every few weeks) while still detecting most attacks. These results suggest that anomaly detection is a feasible strategy for securing email archives, one that does not require changes in user authentication or access patterns.

Acknowledgements

I gratefully acknowledge the insightful feedback, constructive suggestions, corrections, patience and encouragement given by my supervisor Anil Somayaji. As well, I would like to thank Hajime Inoue for his suggestions and corrections and Glenn Wurster, and Julie Thorpe for their support on data collection. I also would like to thank Carlisle Adams and Paul Van Oorschot for serving on my committee and Tony White for chairing my defense. I also would like to thank MITACS and NSERC for funding this work. Finally, I would like to thank my parents and sister for their endless pushing and love. I guess that my mom would say ”my god, finally...” when I tell her that my master's study is finished.

Contents

Chapter 1

Introduction

Email is a very popular Internet based communication service which millions use every day. However, email has many security issues, one of which is that centrally-managed email message stores (called email archives) are illegitimately accessed by masqueraders. Two kinds of approaches can be used to avoid such attacks: authentication and intrusion detection systems. The main authentication method used by email—reusable text passwords—can be easily compromised by many means such as shoulder surfing and users' negligence. This thesis examines whether intrusion detection technology can be used to compensate for authentication limitations without requiring additional hardware or user interface changes. Before I present how I use intrusion detection technology to secure email archives, I first introduce the email system.

1.1 Internet Email Service System

Email is one of the most important Internet applications. Like the world wide web, it has a client/server architecture. Email clients communicate with email servers on behalf of users to send and retrieve messages. Email servers then transport and store email.

Email is convenient, easy to use and asynchronous. Email is convenient because email users can access their email messages from any Internet-connected computer. In addition, the email system is easy to use for most users, requiring no special skills or knowledge. All users need to know is how to click buttons in mail clients to select operations such as sending, receiving and composing email messages. Finally, like the post office system, email service is asynchronous. Instead of waiting for replies from receivers about whether messages have successfully arrived, email users just send messages and leave. Email servers are responsible for delivering the messages to final destinations. It is these three features that have made email a popular communication medium.

Email is in use by millions every day. Virtually every ISP provides email services, and many companies offer free email services. The most famous free email service providers, such as Hotmail [42] and Gmail [24], offer not only large free storage space but also other services like spam filtering and virus scanning. According to email marketing report released by eMarketer [23], email message volume in the United States is projected to nearly double between 2003 and 2007; in the US alone, 88% of adult Internet users have personal email accounts; further, 46% of them have email access at work. Email has become a familiar part of our daily routine.

However, email has many security issues. Email servers and clients are attacked by viruses and worms due to software vulnerabilities. Email messages are intercepted in transmission due to the open nature of the Internet. Received email messages are illegitimately accessed by malicious software (e.g., viruses) or by attackers due to the weakness of authentication and storage mechanisms. Email messages are sent in the name of other senders due to lack of sender authentication (spoofing and phishing). Unsolicited commercial email (spam) floods to user mailboxes.

As discussed in Chapter 2, many of these email security issues have corresponding practical solutions. However, existing protections do not address the problem of inside attacks to email archives.

1.2 Email Archive And Inside Attacker

A user's received email messages can be stored either locally in her own computer or centrally in a server-based mail store. The centralized message storage has advantages over locally stored email. The key ones are that users can access centrally-stored messages (both old and new messages) from any Internet-connected computer and do not need to remember which machine a message has been downloaded to. Therefore, centralized storage has become very popular and is used by many email service providers such as Yahoo, Google and Microsoft. In my work, I refer to these collections of email messages which are centrally stored on the server side as email archives.

Email archives are becoming tempting targets of attackers. The key motives of email attackers are to obtain sensitive user information such as credit card numbers and business correspondence. A user's email archive can contain years worth of valuable messages which are sorted and filtered by users. In contrast to network-captured messages which may contain a large amount of junk mail (spam), a significant portion of the information in email archives is important for users, and thus are also valuable for attackers. In addition, in email archives, attackers can access email from the past, from before the time the attackers decided to target the users. Therefore, attackers can learn about users quickly and completely from email archives. Further, compared with other means of obtaining email messages (e.g., intercepting messages from networks), it requires much less technical knowledge for attackers to access a user's email archive. Once attackers obtain the passwords of target users, attackers can read, modify and delete email as easily as legitimate users. In terms of risk, attackers remotely accessing users' email archives from Internet-connected computers are much safer than those directly accessing the users' local machines. Therefore, it is attractive for attackers to break into users' email archives.

Normally there are two approaches which may be used by an attacker to obtain messages in a user's email archive—one is that the attacker exploits vulnerabilities of software associated with email archives (e.g., the IMAP server); the other is that he compromises the user's authentication credentials.

The latter type is one kind of insider attack[1] and the corresponding attackers are defined as "inside attackers" in my work. I am mainly concerned about the kind of insider attacker who knows the user and makes use of specific knowledge of the user to compromise the user's authentication credentials. The attacker might be the user's friend, colleague, business competitor or family member. If he is the user's competitor, he might want to know something about the user's business. If he is the user's spouse, he might want to know whether the user is having an affair. As the attacker probably wants to monitor the user over the long term, he might act carefully to avoid being noticed, e.g., by erasing any evidence of his activities. As a result, it will be hard for the user herself to discover the insider attacker.

Such insider attacks primarily result from authentication failures. Unfortunately, the most commonly used authentication credentials in the email archive domain, reusable text passwords, are extremely vulnerable due to common patterns of user behaviors. Many users choose simple passwords that are easy to remember; many such passwords, however, can be compromised by online and offline dictionary attacks. Users enter passwords on untrusted machines that may be infected with viruses, spyware, or other malicious software. Such malware can be used to capture passwords. Also, users often share passwords across domains and applications, allowing one weak application (e.g., one that sends passwords in the clear) to result in the compromise

[1]Misuse of legitimate credentials is the other kind of inside attack, where legitimate users do something inappropriate with their privileges. The two are indistinguishable from computers' point of view.

of other, more secure systems. Additionally, users often reveal passwords to friends, family members, and co-workers—sometimes inadvertently, but sometimes to facilitate the sharing of information or resources. Those very same "insiders," however, often have motive for compromising a user's privacy.

Besides reusable text passwords, there exist other novel authentication methods such as smart cards [9], biometric identification [47] and graphical passwords [32]. Compared with reusable text passwords, these novel authentication methods are more difficult to compromise by attackers. However, they either require extra hardware, or are computationally expensive, or offer unfamiliar, less user-friendly interfaces. These disadvantages make these authentication methods impractical for email service, which has millions of users. This large installed base makes a new authentication method difficult to adopt.

However, fortunately, we have another potential technology, intrusion detection, which can be used to detect insider attacks[2] as a complement to the authentication method of text passwords. Intrusion detection systems can have the same goal as authentication systems: to verify that a user is the one that she claims to be. However, these two kinds of systems work at different stages: authentication systems working when attackers request access, and IDSs working after the attackers have obtained access. In terms of recognition methods, authentication systems verify a user by checking her credentials while IDSs verify by checking her behavior patterns. From the point of view of users, IDSs are much less visible to users than authentication systems which directly interface with users. Therefore, an IDS with acceptable computation cost for machines and acceptable effectiveness for users can supplement an authentication system in terms of verifying users.

[2]Besides insider attacks, intrusion detection systems (IDSs) can also be used to detect other kinds of attacks such as viruses, worms and back door trojans.

1.3 Contributions

My work makes several concrete contributions.

First, based on the characteristics of the email archive domain, I describe my proposal for a user-specific email archive IDS. The design includes two parts: user-tailored modelling and user-involved alarm response. With the design, the system would customize a model to a user according to the user's behavior patterns and security needs. In addition, instead of sending alarms to administrators or security officers, the system would send alarms to the corresponding users to verify whether the alarms are true or false. So long as the data acquisition and analysis requirements of such a system are sufficiently small, such an architecture could potentially scale up to the largest email sites—even with a fixed per-user rate of false positives. False positives are defined as the rates that systems reject legitimate users.

Second, as a first step towards a user-specific email archive IDS, I develop and test a simple statistical model of user email behaviors based upon the relationship between dispositions of new received messages and the senders of these messages. Based on the design of user-tailored modelling, a user's behavior model is customized at the parameter level. After training for approximately one month, the system can distinguish between variations in user behaviors over the next two months and simulated attacker behaviors with a low rate of false positives—as low as one alarm per month, but generally not higher than one per week.

Third, I emphasize the importance of systematically designing an attack behavior model, which is used for assessment of my user model. Rather than randomly choosing a type of attack behavior, I systematically design fourteen attack behavior models, and based upon experiments and domain-specific knowledge I choose the most appropriate one from the fourteen attack models. Because of the method I use to simulate attack behaviors, the experimental results of my user model can be more trustworthy in terms of false positives and true positives than those anomaly IDSs which are tested using carelessly simulated attack behaviors[3].

Fourth, I test how different values of each parameter affect each user's model. I demonstrate that parameter values can influence effectiveness of models, and therefore the parameters of every user's model should be dynamically adjusted.

1.4 Overview

The rest of this thesis proceeds as follows. Chapter 2 presents background information and related work. Chapter 3 describes the user-specific design of an email archive IDS. Chapter 4 discusses my choice of observables and modelling strategy. Chapter 5 presents the implementation of data collection and data processing. Experimental results are discussed in Chapter 6. This thesis ends in Chapter 7 with a discussion of limitations, implications, and plans for future work.

[3]For example, some user-behavior anomaly IDSs use other users' data as attack data to test a user's model such as [37].

Chapter 2

Related Work and Background

My work can be characterized as detecting email archive insider attacks by modelling user behaviors. Therefore, I first introduce the background knowledge about the email framework and email security issues respectively in Section 2.1 and 2.2. Then I give a brief overview of the intrusion detection research field in Section 2.3. Later, I describe in detail about user behavior-level anomaly IDSs in Section 2.4. Finally, I summarize the related research work and explain the key differences between my work and that of others.

2.1 Background of Email System Framework

The email system has the same asynchronous structure as the traditional mail system (post office system)—storing and forwarding.

In the traditional mail system, a customer writes a letter, puts it in an envelope with a destination address, and brings it to a post office nearby. The letter is stored there for a short while and then it is forwarded to other intermediate post offices. Finally, the letter arrives at the post office closest to the destination. Either the letter is directly sent from there to the destination or the receiver goes to the post

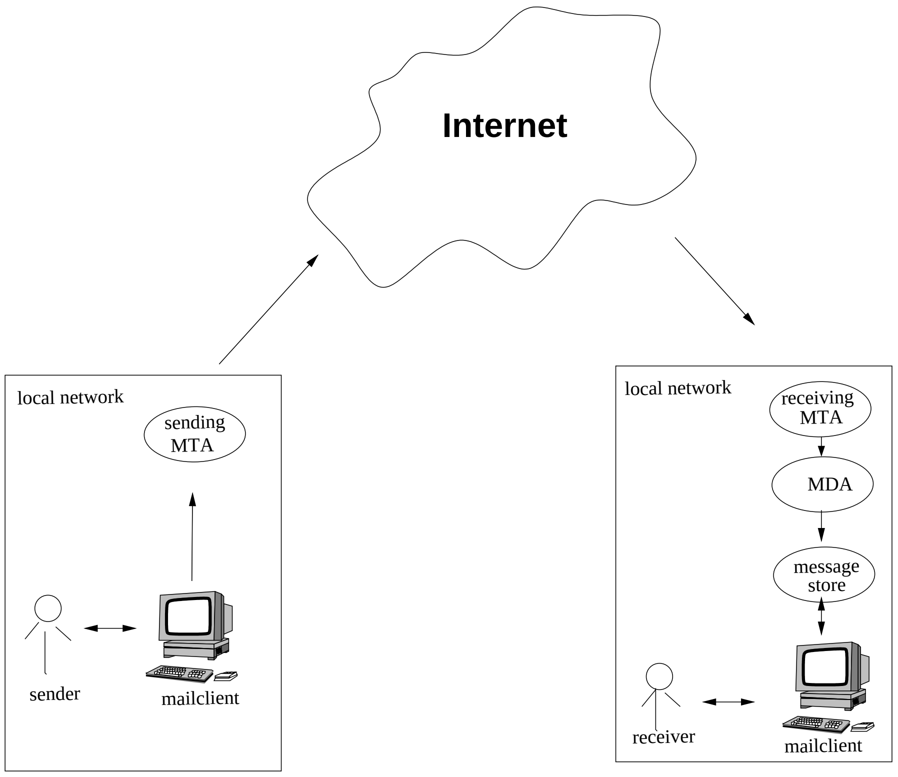

Figure 2.1: Internet Mail Service Framework.

office to pick it up herself with her identification.

The email system (as shown in Figure 2.1) primarily consists of mail user agents (MUAs), mail transport agents (MTAs), mail delivery agents (MDAs) and mail stores. A MUA (e.g., Microsoft Outlook [41], Mozilla Thunderbird [4]) is used to communicate with the email system on behalf of users. A MUA can send a new message upstream to a MTA and retrieve a message from a mail store. Using transportation protocols (e.g., SMTP), a MTA (e.g., Sendmail [8]) routes a message to destination, giving it to other intermediate MTAs if and when necessary. A MDA (e.g., procmail [7]) is passed a newly received message from the destination domain MTA and puts the message in a mail store. A mail store is used to temporarily or permanently store users' email archives.

The architecture of the email system is very similar to the traditional mail system. An email user composes a new message in a MUA. Then the MUA places the message within an “envelope” with the destination email address provided by the user and sends it to a MTA in the local network to which the user belongs. The MTA forwards the message to other intermediate MTAs. Finally, the email message arrives at the local network to which the email receiver belongs and is stored in the receiver’s mail box located in a message store. When the receiver accesses her mail box, she will find it.

Email messages can be retrieved either locally (such as directly opening a file in /var/mail) or remotely. There are two standard mail access protocols used for remote email access: Internet Mail Access Protocol (IMAP [18]) and Post Office Protocol (POP [43]).

POP is a rather simple protocol. It does not keep track of message states, and it does not even facilitate the storing of messages in a number of folders. It does one thing and only one thing: it makes the messages available to the user to download to her local machine.

IMAP is not only a way to retrieve messages from a central mail store, but also a way to manage messages centrally. The messages can remain in the mail store all the time without ever having to download to local machines. You can see the advantages of IMAP very clearly if you work from several computers. With IMAP, you don’t have to wonder which computer you were on when you downloaded and read a given message; you know it is still in the mail store. Figure 2.2 is an example about how an IMAP server, an IMAP client (part of a MUA) and a user communicate with each other. The user selects operations (e.g., reading a message, deleting a message, creating a folder), the IMAP client sends the corresponding IMAP request commands to the IMAP server, and the IMAP server performs actual operations on the user’s email archive. My work is to protect email messages in the mail store from being illegitimately accessed by monitoring IMAP commands in the IMAP server.

Alternatively, an email user can access her email archive through a web mail server,

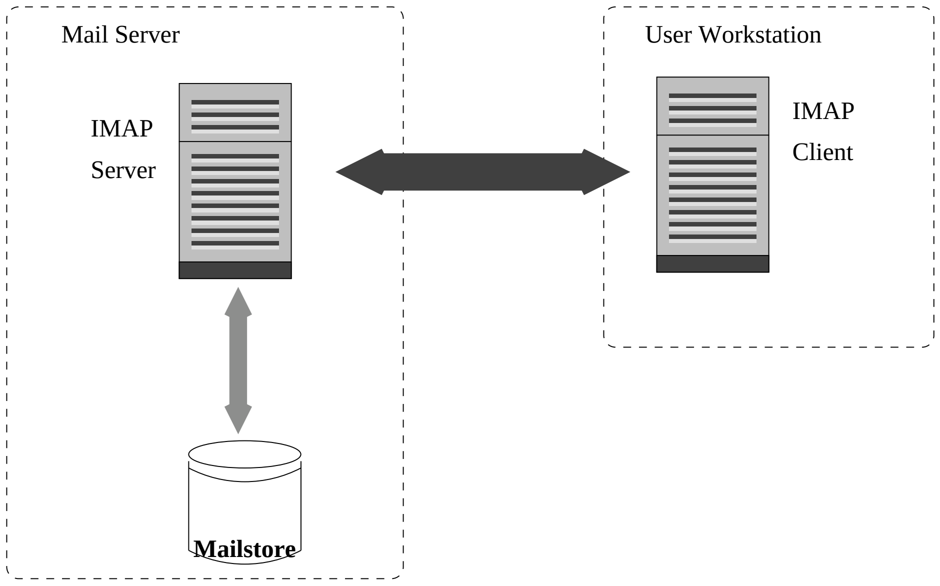

Figure 2.2: IMAP server and IMAP client.

in which a web browser is used as an interface instead of a stand-alone mail client program. Besides the advantages of a friendly, user-familiar interface when compared with an email client program, web mail is also accessible on any computer with no configuration and special software. Web mail is extremely popular and is offered by service providers including Yahoo, Google and Microsoft.

2.2 Overview of Email Security Issues

Email is easy to use, convenient and popular, but it is very insecure. There are two common email security issues: unwanted email, and email messages being illegitimately read and modified.

Unwanted email. Unwanted email includes spam, phishing and email viruses. Email spam is a general term for unwanted junk email. It typically includes advertisements (unsolicited commercial email or UCE) or other messages sent in bulk to many recipients (unsolicited bulk email or UBE). Email phishing is the act of attempting to fraudulently acquire sensitive information, such as passwords and credit cards, by imitating legitimate email. Email viruses are the kind of viruses which generally propagate via email attachments or something equivalent to a HTML page in the text body.

In terms of threat, spam is less destructive than phishing and email viruses. However, all of them can use the same techniques to obtain recipients' email addresses such as guessing addresses by dictionary attack or using spyware to steal addresses from victims' machines. In order to avoid being discovered, they send these unwanted emails via "throwaway" accounts, through open email relays, or via their own mail servers.

There exists a number of technologies to stop these threats. Spam is normally prevented either by means of blocking known spam sources (e.g., checking lists of offending DNS names and IP address ranges), such as Spamhaus [10], or through filtering content such as SpamAssassin [12]. There are several major sender authentication proposals which can be used to detect phishing and the kind of spam which uses forged addresses. Sender Policy Framework (SPF) [60] is IP-based authentication which validates the channel transporting messages. DomainKeys [19] uses a form of public key cryptography to authenticate email senders' domains. DomainKeys Identified Mail (DKIM) [22] is an e-mail authentication proposal that merges Yahoo's DomainKeys and Cisco's Internet Identified Mail [17] e-mail verification technologies.

Email virus scanners, such as Norton AntiVirus [6], use extracted virus signatures to detect and stop email viruses, either on server or client systems. However, like other virus scanners, they have difficulty in stopping novel unknown ones. To address this limitation, some researchers work on building email anomaly intrusion detection systems against unknown email viruses. Gupta et al. [25] detect email viruses by looking for increases in email traffic from email clients to servers. Stolfo et al. [53] use behaviors of email flows and email account usage to detect unknown email viruses. Based upon the observation of DNS MX queries within an enterprize network, Whyte et al. [59] present a technique for detecting and containing SMTP-engine based mass-mailing activity.

Email being illegitimately read and modified. This kind of security issue is referred to as message confidentiality exposure and message integrity compromise. It can happen when a message is in transmission or when a message is stored on a host (e.g., a sending site, a receiving site or an intermediate site).

Because of the open nature of the Internet, attackers can intercept a message in transit through packet sniffers. Attackers can also steal (directly or indirectly) a message by accessing the site where the message is stored temporarily or permanently (e.g., MTAs, mail stores). Indirectly, they can send malicious software to the site to compromise email software (e.g., mail clients, SMTP servers, IMAP servers) and make the software expose or modify information for them. Alternatively, they can directly (locally or remotely) access the site themselves by compromising authentication credentials. Because a computer can not determine who is misusing the authentication credentials, these latter attacks are a kind of insider attack.

In order to protect the confidentiality and integrity of an email message even when the message is intercepted by attackers, encryption and digital signature technologies are applied. Attackers can not obtain useful information from an encrypted message, and the receiver herself can find out that a message is modified by checking its signature. The most notable standards, OpenPGP [61] and S/MIME [46], both use encryption and digital signature to provide a solution for end-to-end integrity and confidentiality. TLS (transport layer security)/SSL (secure socket layer) [20] using public and private key technology, is applied for site-to-site protection such as web mail between web servers and browsers and IMAP messages between user computers and IMAP servers.

In order to stop insider attacks, two kinds of techniques are typically used: authentication and intrusion detection. The former works when an attacker requests access, while the latter works after the attacker has obtained access. In the email domain, the most commonly used authentication mechanisms are based on reusable text passwords. However, passwords may be easily compromised by malicious software, social engineering, or by user negligence. While there exist other technologies that could be used to authenticate email users in a more secure fashion (e.g., smart cards [9]), virtually all of them would require significant changes in how users access their email. Compared with those novel authentication technologies, intrusion detection technologies normally do not require any change on user interface while enhancing security. I discuss them in Sections 2.3 and 2.4.

2.3 Overview of IDSs

An intrusion detection system or IDS is a tool used to detect unauthorized access to a computer system or network. In terms of detection approaches, intrusion detection systems can be broadly divided into two categories—attack knowledge-based and normal behavior-based systems. Attack knowledge-based IDSs apply knowledge accumulated about past attacks to detect future ones. One advantage of attack knowledge-based approaches is that they have the potential for very low false positives. Drawbacks, however, include the difficulty of gathering the required information on known attacks and keeping up with new attacks. Normal behavior-based IDSs detect attacks by observing deviations from normal behavior of systems or users. If a deviation is observed, an alarm is raised. A key advantage of normal behavior-based IDSs is that they can detect new, previously unknown attacks. However, a high false alarm rate, due to complex ever-changing normal behavior, is a typical drawback of most normal behavior-based IDSs.

Based on different data sources, IDSs can also be classified into two groups—network-based and host-based. Network-based IDSs, located at choke points (e.g., routers, gateways) in the network to be monitored, capture and analyze network packets for malicious traffic. Host-based IDSs are used to monitor hosts or programs. Recently, a new category of IDS—application-specific, is coming up, which is designed for a particular application (e.g., web application, database). As its data source comes from an application which is part of a host, I categorize it as host-based IDS.

In the following, I will present an overview of the intrusion detection field according to the two detection approaches—attack knowledge-based and normal behavior-based, because I think this classification is more basic and better to reflect the perspectives of researchers on intrusion detection than the data source based classification.

Attack Knowledge-Based Intrusion Detection.

Attack knowledge-based IDSs can be divided into signature-based and rule-based. Signature-based systems analyze data streams for specific patterns (e.g., network packets for substrings correlated with attacks). Rule-based systems compare data to pre-specified rules (e.g., access policy to a specific file).

Signature-based intrusion detection techniques allow for very efficient implementation, and are therefore applied in commercial intrusion detection products [31, 58]. The techniques are also used by most virus and worm scanners. However, conventional signature extraction for novel viruses and worms is an expensive, slow, manual procedure that can take hours or even days to complete. Some researchers propose automatic generation of signature of unknown worms without human intervention. EarlyBird [50, 49], Autograph [33], and Honeycomb [35] all make use of worm propagation characteristics to automatically generate signatures that can then be used to filter or moderate the spread of worms elsewhere in the network. EarlyBird is based on two key worm behavior characteristics—highly repetitive packet content and increasing traffic volume. Autograph detects worms that propagate by randomly scanning IP addresses. Honeycomb extracts signatures from suspicious traffic caught in honeypots [2].

For the majority of current intrusion detection systems used to detect human inside attackers, rule-based detection plays an important role. That is because the inherent variation of human behavior makes it much harder to extract signatures of inside attacker behaviors compared with those of malicious codes (e.g., viruses). In addition, the other approach—normal behavior-based detection technique, typically has high false alarm rates. Rule-based approaches, in contrast, have the potential to produce low false alarm rates. However, they have difficulty in defining rules which can cover whole attack behaviors because of the diversity of attack behaviors. But with knowledge of a specific domain, it is possible to define rules covering a large majority of attack behaviors in that domain. Therefore, rule-based IDSs are appropriate for those domains which have very clear, critical security requirements such as military, government and finance. In the work of Wisdom & Sense [56], NIDES [11], Haystack [51] and Emerald [45], intrusive behavior rules play an important role in detection. The rules defined in Haystack are tied to characteristics and security requirements of the domain (Air Force defense).

Normal Behavior-Based Intrusion Detection.

Normal behavior-based intrusion detection systems (also called anomaly intrusion detection systems) rely on models of the "normal" behavior of computer systems, users, applications or network usage to detect intruders. Behavior profiles can be built by performing a statistical analysis of historical data or by using rule-based approaches to specify behavior patterns. Anomaly IDSs can be classified from the two data sources—network packet data and host data. The corresponding IDSs are respectively called network anomaly IDSs and host anomaly IDSs.

For network anomaly IDSs, different parts of packets are selected as features to build models. NSM [27] uses a four dimensional matrix of which the axes are: source, destination, service, and connection ID (a unique identifier for a specific connection). LISYS [29] is an immunological model of distributed detection which was similar to NSM except that its architecture permits the set of normal network flows to be distributed across a set of hosts. PHAD [39] extracts a total of 34 attributes from the packet header fields of Ethernet, IP, TCP, UDP and ICMP. Similar to PHAD, NATE [55] treats each of the first 48 bytes as a statistical feature, the first 40 bytes of which are in the header part and the latter 8 bytes of which are in the payload part. The work of Krugel et al. [36] and PAYL [57] present service-specific intrusion detection systems that combine the service type, length and payload distribution of the request as features. Instead of directly using packet data, some systems like EMERALD [45] reconstruct the network packets and extract features from semantic level data.

Host-based anomaly IDSs use two basic strategies: one is to model program behavior, which is mainly used for detecting viruses and worms; the other is to model user behavior, which is mainly used for insider attacks. In order to model program behavior, a number of researchers have studied system calls (see, for example, [34, 52]). One of noteworthy work is pH [52], which detects changes in program behavior by observing changes in short sequences of system calls. pH can detect buffer overflows, Trojan code and kernel security exploits; however, it has very limited ability to detect masqueraders. Higher-level system behavior has also been studied. For example, Inoue [30] uses the fundamental units (methods) in Java byte code to construct profiles for applications running in the Java virtual machine in order to stop back doors and viruses.

In the next section, I present in detail work on user behavior modelling.

2.4 User Behavior-Level Anomaly IDSs

Some researchers think that user behaviors might be more easily modelled for a specific domain or application. My work is related with both user behavior modelling and a specific domain (email domain). Therefore, in this section, I discuss user behavior-level anomaly IDSs from two perspectives—general host-based and domain-focused. The data collected by a general host-based IDS represents how users generally behave in a host, while the data collected by a domain-focused IDS is about user behaviors with respect to a particular set of applications.

General Host-Based. Owing to the inherent variation of human behavior, the biggest challenge in user-level anomaly IDSs is decreasing false positives while maintaining the ability to capture attackers.

A number of researchers focus on using different methods to model users via patterns of UNIX commands. DuMouchel [21] uses Bayesian statistics to model command sequences. Schonlau et al. [48] investigate and evaluate six statistical approaches for detecting masqueraders. Oka et al. [44] propose a novel method, called Eigen co-occurrence matrix (ECM) to model UNIX commands. As an extension of Schnlau's work, Maxion et al. [40] propose a new classification algorithm with improvement on detection rates and false alarm rates.

Like the above researchers, Lane [37] also uses UNIX command sequences as features to model user behaviors. Along with computer security issues (storage, speed), he also adopts machine learning techniques to address these problems in modelling—high noise environment, concept drift, skewed class distribution, and variable misclassification costs.

One noteworthy user behavior-level intrusion detection system is NIDES [11], which consists of both rule-based and anomaly-based detection approaches. In the anomaly detection part, NIDES creates a set of statistics components to model behavior for individual subjects: users, groups, remote hosts, and the overall system. Parameters of models are dynamically adjusted and specific to each subject. Audited activities are described by a vector of intrusion detection measures. As each audit record arrives, the relevant profiles are retrieved from the knowledge base and compared with the vector of intrusion detection measures. Thus, NIDES evaluates the total usage pattern, not just how the subject behaves with respect to each measure considered singly. Distinguishing features of NIDES are that multiple measures and dynamic adjusted parameters are considered. However, like other user behavior based anomaly IDSs, NIDES has high false positives.

In terms of feature selection, my work is more similar to the work using UNIX commands, because both use users' actual, direct operations rather than those indirect behaviors such as computer memory usage. In terms of modelling methods, my work has more similarity with NIDES, since both use statistical methods.

Haystack [51] also consists of both rule-based and anomaly-based detection approaches. In the anomaly detection part, two concepts are combined: per user models of how users have behaved in the past, and pre-specified generic user group models that specify generally acceptable behavior for a particular group of users.

Wisdom & Sense [56] studies historic audit data to produce a forest of rules which describe normal behavior. These rules are then fed to an expert system that evaluates recent audit data and alerts security officers when the rules indicate anomalous behavior. The majority of raw information were obtained from computer audit data (e.g., locations, object types, days of the week, person names).

Domain-Focused. In the computer world, a domain can be considered as one kind of computer application such as web application, database, or email application. In this section, I explain IDSs in four domains—database, credit card, wireless, and web application.

DEMIDS [16] is an IDS tailored to relational database systems. The essential approach is that users are associated with working scopes which comprise sets of attributes referenced with some values accessed by the users. A user's profile is built based on his working scope. When a user's behavior exceeds his working scope, his behavior is considered as anomalous. In this work, domain knowledge such as the data structure and semantics encoded in a given database schema is considered. DEMIDS can be used for misuse detection, in particular insider abuse. In my work, I view email archive as a kind of database. The key difference between my work and DEMIDS is that one email archive belongs to only one user and is accessed by the user while one database can be shared and accessed by thousands of users.

Chan and his colleagues [15] use data mining technology to detect credit card fraud. They devised a multi-classifier meta-learning[1]approach to address three issues related with credit card fraud detection—the number of fraudulent transactions being small compared with legitimate ones, the non-uniform cost of error, and the large volume of transactions per day. Their classifiers depend on both fraudulent and legitimate transactions. In most other user-level intrusion detection fields, it is hard to get fraudulent behavior data for modelling; for credit card companies, fraud data is easy to obtain. Therefore, the modelling methods used in their work can not be used for my work.

Other domains have been studied; for example, the use of different profiles for wireless intrusion detection. The profiles can be divided into two kinds. One is a device profile constructed through exploiting the unique hardware signature of wireless interfaces, operating systems, and other characteristics of wireless devices. The other is user behavior profiles. Hall and her colleagues [26] incorporate radio frequency fingerprinting (RFF) into a wireless intrusion detection system. Boukerche and his colleagues [13] use calling patterns of users for fraud detection in cellular network commercial systems, namely the Fraud Management System by Hewlett-Packard(FMS-HP) [28]. Work in [14, 54] makes use of sequences of cells traversed by users as a feature of the profile.

[1]Meta-learning is a technique recently developed that deals with the problem of computing a "global" classifier from large and inherently distributed databases. Meta-learning aims to compute a number of independent classifiers (concepts or models) by applying learning programs to a collection of independent and distributed databases in parallel.

2.5 Summary

My work is used to protect email archives from insider attacks. Because insider attacks consist of "authorized" accesses and manipulations of data, access control and authentication mechanisms are not sufficient to prevent them. Therefore, I use intrusion detection technologies to prevent them. There are two kinds of intrusion detection technologies—attack knowledge-based and anomaly-based. In the attack knowledge-based technologies, there are two types—signature-based and rule-based. Signature-based IDSs are normally used for detecting worms and viruses through extracting signatures of the malicious code. The characteristic of worms—copying themselves to propagate—makes this method work well. However, this method is not appropriate for insider attacks, because inside attackers are independent human beings who behave differently from one another. It is extremely hard to extract a common behavior signature from different individuals. Rule-based IDSs can be used for detecting inside attackers, but they are normally used for those applications which have clear, specific security requirements such as finance and military. Email is of more personal use and it is hard to define a set of uniform security policies for all email users. Therefore, I use anomaly technology by modelling legitimate users' behaviors to detect inside attackers.

However, most anomaly IDSs have a common problem—high false positives. With insider-focused anomaly IDSs, which model users' behaviors, it is difficult to obtain an acceptable false positive rate while allowing legitimate accesses. In order to reduce false positives or mitigate the influence of false positives, researchers work from two perspectives—modelling methods and response systems. Some researchers working on user behavior models have noticed differences between individual users, and therefore customize models for each user to some degree. NIDES, for example, dynamically determines parameter values for each user's model. Some researchers work on response systems in order to mitigate the influence of false positives. However, none of them analyzes characteristics of human being behavior and takes advantage of these characteristics. The key difference between my work and others' is that I always consider characteristics of human being behavior as a basis for my system design and architecture.

On this basis, I make user-specific design the focus of my system. My design is presented in the next chapter.

Chapter 3

Design of An Email Archive IDS

The key challenge that a designer of an email archive IDS faces is how to control false alarm rates at an acceptable level while maintaining detection ability. To meet this challenge, I propose a user-specific design for the system, which makes use of characteristics of both human beings and the email domain. This design includes two parts: modelling and response. In Section 3.1, I first describe the characteristics of the email domain which form the basis of my user-specific design. In Sections 3.2 and 3.3, I present my user-specific design and system architecture. Finally, I explain how this design has a significant impact on my implementation work described in the subsequent chapters in terms of modelling strategy and experimental methods.

3.1 Characteristics of the Email Archive Domain

A practical email archive IDS needs to meet two basic requirements: low computational cost and high effectiveness (acceptable false alarm rates and acceptable detection rates). An email system can have thousands of users, so even a moderate computational cost for each user can result in unbearable computational burden for the system. Therefore, the computational cost per user should be very low. In addition, false positives (or false alarm rates) must not be more than what the administrator or users can bear while maintaining detection ability. Otherwise, the administrators or the users will feel annoyed and ignore the alarms.

However, in general, there is a conflict between low computational cost and effectiveness. Normally, compared with a simple modelling method, a complex modelling method is more effective. However, the more complex a modelling method is, the more costly the computation.

It is much harder for user behavior-level anomaly IDSs to build effective models with simple modelling methods because the inherent variation of human beings makes their behavior hard to model. A person can frequently change his behavior for no apparent reason. Some changes result from status changes such as job change, while some of them are from emotional changes. A group of persons can have very different behavior patterns in terms of email access. Some persons might regularly login from the same machines, while some might often change login machines as they often travel. These two factors in human behaviors—the inherent variation of a single person and the variety of patterns of a group—make it difficult to effectively model human beings' behaviors. However, there still exist some practical off-line applications which predict events directly or indirectly related to human behaviors, such as stock predictions and the USA presidential selections. However, complex modelling methods (e.g., neural networks) are normally applied in these off-line applications, because they can afford much larger computational cost and data storage. Such complex modelling methods are not suited for the email domain because of the large number of users.

Therefore, it will be hard to make a general user behavior-level anomaly IDS which meets both of the requirements—effectiveness and low computational cost. However, for an anomaly IDS which is used for a specific domain, it is possible to take advantage of characteristics of that domain and design a domain-specific intrusion detection system in order to meet the two requirements.

Characteristics of The Email Archive Domain. One key feature of the email archive domain is that email is for per user use—each email archive belongs to only one user. The per-user characteristic manifests itself in four ways. First, every user can manage his email archive in his own way, for example by creating folders and setting up automatic filters. Second, users have different security requirements. People who use email for business might be more concerned with security than those whose email messages are for personal, casual use. Third, the behavior patterns of each user can be different. For example, some users always login at the same time or at the same places, while others always use the same mail clients. Finally, no one knows more about an email archive than the user himself. Thus, a knowledgeable user is able to notice many kinds of tampering. For example, he will notice that a new email has been read by someone else because of the obvious having-been-read mark on the email. He might suspect that something is wrong when the amount of newly received email in one day is much less than normal. On the other hand, he might not notice if the amount is a little less than normal. Therefore, users can serve as a limited monitoring function. Owing to these characteristics, I have designed my email archive IDS to be user-specific.

3.2 The User-Specific Design

Instead of designing a uniform IDS for all users, a user-specific system builds an IDS for each individual user according to his behavior pattern and security needs.

This user-specific design contains two basic parts—user-tailored modelling and user-involved alarm response. With user-tailored modelling, every user is assigned a model (or an IDS instance) according to his behavior pattern. For user-involved alarms, an alarm is sent to the corresponding user instead of administrators or security officers and the user gives the system a response indicating whether the alarm is a real security violation or not.

As users can have variable behavior patterns (which have been discussed in the last section), it is difficult for one simple model to apply equally well to all users. To overcome this problem and to maximize the effectiveness of a user's model, the user should have a unique model based on his behavior pattern.

In conventional IDSs, security officers or administrators are responsible for handling alarms (false or true) for all users and therefore, it is the administrators who tolerate false positives (false alarms). In this way, the presence of false positives inherently limits the scalability of intrusion detection systems. For example, a system that would only produce one false alarm per week for 50 users would, without modification, produce around 14 alarms per day for 5000 users. Inherent in the design of any IDS, then, are assumptions about the size of the observed population relative to that of the monitors (system administrators or security officers). With my design, we have a logical IDS for each user and it is users who manage their own false positives. Therefore, the system is inherently scalable—more users means that there are more people monitoring the IDS and thus, as a whole, the system can tolerate a higher rate of false positives. In addition, because a user understands his own behavior, he is best equipped to determine whether or not a given alarm reflects a genuine security violation.

3.3 The Architecture

3.3.1 Overall Architecture

Based on these two parts (user-tailored modelling and user-involved response) of the user-specific design, an email archive IDS should consist of four components—a template store, a user-specific analysis system, an IDS instance store, and an alarm response system. They cooperate to achieve two tasks—generating a unique IDS instance for a user and monitoring the user's behavior.

Components. The template store contains a number of sub-models and rule sets which are used for building an IDS instance. The key idea behind this is that user behavior patterns are variable, and therefore, multiple observables are needed. Observables are actually features used for modelling. The observables can be login times, login places, the first few commands and so on. A sub-model in this design is built based on an observable. For example, based on the observable of login time, a sub-model is built and referred to as the login-time sub-model. Normally, in many IDSs, one single observable is used to model all users' behavior. In my design, multiple observables are used to build multiple sub-models. The terminology "sub-model" in my architecture is equivalent to the terminology "model" used in those IDSs which use only one observable. Figure 3.1 is an example of a template store. In the template store, each sub-model is associated with one behavior observable. Each parameter in a sub-model is assigned a default range rather than a fixed default value. The idea behind parameter ranges is that parameter values should be dynamically determined by user behaviors rather than being derived solely from the experience of designers. The rule sets include a selection rule set for selecting sub-models for a user, a combination rule set for combining the selected sub-models of the user, and a performance rule set for assessing effectiveness of the user's model. The rules determining how to select, combine and assess will be explained later in this section.

The user-specific analysis system collects a user's behavior data, and then with the template store builds an IDS instance for the user according to his behavior pattern and security needs.

The IDS instance store contains users' IDS instances generated by the template store and the user-specific analysis system. Figure 3.2 shows components in an IDS instance for a user. In the IDS instance, there are m sub-profiles respectively equivalent to the m sub-models, the subset of n sub-models in the template store. Each

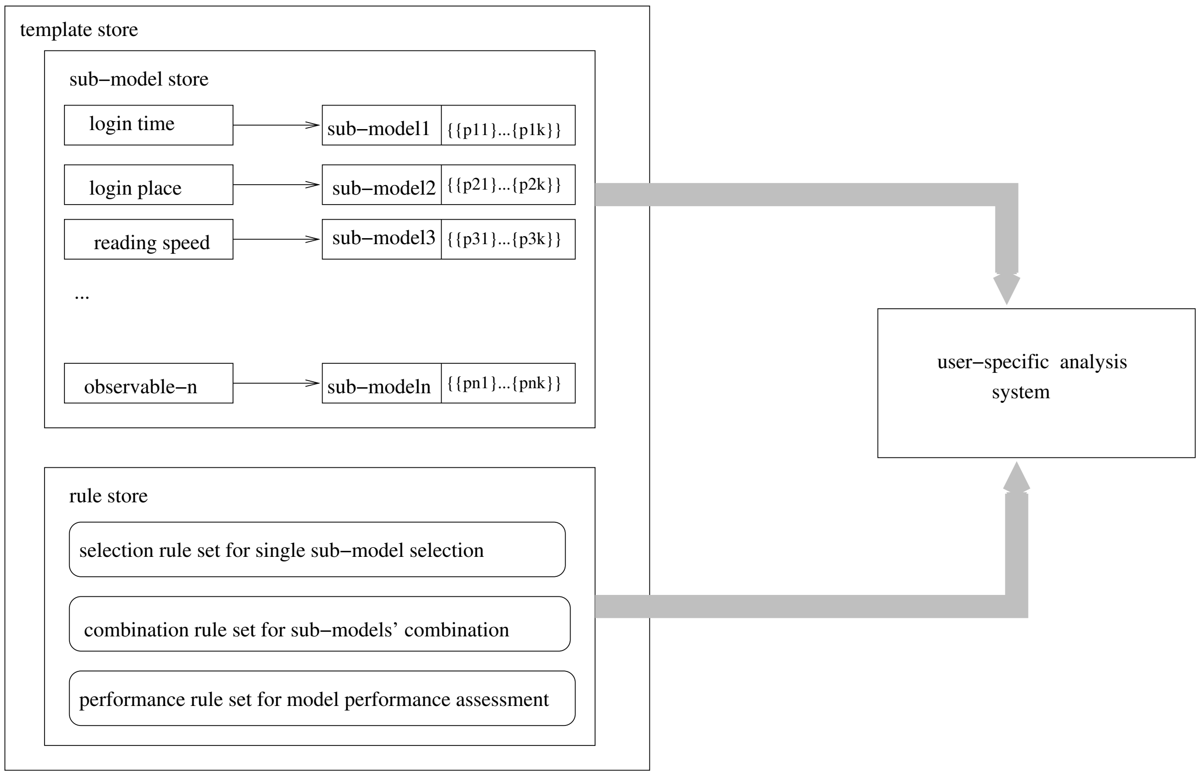

Figure 3.1: Template store and user-specific analysis system.

sub-profile (namely, sub-model) has a set of parameter values. Each sub-profile is associated with one observable, as discussed above. How should we combine different kinds of sub-profiles (or observables) of a user to make them cooperate to monitor the user's behavior? I use the vector measure method, which is used in NIDES [11]. In my architecture, I use one vector to represent one sub-profile of a user, and therefore there is a total of m vectors for the user in Figure 3.2. The m-vector measure is used to measure the difference between the user's profile composed of the m sub-profiles and his current behavior. If the difference exceeds the threshold, an alarm is raised.

The alarm response system directly interacts with a user when an alarm is raised by the user's IDS instance, without involvement of administrators or security officers. An IDS instance and the alarm response systems form a complete intrusion detection system for the user with functions of monitoring and alarm response.

Detection. Suppose that a user has an IDS instance built. Figure 3.3 shows

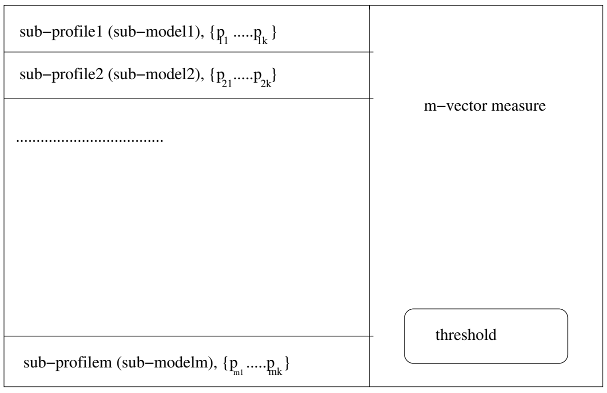
sub–profile1 (sub–model1), {p_{11} .....p_{1k} }
sub–profile2 (sub–model2), {p_{21}.....p_{2k}}
....
sub–profilem (sub–modelm), {p_{m1} .....p_{mk} }
m–vector measure
threshold

Figure 3.2: An IDS instance for a user.

how Alice's IDS instance (IDS-Alice) collaborates with the alarm response system to monitor Alice's behavior. There are six steps in the figure. 1) The IDS instance (IDS-Alice) monitors Alice's behavior. 2) The IDS instance (IDS-Alice) notices that Alice's behavior is unusual and then sends an alarm to the alarm response system. 3) Over an independent channel or an alternative account (which will be talked later), the alarm response system sends Alice an alarm message to verify whether the alarm is false or not. 4) If the alarm is real, Alice might take action herself such as changing her password. If the alarm is false, Alice indicates this to the alarm response system. 5) With a false alarm, the alarm response system will report to the user-specific analysis system about this event. 6) The user-specific analysis system will update (or adjust) Alice's IDS instance (IDS-Alice) immediately or later.

Generation of An IDS Instance. Figure 3.4 shows how an IDS instance is generated for a user. There are five steps. 1) When the system gets a request from a user for an intrusion detection system, the user-specific analysis system starts to collect the user's behavior data. 2) The user-specific analysis system retrieves the relevant template parts (sub-models and rules) from the template store. 3) The user-specific analysis system presents the user in an easy-to-understand way the effectiveness of several (e.g., 10) best choices of intrusion detection systems. According to his security needs, the user chooses the IDS he prefers. 4) Based upon the choice, the user-specific

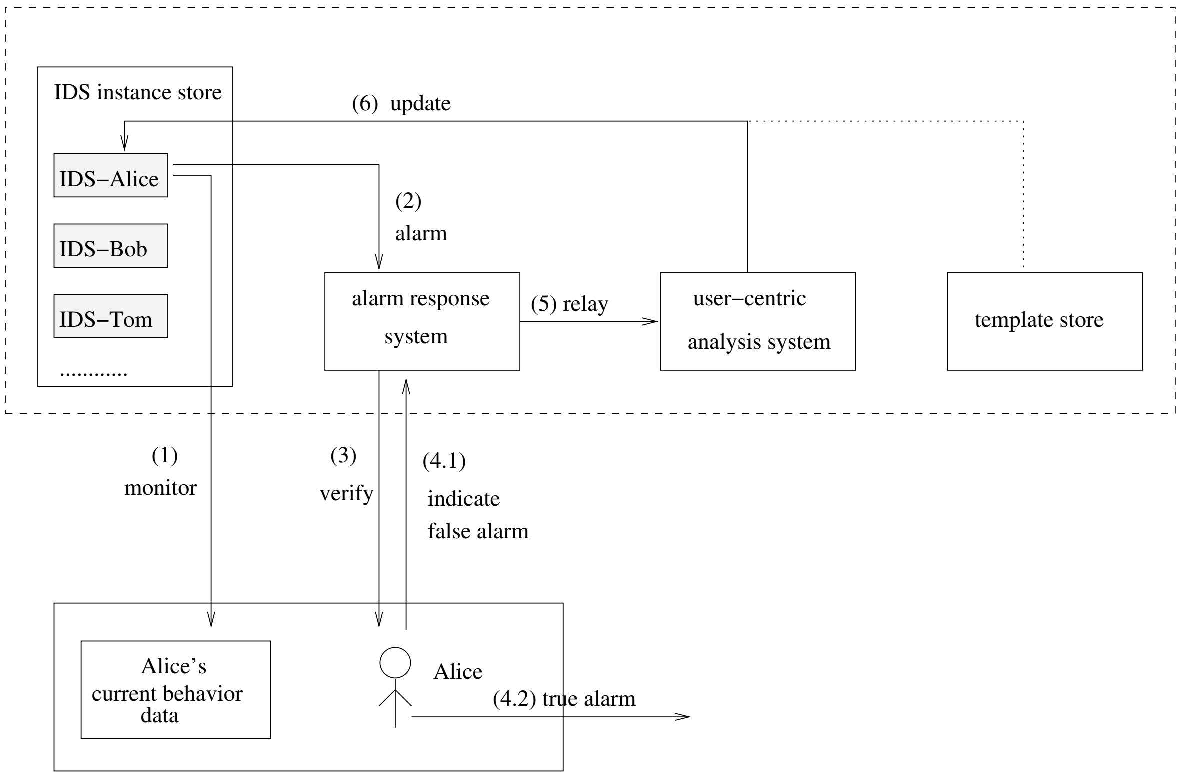

Figure 3.3: How an IDS instance works with the alarm response system.

analysis system finalizes an IDS instance for the user. 5) The IDS instance is stored in the IDS instance store.

Details about how the user-specific analysis system interacts with the template store are as follows. First, the user-specific analysis system selects appropriate sub-models from the template store for the user based on the selection rule set and the user's data. Second, the user-specific analysis system combines the chosen sub-models into a rough model based on the combination rule set. The M-vector measure (part of the IDS instance) is formed in the process of combining those selected sub-models. Third, the user-specific analysis system refines the rough model by adjusting parameters. This refining process is equivalent to assigning a set of optimal parameter values for each sub-profile based on the performance rule set.

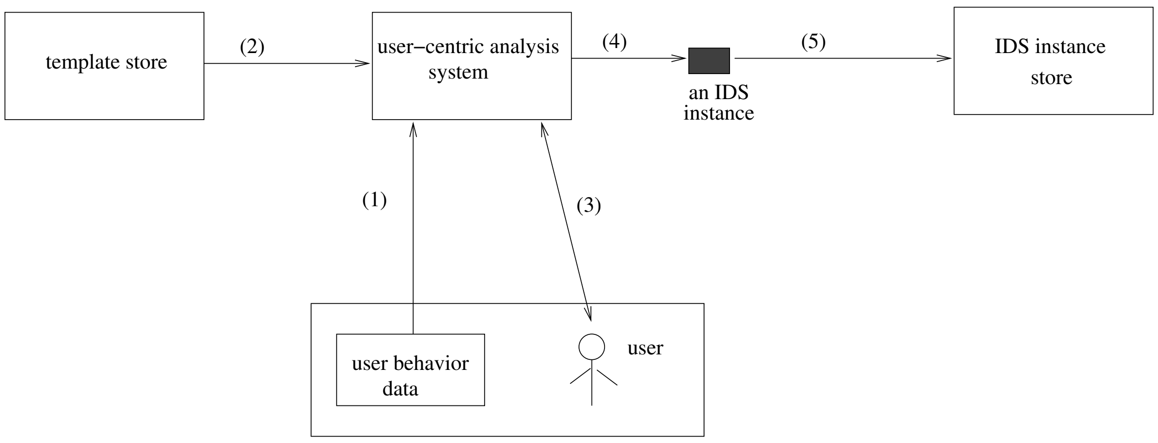

Figure 3.4: Generation of an IDS instance for a user.

3.3.2 Requirements and Challenges.

There are a number of requirements and challenges involved in implementing such a system. First, as users get involved in the processes of generating IDS instances and responding to alarms produced by the system, the interaction between the system and users must be easy to understand for the users. Second, the interaction should occur over a secure channel. Even if the channel used for the email archive access has been compromised, the interaction between a user and his IDS still can be safely processed. Third, though the generation of an IDS instance is processed off-line, computation and storage costs still need to be modest if the system is to scale. Fourth, it is a big challenge to build a template store. The challenge can be divided into how to define observables, how to build a sub-model based on the defined observables, how to define a value range for each parameter in a sub-model, and how to define the three types of rules (selection rules, combination rules and performance rules). Finally, it is another security question about how the template store and IDS instance store are secured so that the attacker doesn't simply modify them before attempting to impersonate the user.

In the following, I discuss what the problems are and my proposed solutions.

Easy-To-Understand Interaction. As email is used for many kinds of persons, including those who are not computer literate, the interaction between users and IDSs must be designed to at least as easy to use as the email interfaces. Users are not necessarily intelligent on IDSs. However, the users need to have some basic knowledge on IDSs as they have of virus scanners. Suppose that the user-specific analysis system provides a user with two choices of intrusion detection systems. According to her security needs, the user will tell the user-specific analysis system which one she prefers. The following is an example about how the system could present the available IDSs to a user.

"Dear user, Based on your email access behaviors, you have two recommended intrusion detection systems. You can choose one of them.
Option 1: You will be bothered approximately three times each week by false alarms. The probability that an attacker can be detected on first intrusion is 80%.
Option 2: You will be bothered approximately one time every two weeks by false alarms. The probability that an attacker can be detected on first intrusion is 50%."

Secure Channel. Because the user-involved response system interacts directly with the protected user, it needs a secure path of communications with that user; however, by assumption the protected user's password(s) may be compromised, so we cannot create such a trusted path by using a password. While this might seem like a fatal flaw, in practice there are many potential solutions. For example, the IDS could use an independent communication channel to transmit alarms such as instant messaging, mobile phone text messaging via Internet gateways; or an alternative email account with a different password is used for storing the user's alarm messages[1]. Another, potentially simpler alternative would be a clear alert in the mail reading client or web browser that cannot be removed by the user; instead, it would persist

[1]With this approach, attackers might attempt to compromise both of the user's email accounts together—the email message account and the alarm message account.

on-screen for a fixed period of time (e.g., a few days) and then be automatically removed by the system. The persistence might be annoying for legitimate users. The most appropriate design for such a system would depend upon the deployed system and the security requirements of the specific users; what is important is that attackers should not be able to stop alarm messages, while false alarms should not create too much of a burden on the protected user.

Building the template store. A number of useful observables need to be first studied and defined based upon knowledge of the email domain and observation of user behaviors. Email access observables are numerous, including events such as login time, login place, the first few sent IMAP commands, mail client types, and sequence of reading email messages. Compared with building sub-models and defining value ranges of parameters, I feel that definition of the three types of rules (selection rules, combination rules and performance assessment rules) are much more complicated. It is a big question how to define the rules so that sub-models can be combined well.

Computational Cost. The computational cost mainly lies in the generation of an IDS instance for a user. The generation process can be roughly divided into two steps—generating a rough model and generating a number of refined models. The rough model is generated through selecting single sub-models and combining these single sub-models via combination rules. A number of refined models are generated by adjusting parameter values of the rough model and then choosing a couple of refined models according to the performance rules. The chosen refined models will be sent to the user for final selection.

The generation of refined models can cost much more computation time than that of the rough model. To see why, consider the following example about how a number of refined models are generated from a rough model using an exhaustive search algorithm. Suppose a user has had a rough model which is composed of two sub-models, s_1 and s_2. As described above, each sub-model has a number of parameters and each parameter in a sub-model has a parameter range. Suppose the sub-model s_1 has two parameters—p_{11} with 2 value choices and p_{12} with 3 value choices. The sub-model s_2 has three parameters—p_{21} with 2 value choices, p_{22} with 3 value choices and p_{23} with 4 value choices. According to the exhaustive search algorithm, the total number of combinations of parameters is $2 * 3 * 2 * 3 * 4 = 144$. With each combination, a refined model is generated with false and true positives, which are then used for performance assessment. Therefore, the overall computation will be 144 times the computation of a single set of parameters. More specifically, with 96 combination sets of parameters, my experimental program spent around 60 seconds on the user Faculty's three months of data. Both the total number of parameters and the value range of each parameter are very small in the above example; therefore, we can imagine how large the computational cost will be in a practical environment with the exhaustive search algorithm. However, if we choose a more efficient algorithm (e.g., a genetic algorithm [3]), the computational cost might significantly be reduced.

3.4 The User-Specific Design and My Implementation Work

In order to explore the feasibility of the user-specific design, I have implemented in the next chapters part of the design using only one observable—the correlation between user dispositions of newly received email messages and the corresponding mail senders. Based upon the observable, a sub-model (hereafter called the model) is built. This work is presented in detail in Chapter 4.

According to my previous discussion about the user-tailored modelling, there are two major steps for building a model for a user—combining multiple sub-models into a rough model and refining the rough model by adjusting parameters. Because currently I only have one sub-model, I have only implemented the model refinement part. I establish a value range for each parameter of the model and then choose the optimal set of parameter values for each user according to false and true positives. The model, with an optimal set of parameter values, is thus customized to a user.

As my current work mainly focuses on the feasibility of the design and the user-involved response system is more an issue of deployment, I have not developed the response system. However, my experiments are designed using the presented architecture as a base. The idea is that alarms will be sent to the corresponding users without involvement of administrators. It therefore makes more sense to count the alarm rates per user instead of for all users in the email domain. Therefore, my experiments are per user, and the program respectively calculates false positives (false alarm rates) and true positives for each user.

In the next chapter, I explain how the chosen observable is used to build a model of user behavior.

Chapter 4

Modelling Email User Behavior

In this chapter, I present a model of user email archive access behavior based on the correlation between user dispositions of newly received email messages and the corresponding email senders. Section 4.1 explains the intuitions and feature selections that underly my model. Section 4.2 describes the model itself. To evaluate this model, I need attacker behavior data in addition to normal user behavior. Section 4.3 explains my approach to simulating attacker behaviors.

4.1 Intuition, Observation and Feature Selection

There are many possible observables in user email access, such as the time of day of archive accesses, the amount of data transferred, and the program used to access the archive. It seems that every observable has the potential to be a feature that can be modelled. I have chosen one that I feel from my intuition and knowledge of daily life is more consistent than other observables. In daily life, behavior consistency of a real person depends on two aspects—habits and stable internal factors. I take eating as an example. A person might eat ice cream every day because ice cream is his favorite food. It is a habit that causes him consistently to eat ice cream. He also eats celery every day, because he has high-blood pressure and celery can help mitigate the condition. The condition is the stable internal factor which causes his regular consumption of celery. In social lives, every one has social relationships such as family (relation between wives and husbands, between brothers and sisters), and business (relation between students and professors, co-workers). And when facing other persons, how a person talks and behaves normally depends on the relationships with those persons. For example, in front of family members (e.g., parents, spouse), he behaves casually. However, in front of bosses, he normally behaves and talks carefully. Therefore, social relationships are one of the internal factors which can influence a person's external behavior. As one social relationship can exist for a long time, the external behavior associated with the relationship can be consistent for a long time too. In the email domain, a social relationship exists between a user and his email sender. How a user disposes of email messages from an email sender (e.g., reading, deleting, forwarding it) can reflect the relationship between the user and the email sender, causing the disposition behavior to be consistent to some degree. For example, in many cases a user would read and then archive email messages from his boss. For email messages from an announcement mail list, the user might delete them directly without reading. In addition, disposition behavior can also reflect a user's habits such as filing habits. Thus, I choose this observable for my model of user behavior.

Note that there is a fundamental difference between how users approach newly arrived email and already received email. User accesses to old email are dictated by the specific semantics of individual messages and the tasks the user is engaged in; new email messages, however, are frequently read, deleted, or responded to soon after they have arrived. Because of this observation, I have focused on user disposition behavior on newly received email messages rather than old email messages.

There are a number of user dispositions related with new email messages, such as reading, deleting, copying, moving, forwarding, replying, marking unread and so on. Rather than building a model that incorporates all of these possibilities, I have instead chosen to focus on the following six common operations: doing nothing, reading, deleting, copying, moving, and marking a message unread, because I believe that these six operations can reflect a user's habits or relationships to email senders better than others. The two operations, copying and moving, can reflect a user's filing habits. Another three operations, doing nothing, reading and deleting, reflect relationships between the user and mail senders. Some email originating from an announcement email list might be read and deleted or deleted directly; email from an important colleague, though, would typically be read and archived. The operation, marking a message unread, can reflect a user's habit a little, but can more reflect an attacker's actions. In order to hide himself, an attacker might choose to mark unread a message which he has read.

Therefore, for each newly received email message for a given user, I record the sender of the email (from the "From:" line of the message) along with which of the selected disposition options was performed. In order to simplify my model, I ignore the parameters of these operations, e.g., the destination folder for a move operation.

Note that in reality whether a user does the five dispositions or not is mainly driven by contents of email. But mail senders have high correlation with contents.

4.2 The Model

To test my idea about correlations between disposition operations and email senders, I have developed a model using this feature. In the model, each user has a profile consisting of two parts—long-term (e.g., one month) data which contains frequencies of each disposition operation along with the corresponding email senders; a set of parameter values. After a profile has been trained on a sufficient amount of user data, it is then used to measure significant changes in that user's recent short-term (e.g., one day) behavior. When those changes exceed a fixed threshold, that short-term behavior is considered to be anomalous.

The long-term data reflects the behavior patterns of a user. It is symbolized as $\{l\}$, with each l representing the record of dispositions of messages sent from one address. The short-term data represents recent monitored behavior and is symbolized as $\{s\}$, with each s representing a message as well as its disposition operations. The sizes of both the short-term data and the long-term data are related with statistical requirements and a user's actual situation. If the sizes are too small, they will not provide a sufficiently large sample. If the sizes are too big, they will bring difficulties to the IDS on storage and computation. How the two sizes are set up is presented in detail in Chapter 6.

To test my model in Chapter 6, a sliding window is used to select messages representing respectively the long-term data and the short-term data. To be consistent with the sliding window concept, the long-term data is also referred to as a long-term data window and short-term data a short-term data window in this and the following chapters. Details about the windows are presented in Section 6.2.

To measure the difference between short-term data and long-term data, I have defined three specific measures—message variation M, sender confidence C, and window variation W. They are defined as follows.

4.2.1 Message Variation M

The message variation M is the distance between the dispositions of a given message in recent short-term data and the average dispositions of past messages in long-term data from the same sender as the given message.

In order to compute the distance, dispositions are placed within a five dimensional space, with each dimension corresponding to a particular type of message disposition:

reading (r), deleting (d), moving (m), copying (c), marking unread (u). A single recent message s in short-term data is represented in this space as:

$$s = [sender\#,\ [s_r, s_d, s_m, s_c, s_u]]$$

The value of sender# is the number of the mail sender who sends this message. The first time that a mail sender sends a message to the user, the program assigns a number to the mail sender by increasing the value of the largest sender#. For example, a user has had five email senders' messages and therefore, the largest sender# is 5. When a new mail sender sends a message to the user for the first time and program can not find the corresponding sender# for the mail sender, the program will assign 6 as sender# to that mail sender. The values of s_r, s_d, s_m, s_c and s_u, either 1 or 0, represent a user's actual operations on the five types of dispositions. Value 1 means that the user does the corresponding disposition operation and value 0 means that the user does not do that. Based upon the three users' dataset, I have observed that during the period of the short-term (one day by default), most mail senders only send one message to the user. Therefore, I normalize the frequencies as 1 and 0. Table 4.1 is an example of records in a user's short-term data.

A record in long-term data is represented in the space as:

$$l = [sender\#,\ [l_r, l_d, l_m, l_c, l_u]]$$

The value of sender# is the number of a mail sender who has sent at least k messages in the long-term period and is assigned in the same way as the sender# in s. The values of l_r, l_d, l_m, l_c and l_u respectively represent frequencies of each disposition of the k messages sent by the sender in that period. Table 4.2 is an example of records in a user's long-term data.

sender	s_r	s_d	s_m	s_c	s_u
1	1	1	0	0	0
2	1	0	0	0	0
4	0	0	0	0	0

Table 4.1: An example of short-term behavior data. This user has received 3 messages from three distinct email senders (sender 1, sender 2, and sender 4) in a day. For the email from sender 1, this user reads ($s_r = 1$) and deletes ($s_d = 1$) it. For the email from sender 2, he only reads ($s_r = 1$) it. For the email from sender 4, he does nothing with it.

sender	k	l_r	l_d	l_m	l_c	l_u
1	10	0.8	0.2	0.0	0.0	0.1
2	8	0.5	0.0	1.0	0.0	0.8
3	20	0.0	1.0	0.0	0.0	0.0

Table 4.2: An example of long-term data. This user has received 38 messages from three distinct email senders. From the first email sender (sender 1), the user has received a total of 10 messages. He has read 8 messages, deleted 2 messages, and marked one message unread.(The numbers of each operation are not shown in the table.) Therefore, for messages from sender 1, the frequencies of reading, deleting, and marking unread are respectively 0.8, 0.2, and 0.1.

I have used Euclidean distance to compute the distance between two points, s and l, which correlate with the same sender# in the five dimensional space. Euclidean distance is the ordinary distance between the two points $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$ in $n-$space, which is defined as:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$$

Based on the Euclidean Distance metric, the message variation M is defined as:

$$\Delta_r = s_r - l_r$$

$$\Delta_d = s_d - l_d$$

$$\Delta_m = s_m - l_m$$

$$\Delta_c = s_c - l_c$$

$$\Delta_u = s_u - l_u$$

$$M = \sqrt{\Delta_r^2 + \Delta_d^2 + \Delta_m^2 + \Delta_c^2 + \Delta_u^2}$$

For example, assume that a user's long term behavior profile is described in Table 4.2, and the user receives a new message from sender 1 which the user decides to read and delete, as shown in the first line of Table 4.1. Then, the message variation M for this message is:

$$\begin{aligned}\Delta &= \langle (1-0.8)_r, (1-0.2)_d, (0-0)_m, \\ &\quad (0-0)_c, (0-0.1)_u \rangle\end{aligned}$$

$$\begin{aligned}M &= \sqrt{0.2^2 + 0.8^2 + 0^2 + 0^2 + (-0.1)^2} \\ &\approx 0.8307\end{aligned}$$

Alternative Calculation of M. Among the five dispositions (reading, deleting, moving, copying, making unread), frequencies of the first two dispositions (reading and deleting) can represent how important a mail sender is for the user. Frequencies of the next two dispositions (moving and copying) can reflect a user's habits. But as for the last disposition, marking unread, it does not reflect either relationships or habits. According to my observations, whether a legitimate user marks a newly received message unread depends on the actual situation. For example, when a user does not have time to finish up reading a message, he might mark the message as unread to reminds himself of that. Based on this observation, I have modified the Euclidean Distance metric for the operation of marking unread. In my modified Euclidean Distance, Δs for other operations are the same as ones in the Euclidean Distance metric except for this operation. In the modified method, I only care whether a user ever did that operation (marking unread) rather than the frequency of the operation.

l_u	s_u	Δ'_u
> 0.0	1	0
> 0.0	0	1
$= 0.0$	1	1
$= 0.0$	0	0

Table 4.3: Calculation of Δ'_u in my modified Euclidean Distance Metric.

If the user ever did that operation in his long-term data (l_u more than 0.0) and he also did that in short-term data, or if the user never did that operation (l_u equal to 0.0) in long-term data and he did not do that operation either in short-term data, Δ'_u will be 0. Otherwise, Δ'_u will be 1. Therefore, the value of Δ'_u is either 1 or 0. Calculation of Δ'_u is described in Table 4.3.

Based on my modified Euclidean distance metric, the alternative calculation of *message variation* M' is defined as:

$$M' = \sqrt{\Delta_r^2 + \Delta_d^2 + \Delta_m^2 + \Delta_c^2 + {\Delta'_u}^2}$$

For example, assume that a user's long term behavior data is described in Table 4.2, and the user receives a new message from sender 1 which the user decides to read and delete (as shown in Table 4.1). Then, the alternative calculation of message variation M' for this message is:

$$\begin{aligned}\Delta &= \langle (1-0.8)_r, (1-0.2)_d, (0-0)_m, \\ &\quad (0-0)_c, (1.0)_u \rangle\end{aligned}$$

$$\begin{aligned}M' &= \sqrt{0.2^2 + 0.8^2 + 0^2 + 0^2 + (1.0)^2} \\ &\approx 1.2961\end{aligned}$$

Parameter p_v refers to which calculation method of message variation M is used in the model. When the parameter $p_v = 0$, the normal Euclidean Distance based method is used; when the parameter $p_v = 1$, the alternative method based on my modified Euclidean distance is used. By default, p_v is set to 0.

4.2.2 Sender Confidence C

The number of messages from different mail senders is extremely variable in long-term behavior data. A given user, in the period of one month, can receive dozens of messages from one sender while only receiving one from another. Accurate modelling requires an adequate number of samples; to determine whether I have seen a sufficient number of messages for predictive purposes, I need a per-sender measure of my confidence in the model. The measure, referred to as *sender confidence* (C) is assigned to each mail sender in the long-term data profile according to the number of messages the mail sender sent in the long-term period.

To measure this C, I divide the number of messages k received from an email sender by a fixed threshold p_C. I also, though, set the maximum possible C value to be 1. In other words, I define C as:

$$C = \min(\frac{k}{p_C}, 1)$$

By default, $p_C = 10$; thus, I am maximally confident in my model with respect to an email sender once I have received ten or more messages from that sender.

4.2.3 Window Variation W

Window variation W measures the difference between short-term behavior data and long-term behavior data. The smaller W is, the closer a user's current behavior is to his past behavior. It is defined in terms of the values of *message variation* M of all messages within the short-term behavior data and their corresponding *sender* *confidence* (C). Specifically, I define W as:

$$W = \frac{\sum_i C_i M_i}{\sum_i C_i}$$

Here, i ranges over all messages in a user's short-term data window whose corresponding senders can be found in the long-term data window. C_i is confidence value of the mail sender who sends the message i. For example, consider that a user's long-term data is as Table 4.2 and his short-term data is as Table 4.1. Then the *window variation* W (with $p_v = 0$ and $p_C = 10$ by default) is calculated as :

$$\begin{aligned}
M_1 &= \sqrt{(1-0.8)^2 + (1-0.2)^2 + (0-0.0)^2 + (0-0.0)^2 + (0-0.1)^2} \\
&\approx 0.8307 \\
C_1 &= 1.0 \\
M_2 &= \sqrt{(1-0.5)^2 + (0-0.0)^2 + (0-1.0)^2 + (0-0.0)^2 + (0-0.8)^2} \\
&\approx 1.3748 \\
C_2 &= 0.8 \\
W &= \frac{M1 * C1 + M2 * C2}{C1 + C2} \\
&= \frac{0.8307 * 1.0 + 1.3748 * 0.8}{1.0 + 0.8} \\
&\approx 1.0725
\end{aligned}$$

As sender 4 doesn't have a corresponding record in the long-term data, the message from sender 4 in the short-term data is not counted in the calculation of W.

4.2.4 Detection Threshold Setup

A detection threshold T is used to decide whether the window variation W of a user's short-term data is anomalous or not. Specifically, if $W > T$, then an anomaly is signalled. A user's detection threshold is set in terms of $\overline{W}$, the average value of window variations in training data. The calculation of T is as follows:

$$\begin{aligned}\overline{W} &= \frac{\sum_j W_j}{j} \\ T &= p_W \overline{W}\end{aligned}$$

Here, j ranges over all short-term data windows in the training data. p_W is a parameter that by default is set to 2.0 according to my preliminary experiments and knowledge on the user model.

4.3 Simulated Attack Behaviors

To assess the quality of a user's model, we at least need two values—a false positive rate and a detection rate (a true positive rate). False positive rates are calculated using user data, while detection rates are calculated using attacker data. However, it is extremely difficult to obtain real attack behavior data in the email archive domain. Therefore, I need to simulate attack behaviors. As there is generally a trade-off between false positives and true positives in anomaly intrusion detection systems, a casually simulated attacker might bring about very good experimental results—very low false positive rates and very high true positives. Therefore, the choice of an attacker simulation method directly affects the trustworthiness of the experimental results[1].

Because there are too many kinds of attack behaviors, it is not possible to do experiments for all of them. What we can do is to choose a type of attacker behavior which has been verified to be harder to distinguish from normal user behavior, compared with other types of attack behaviors. If the system can detect the tougher attack behavior type, it should have no problem in detecting easier types. It is hard

[1]In terms of attack behavior models, the key difference from misuse IDSs is that I use attack models to test my user model and misuse IDSs use attack models to detect attackers through matching. In the email domain, attackers' behaviors are very variable, and thus it is very hard to detect email attackers only through misuse IDSs.

to verify that a single type of attack behavior is tougher without comparison to other types. Therefore, we need to systematically design a number of attack behavior models and choose one out of them.

How can we systematically design a number of attack behavior models? My method is based on two concepts—attack space and attack scenarios. Their relationships are shown in Figure 4.1. First, I define attack scenarios starting with relationships between users and attackers. The relationships are divided into family relations, friend relations, coworker relations and non-relations (where attackers do not know users). Each relationship can reflect one kind of attack goal. Attackers having family relationships with users are most likely interested in the users' personal life. However, attackers having coworker relationships with users most likely want to know the users' business information. Well-defined attack scenarios should cover the majority of the attack space. Then, based on different goals in each attack scenario, I define attack behavior models covering as much of the space of attack scenarios as possible. Finally, based on experimental analysis and my knowledge of the email domain, I choose the most appropriate attack model from the defined attack models for further testing of the user model. The criteria for choosing the most appropriate attack model includes two parts—the system should have more difficulty in detecting this attack model than others; users themselves should also have difficulty in detecting actions of this simulated attacker. These criteria will be presented in detail in Section 6.3. In this section, I only focus on the first two steps—defining attack scenarios and defining attack behavior models in each attack scenario.

Attack Scenario. Based on the four kinds of relationships between attackers and users, I have designed five attack scenarios, as described below. In order to be easily understood, I present the attack scenarios in the form of stories and also give them story-like names.

1. Jealous Wife #1. A wife suspects that her husband is cheating on her. So, she

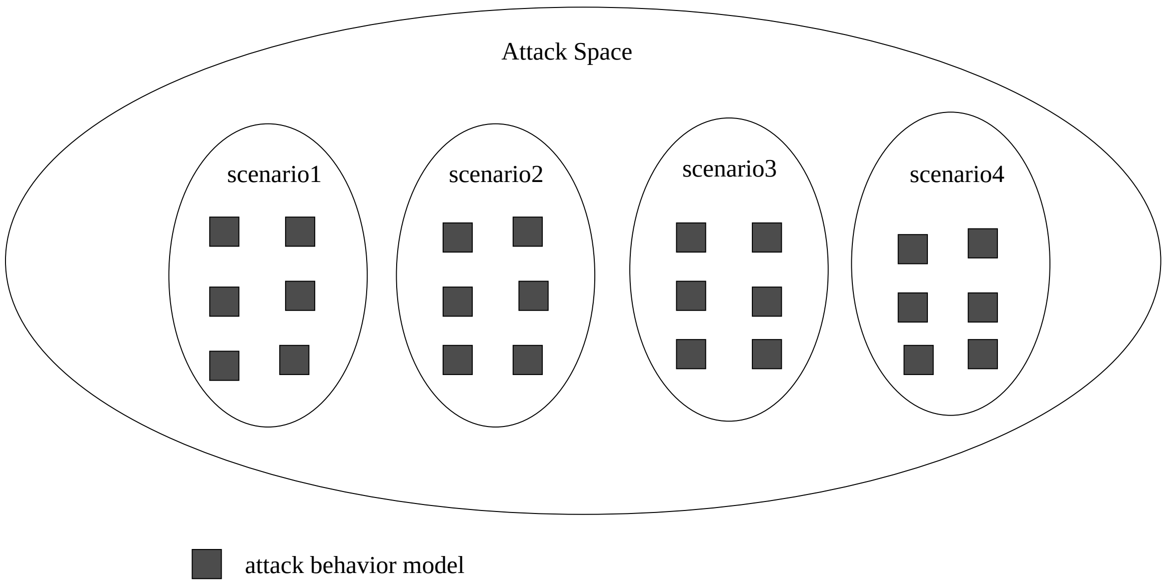

Figure 4.1: Relation between attack space, attack scenario and attack behavior model. Note that intersections between the scenarios can also exist.

checks her husband's email to look for proof. Since she is not familiar with her husband's email correspondents, she reads every newly received email. Because I think a spouse is the most likely potential attacker for a user in a family relationship, I choose a spouse as an attacker representative of family relations. This scenario can represent both the family and the friend relationships.

2. Jealous Wife #2. A wife suspects that that her husband has an inappropriate relationship with a woman she knows. She obtains the woman's email address by checking contents of her husband's email. After that, she only checks email sent by that woman. This scenario can also represent both the family and the friend relationships.

3. Co-workers. There is competition between two business men. One business man illegitimately accesses the other's email account in order to obtain his business information. Therefore, the attacker is only interested in those email messages from the few email senders related with the business. This scenario

can represent the co-worker relationship.

4. Smart Attacker. An attacker knows the mechanism of intrusion detection system somehow, and therefore he acts very carefully. So he only reads old messages and does not touch new mail. This scenario can represent the family, the friend and the co-worker relationships.

5. Random Vandals. The attackers do not know users. They get the users' email account information through dictionary attacks, spyware or other means. Most of the information in the email archive does not make sense for them and they do not care whether they will be discovered or not. So they act casually and randomly. This scenario represents the non-relations.

Attack Behavior Models. Each scenario corresponds to a number of possible attack behaviors. Based on the five scenarios, the 14 types of attack behavior models are defined in Table 4.4.

When a simulated attacker must choose an operation between deleting or marking a message as unread (the attack models AUD, OUD and IUD), either of the operations is chosen with probability 0.5. For the model RO, the five disposition operations are independent of each other, each with probability 0.5 to be chosen.

In the first four models (AU, AD, AUD and AN), the hypothetical attacker is assumed to read every message; in practice, however, it is likely that an attacker would only be interested in messages from one or a few correspondents. To partially account for this scenario, I divide email senders into two categories, important and non-important. More specifically, I define *important email senders* as those correspondents with whom a user has a significant social or work connection. I assume that users are most likely to notice messages from important mail senders; further, in many attack scenarios, these are the messages that are most likely to be targeted by an adversary. *Important messages* are the set of messages that are sent by important mail senders.

Note that these measures are inherently subjective; in my experiments, I determined these sets after discussions with each monitored user. For the four models (OU, OD, OUD and ON), from the important mail sender list of a user, I choose one as the particular mail sender who has sent a relatively large number of messages.

Though for a particular user a real attacker might behave differently from the above defined attack models, I believe that for most users the attack models have covered a large portion of the possible attack space. Which attack model should be selected to test user models? I discuss this issue further when I present the experimental results of the attack behavior models in Section 6.3.

In addition, in the defined attack models there is a lack of automated attackers. Automated attackers are malicious programs which could be used to search for sensitive information in email archives such as credit card numbers. Because their email access behaviors are distinct from normal users, they should be very easy for the system to detect. Therefore, I do not include them in my attack model design.

Attack Model	Scenario	Description
All-Unread (AU)	Scenario 1	Read all new messages and then mark them as unread.
All-Delete (AD)	Scenario 1	Read all new messages and then delete them.
All-Unread-or-Delete (AUD)	Scenario 1	Read all new messages and either delete or mark them unread.
All-Nothing (AN)	Scenario 1	Read all new messages only, no evasion.
One-Unread (OU)	Scenario 2	Read only new messages from a particular mail sender and then mark them unread.
One-Delete (OD)	Scenario 2	Read only new messages from a particular mail sender and then delete them.
One-Unread-or-Delete (OUD)	Scenario 2	Read only new messages from a particular mail sender and then either delete or mark them unread.
One-Nothing (ON)	Scenario 2	Only read new messages from a particular mail sender, no evasion.
Important-Unread (IU)	Scenario 3	Read only important messages and then mark them unread.
Important-Delete (ID)	Scenario 3	Read only important messages and then delete them.
Important-Unread-or-Delete(IUD)	Scenario 3	Read only important messages and then either delete or mark them as unread.
Important-Nothing (IN)	Scenario 3	Read only important messages, no evasion.
Nothing (NO)	Scenario 4	No operation is done on new messages.
Random-Operations (RO)	Scenario 5	Randomly operate on new messages.

Table 4.4: Description of the 14 attack behavior models. Note that the words“either” and “randomly” are explained in the part of Attack Behavior Models in Section 4.3.

Chapter 5

Implementation

My implementation includes two parts—collecting data and then analyzing it. Corresponding to the two parts, my programs are divided into an online data collection program and an off-line data analysis program. The data collection program is presented in Section 5.1. In Section 5.2, I discuss how the data is analyzed.

5.1 Data Collection

IMAP is an international email access protocol used by most mail clients (e.g., Outlook, Netscape Mail, Mozilla Mail) and many mail servers (e.g., UW IMAP server, Microsoft Exchange Server). Therefore, I collected IMAP request commands relative to the five chosen email disposition operations (reading, deleting, coping, moving and marking unread), which are sent by users from mail clients to IMAP servers. However, IMAP is not used by that very popular email access method—web mail. But even so, my results can also apply to web mail, as explained later in Section 5.2.

Privacy Protection. There are fundamental privacy concerns that arise in any situation where email activity is collected and analyzed. The collected data should not expose any confidential information about users. In my collected data, only one kind

```
login time: Mon Oct 25 15:12:19 2004
logout time: Mon Oct 25 20:32:19 2004


msg#        sender#        dispositions

111         25             12523
112         67             12
113         12             12
114         11             12523
```

Figure 5.1: Sample of one-session records in the activity file of the Ph.D. Student. Each row represents a new received message in that session. Each number in the disposition column represents an IMAP request command as shown in Table 5.1. In this sample, a disposition is actually represented by two or three IMAP requests sent by mail clients based on the user's actual operations. For example, the combination of numbers, 12, represents the disposition—reading, and the combination of numbers, 523, represents the disposition—deleting (see details in Table 5.5).

of information is related with user privacy—email addresses of senders. Therefore, email addresses of senders are hashed before they are output to hard disk. In a user's log file, an email address of a sender is represented in the form of a number. Later, I present how email addresses are hashed.

Data Collection. Each user has two log files—an email activity file and an email sender address file. The activity file includes email sender numbers of newly received messages and the corresponding IMAP request commands relative to the five dispositions. A user's activity is recorded when an IMAP session gets closed. The address file is used to store hashed addresses of all of the email senders who have ever sent messages to the user. All hashed addresses have the same length of 160 bits. As it is not convenient to repeatedly store such long numbers in a user's activity file, I use the position number of a hashed address in the address file as the sender's number in the activity file.

Samples of the two files of the Ph.D. Student user are shown as Figure 5.1 and Figure 5.2.

Hashed addresses of email senders

```
81569c6ccf50b26e8f73461a157b06537122b3fd

463e9c98ddbb1215bea1b21b6bb177a233aa79b0

f16a2e32973ce83d9ff62097e8045d7dd4683e7a

7f7b738462d84c9c60a0eddb9fd3c904932cbf2e

6d233205804858c5424f62385caa9a4b81d6bc4b

a08fb7d3093af1c2fc7a3e7bf8e3f73558f4d4fa
```

Figure 5.2: Sample of records in the address file of the Ph.D. Student. Each row has a 160-bit SHA-1 hash encoded as 40 hex characters and represents an email sender address.

#	IMAP command	Description	Dispositions
1	fetch body content	fetch body content of message	read
2	UID store seen	mark message as seen	read, move, delete
3	UID store delete	mark message as deleted	move, delete
4	copy not trash	copy message to a non-trash folder	move, copy
5	copy trash	copy message to trash folder	delete
6	UID store unseen	mark message as unseen	mark unseen

Table 5.1: The mapping between IMAP numbers used in the activity file and IMAP request commands.

Data Collection Codes. Data collection codes were embedded into the daemon file imapd.c of the University of Washington's IMAP server (version 2003.339) running in the Carleton Computer Security Laboratory (CCSL). The CCSL has 4 Linux servers and 10 desktops (2 Windows and 7 Linux). There are around 15 users in the CCSL. The data collection code starts to record a user's activity when a session is initiated and outputs the records to the file system when the session gets disconnected.

The data structures I added in the imapd.c file are described in Table 5.2. Table 5.3 shows the methods I added or modified for data collection.

Data Structure	Type	Description
yr_data1	array	to record a user's disposition operations on new messages
login_time	array	to store login time of a session
logout_time	array	to store logout time of a session
yr_dir	array	directory where records will be output

Table 5.2: The data structures added in the file imapd.c.

Function	Status	Function Description
convert	new	to convert character to number
yr_getunseenmail	new	to get new received email from mail store
yr_saveactions	new	to save disposition operations
mail_fetchfromaddr	new	to get email address of an email sender
sha1	new	to hash email address of an email sender
yr_outputfile	new	to output records to disk
hupint	modified	to output record to disk
trmint	modified	to output record to disk
inerror	modified	to output record to disk
fetch_work	modified	to get IMAP command—fetch content body
fetch_body_part_binary	modified	to output
ptext	modified	to output
main	modified	to record activity

Table 5.3: The methods added or modified in the file imapd.c.

5.2 Data Analysis

My data analysis program does three kinds of work—processing data, building models for each user and testing the models. The last two kinds of work are presented in Chapter 6. In the following, I discuss in detail why and how the data are processed before they are used for building models.

Data Processing. I gathered the IMAP request commands that were sent from a user's email client to the IMAP server. But instead of directly modelling the IMAP request commands, I model a user's actual operations on messages such as reading and deleting. As I have discussed in Section 2.1, a user's actual operations (e.g., reading, deleting) in his mail client are translated into IMAP request commands which are sent to the IMAP server.

Even though IMAP is an international standard, mail clients still have slight differences in what IMAP commands they send for the same user action. However, according to my observation, users' actual operations have much more impact on the variability of IMAP request commands than mail clients. Therefore, I believe that if pure user behavior (disposition operations) can be modelled, it should be easier to model the IMAP request commands themselves. In addition, modelling only pure user behavior actually makes the detection task harder, because with involvement of email clients, an attacker could be detected if he used a different email client from the targeted user's. Finally, because the same basic user operations would also be used when accessing an archive via a web interface or other access protocols, the results potentially translate to these other systems in addition to IMAP-based email archives.

Therefore, I extract the chosen five disposition operations from the IMAP command data such that variation in email client IMAP behavior was excluded. As mail clients translate users' operations slightly differently, I contacted each intended user

user	mail clients
Faculty	Mozilla Thunderbird [4]
Ph.D. Student	Evolution [1], Mutt [5]
Masters Student	Mutt [5]

Table 5.4: Mail clients used by each user.

User	Disposition	IMAP command sequences
Faculty	read	12, 2
	delete	523, 53, 3
	copy	4
	move	423, 43
	mark unread	6
Ph.D. Student	read	12, 1, 2
	delete	523, 53, 3
	copy	4
	move	423, 43
	mark unread	6
Masters Student	read	12, 1, 2
	delete	53, 3
	copy	4
	move	423, 43
	mark unread	6

Table 5.5: Translation table between each email disposition operation and IMAP requests. As each user uses different mail clients, the translations might be different.

to ask what mail clients he uses. And then I tested each intended mail client and made a translation table between each disposition operation and IMAP request commands for each mail client. Table 5.4 shows the mail clients used by the three monitored users (for details about the three users' data, see Chapter 6). Table 5.5 is the translation between each disposition operation and IMAP requests for each user, based on what mail clients the users use. Note that for the Ph.D. student, the translation map comes from the two mail clients she uses. Because there is no mapping conflict between the two mail clients, for example “1” meaning “unread” in one client but meaning “read” in the other, I use one mapping for both clients.

Chapter 6

Empirical Results

In this chapter, I use experiments to verify the feasibility of my user model and the parameters' influence on the user model. I present results from testing attack models, testing the feasibility of the user model, and testing parameters' influence on each user's model. Before presenting experimental results, I first discuss data sources and how experiments are set up in Section 6.1 and Section 6.2. In Section 6.3, I present an experimental analysis of 14 attack behavior models, one of which is used in the subsequent sections. User model feasibility is assessed in Section 6.4. The last section, Section 6.5, analyzes how different values of each parameter impact the effectiveness of each user's model.

6.1 Data Source

In September 2004, there were a total of twelve users in the CCSL, but only six of whom ever used their lab email accounts. Of the six users, three users regularly used their email accounts. I logged three months worth of IMAP server activity for these three users. The collected data sets are outlined in Table 6.1. The model was initially developed and tested using the data from the Faculty user; it was then further tested

user	email usage	#days	# msgs	# senders
Faculty	work & personal	85	3997	930
Ph.D. Student	work	84	484	202
Master's Student	sysadmin & work	65	2340	73

Table 6.1: Description of the three users' email account usage. # days: total number of days of data collected; # msgs: total number of new email messages received in the days; # senders: total number of distinct senders of email. "personal" means that email account is for personal use, for example, email messages from friends and family members. "work" means that email account is for work use, for example, email messages between student and professor. "sysadmin" means email messages sent by the systems in CCSL.

on the data sets from the two graduate students. While this user population is not large or comprehensive, the volumes, backgrounds, and purposes of the email received by these three individuals are all extremely varied, and as such this data has been sufficient for an initial evaluation of my approach.

6.2 Experiment Setup

6.2.1 Sliding Window

A user's message records are taken from his activity file and are put in a record stream. A sliding window mechanism is used to retrieve the email message records from the record stream. A record window slides sequentially along the record stream. Figure 6.1 shows how a window slides. A record window is divided into a long-term data window and a short-term data window. As discussed in Section 4.2.1, the short-term window represents a user's current monitored behavior while the long-term window represents a user's past behavior. By default, the size of the short-term window (p_{sw}) is set to be equal to one day's worth of message records of the user, and the size of the long-term window (p_{lw}) is set to be 20 days' worth of message records. Each time, a window slides past the number of message records in a user's short-term window

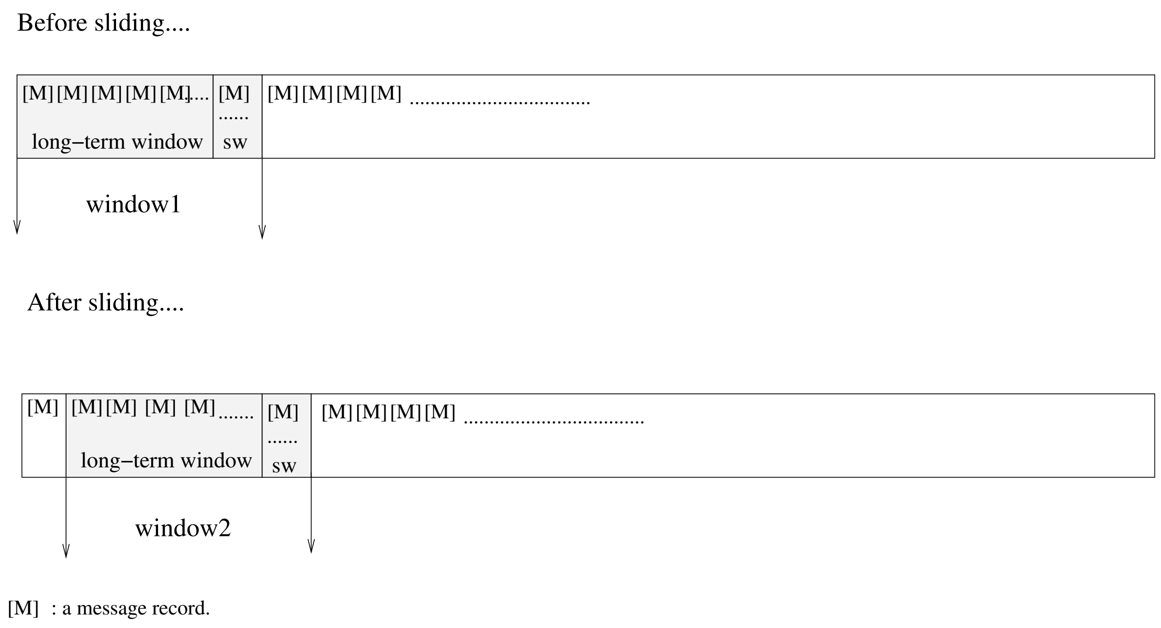
Before sliding....
[M] [M] [M] [M] [M]..... [M] [M] [M] [M] [M] ....
......
long−term window
sw
window1
After sliding....
[M] [M] [M] [M] [M]....... [M] [M] [M] [M] [M] ....
......
long−term window
sw
window2
[M] : a message record.

Figure 6.1: Window sliding mechanism. sw: short-term window; $[M]$: a message record.

(e.g., 40). Anomalies are then detected by comparing the behavior on one day to the user's average behavior over the 20 immediately preceding days.

Both the short-term window and the long-term window are composed of message records in the stream, as shown in Figure 6.1. Each message record contains two kinds of information: one is who sent the message; the other is how the message was disposed of by the user. A message record is presented as follows:

$$[M] = [sender\#, [O_r, O_d, O_c, O_m, O_u]]$$

The value of $sender\#$ represents the number of the email sender who sent the message. O_r,O_d,O_c,O_m and O_u respectively represent the user's five actual operations—reading, deleting, copying, moving and marking messages unread. Their values are normalized either 1 or 0, which has been explained in Section 4.2. Value 1 means that the user did the corresponding operation. Value 0 means that the user did not do the corresponding operation.

Para	Description
p_v	refer to which calculation method of message variation M is used
p_C	the least number of messages by a sender with full sender confidence C
p_W	associated with threshold setup
p_{sw}	the number of messages in short-term window
p_{lw}	the number of messages in long-term window

Table 6.2: Description of each parameter.

For each window, my analysis program does two things—it calculates the frequencies of each disposition of message records from each mail sender in the long-term data window; and it measures the window variation W (see Section 4.2.1) between the short-term window and the long-term window. Note that I do not attempt to prevent days with unusual behavior from being included in a user's long-term behavior data.

6.2.2 Parameters

There are a number of parameters employed by my user model and summarized in Table 6.2. The default values of each parameter are set up in Table 6.3. I also assign each parameter a value range as shown in Table 6.4 in order to adjust a set of optimal parameter values for each user's model. All these parameters have been explained in detail in Chapter 4. Value ranges are set up based on my preliminary experiments and knowledge of the model. The value ranges should be larger with a large number of users being monitored, because the large number of users might bring more variable behaviors. However, as I use three users' data for the preliminary experimental analysis, the value ranges are currently enough.

6.2.3 Experimental Methods and Types

In terms of experimental methods, the key difference from conventional IDS experiments is that my experiments are per-user based. In my experiments, a pair of false

user	p_v	p_C	p_W	p_{sw}	p_{lw}
Faculty	0	10	2.0	40	80
Ph.D. Student	0	10	2.0	5	100
Master's Student	0	10	2.0	30	600

Table 6.3: Default values of each parameter. The values of p_{sw} and p_{lw} are related to the average number of new received messages of a user in one day; therefore, they are different for each user. See Table 6.2 for each parameter's description.

parameter	value list
p_v (all users)	0, 1
p_C (all users)	1, 2, 10, 20
p_W (all users)	1.5, 2.0, 2.5
p_{sw} (Faculty)	40
p_{lw} (Faculty)	200, 400, 600, 800
p_{sw} (Ph.D. Student)	5
p_{lw} (Ph.D. Student)	25, 50, 75, 100
p_{sw} (Master's Student)	30
p_{lw} (Master's Student)	150, 300, 450, 600

Table 6.4: Value sets defined for each parameter. The values of p_{sw} and p_{lw} are related to the average number of new received email messages of a user in one day.

positive and true positive rates is calculated for each user. In the conventional IDS experiments, false positives and true positives are reported based on data from all users. The difference results from the different design of alarm response systems[1]. Conventional IDSs send alarms to administrators. In my user-specific design, an alarm is sent to the corresponding user, and therefore it makes more sense to calculate false positives for each user.

Three types of experiments were performed. They were attack model experiments, user model experiments and parameter experiments. Attack model experiments were used to choose the most appropriate attack behavior model for use in the other two types of experiments. User model experiments were used to test the feasibility of each user's model. Parameter experiments were used to test how distinct values of a parameter affect a user's model.

[1]Though I do not have a response system yet, I intend for my model to be used in this way.

I used two methods to evaluate these experimental results, one method directly using values of window variation W (referred to as the window-variation method), and the other using false positives and true positives (referred to as the conventional method). As a conventional method to evaluate models in the intrusion detection field, false positives and true positives are used to show the effectiveness of the whole system. However, from the window-variation method, we can learn about unusual events in more detail, such as on what exact dates the unusual events happened. As a result, we are able to analyze causes of the events. Therefore, the window-variation method can present the models at a lower level and in greater detail. These two methods can complement each other, helping us view the models from different perspectives.

In the window-variation method, the whole data set of a user is used to calculate values of window variation W between the short-term window and the long-term window in each sliding window. A number of window variations (W) are obtained for the user. In the conventional method (using false positives and true positives), a user's data set is divided into two parts: the first third of the data (approximately one month of data) are used to determine the value of the attack detection threshold (see Section 4.2.4); the latter two-thirds of the data are then used to determine true and false positive rates based on the mixed detection threshold. We refer to the former part of data as training data and the latter part as testing data. Table 6.5 shows the three experimental types as well as the evaluation methods used.

Note that the training data used to build my user model might include attacker activity data. In this case, I assume that attacker activity is only minor part of the training data, and can be considered as noise similar to user abnormal behavior.

Experiment Type	Evaluation Method	Data Division	Parameter Setting
attack model	window-variation	no division	default set
user model	window-variation	no division	default set
	false and true positives	dividing 1/3 and 2/3	whole value list
parameter	false and true positives	dividing 1/3 and 2/3	-

Table 6.5: Experiment types as well as their experiment methods, data division methods, and parameter settings.

6.3 Attack Model Analysis

Before I analyze the feasibility of the user model, I first need to study the fourteen attack behavior models (which have been presented in Section 4.3) through experiments and then choose the most appropriate one as the attacker behavior for testing the user model. There are two requirements for the most appropriate attack model—detection difficulty for the system and detection difficulty for users. More specifically, the most appropriate attack model should be the one that is more (or the most) difficult for the system to distinguish from user behavior compared with other attack models. In so doing, I assume that if the system can detect the most evasive simulated attacker, it can also detect the others. Meanwhile, users themselves have difficulty in detecting the real attackers who have the same behavior as the most appropriate attack model. I assume that users are a sort of monitor function, as they are able to notice some obviously unusual changes in status, such as a new mail being marked as read.

I evaluate the difficulty of system detection by comparing values of *window variation difference* ΔW caused by each of the fourteen attack models and each of the three users. In each window, the calculation of ΔW is

$$\Delta W = W_a - W_u$$

where both W_a and W_u refer to W window variation defined in Section 4.2.3. W_a is the window variation caused by an attack model and W_u is the window variation of a user's normal behavior. For both of the values, I use the same set of default parameter values (see Table 6.3) and the same profiled behavior calculated by sliding the long-term window across all available user data. To calculate W_u, I use user data for the short-term windows; for W_a, though, the user's short-term behavior is replaced with attacker operations based upon the chosen model. The smaller the ΔW is, the harder it is for the system to distinguish between the attack behavior model and user behavior.

Figure 6.2 shows ΔW for the fourteen attack models and each user. From this figure, we can see that the values of ΔW are smallest for the attack models AD, AUD, and AN (for all attack models, refer to Table 4.4); thus, these attack models are the hardest to distinguish from normal user behavior. According to these results, AD and AN are closer to user behavior than AUD; these two attack models, however, are of minimal concern, as they can be discovered by the users themselves. With attack behaviors of the model AD, attackers read all new mail messages and then delete them. The users should suspect that something is wrong when the number of newly received mail messages is much less than normal. With attack behavior of the model AN, attackers read new mail messages and then just leave them there. Virtually all mail client programs have a function of marking as read to a message which has been viewed. With the obvious mark, users can detect the attackers themselves and therefore, I leave detection of this kind of attack to users. In contrast, it is harder for users themselves to discover an attacker who is behaving like the model AUD (reading all new mail messages and then either marking them unread or deleting them). Because the attacker deletes only some new email messages, the difference in the number of new messages is not significant and the user is less likely to notice that change. As the attacker marks the other new messages unread which have been read by him, those messages look like normal new messages. Because there is no obvious unusual change left by the attacker, it is hard for the user herself to detect the attacker. **Thus, I chose the model *AUD* as the attack model for evaluating the detection ability of my user model.**

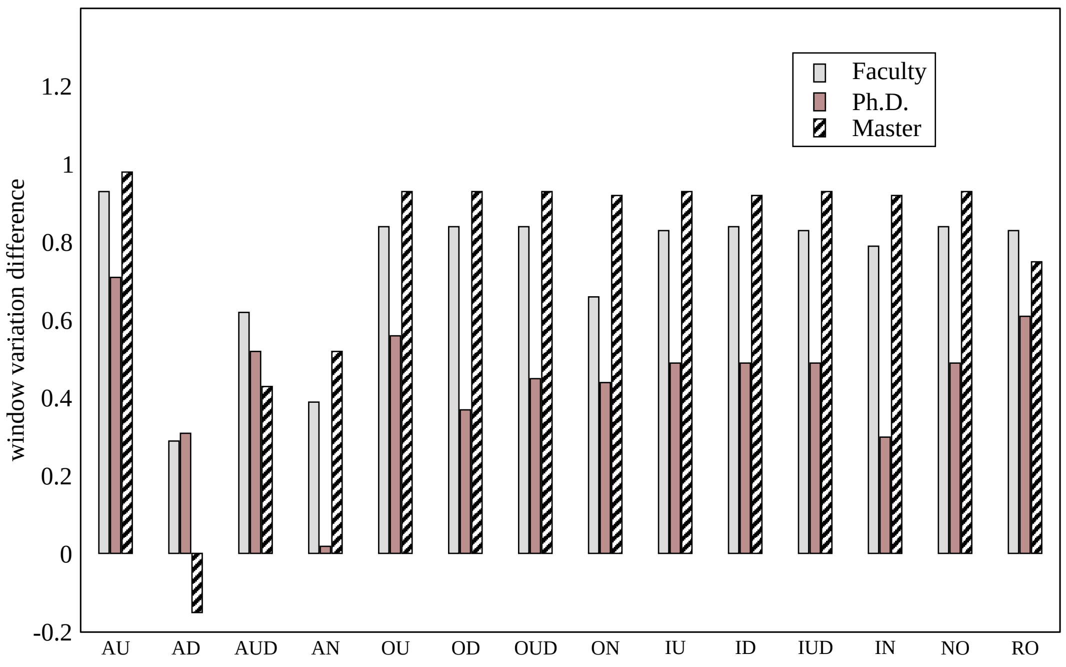

Figure 6.2: *Window variation difference* ΔW between the fourteen attack models and users. Each group of bars represents an attack model marked on the X axis. The Y axis represents ΔW. In each group of bars, the three bars in order respectively represent *window variation difference* ΔW between the attack model and each user (the Faculty, the Ph.D. Student, the Master's Student). With the attack model *AD* (refer to Table 4.4), the negative bar of the Master's Student means that window variation caused by the attack model is less than one caused by the Master's Student.

6.4 User Model Analysis

To evaluate my user model in this section, the experiments are divided into three parts: the first part is used to test the feasibility of each user's model based on the default experiment setup; another two parts are used to assess my user model in a more practical environment in order to complement the default experiment setup. In the default experiment setup, I assume that the system updates a user's profile every day (seeing the long-term window sliding mechanism). I also assume that the system measures a user's behavior once a day (seeing the default size of the short-term window). Therefore, the last two parts of the experiments are used to test how often a user's profile should be updated and whether the system is able to detect attackers from the monitored data mixed with user and attack behaviors.

6.4.1 User Model Feasibility

In this part, I've focused on testing the feasibility of each user's model using the two evaluation methods—pairs of false positives and true positives, and the window variation comparison method. The reasons for using the window variation method are explained in Section 6.2.

In order for a model to be feasible, it must be able to detect a significant number of attacks while also generating no more false positives than a regular user could be expected to handle. What is the criteria for a feasible model in the email domain in terms of false positives and true positives? Although email carries some sensitive information (e.g., bank information) sometimes, in most cases messages are about users' work and daily life. Therefore, I think that it should be acceptable to most users that the system can detect one intrusive access out of two (or a detection rate 50%). As for false alarm period, it should be acceptable for most of users to be bothered by false alarms once a week. Of course, the real acceptable false alarm period and detection rate for a specific user will depend on the user's security needs.

ROC curves, False Positives And True Positives. Receiver operating characteristic (ROC)[2] curves are used to analyze the trade-off between false positives

[2]ROC analysis was originally developed in the field of signal detection. More recently, besides the intrusion detection field, it has been used in fields as diverse as language and speaker identification and medical risk prediction [38].

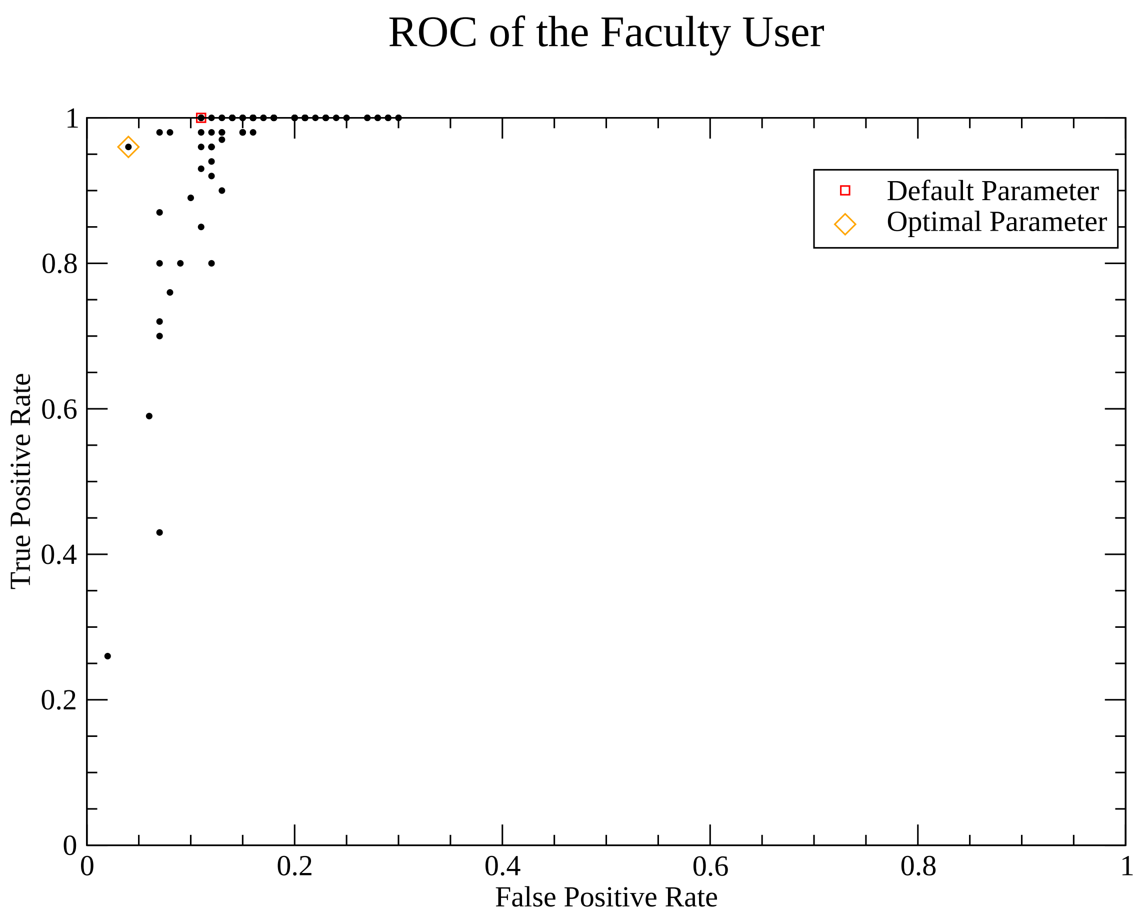

Figure 6.3: ROC plot for the Faculty.

and true positives for IDSs. ROC curves can indicate how detection rates change as internal thresholds are varied to generate more or fewer false alarms.

Figure 6.3, Figure 6.4 and Figure 6.5 are ROC curves for each user based on 96 parameter sets, with one point per parameter set. These parameter sets are combinations of the values listed in Table 6.4. With each parameter set, I use the user's training data to determine his detection threshold, and then use his testing data and the detection threshold to calculate a false positive rate and a true positive rate. Each point in the user's ROC represents his false positive rate (X axis) and true positive

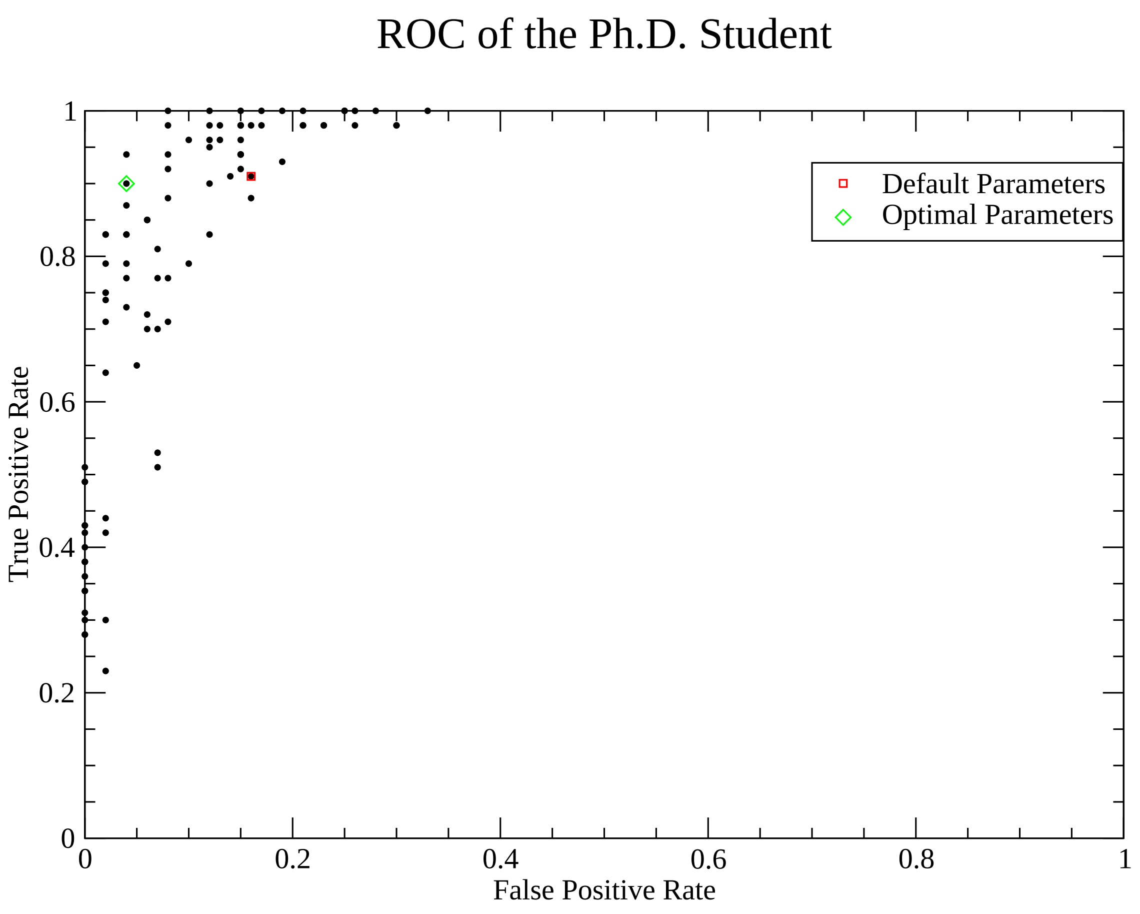

Figure 6.4: ROC plot for the Ph.D. Student.

rate (Y axis) based on each parameter set.

For the Faculty and the Ph.D. Student, most of the points are located in the upper left corner, with false positive rate less than 20% and true positive rates more than 80%[3]. For the Master's Student user, however, the points are spread much more evenly across the range of false positives. These plots show that my model is a relatively accurate representation of the behavior of the Faculty and the Ph.D.

[3] I think that the values, 20% and 80%, should be acceptable for most users. The false positive rate, 20%, also means that the user will be bothered by false alarms once every five days. The translation has been explained later.

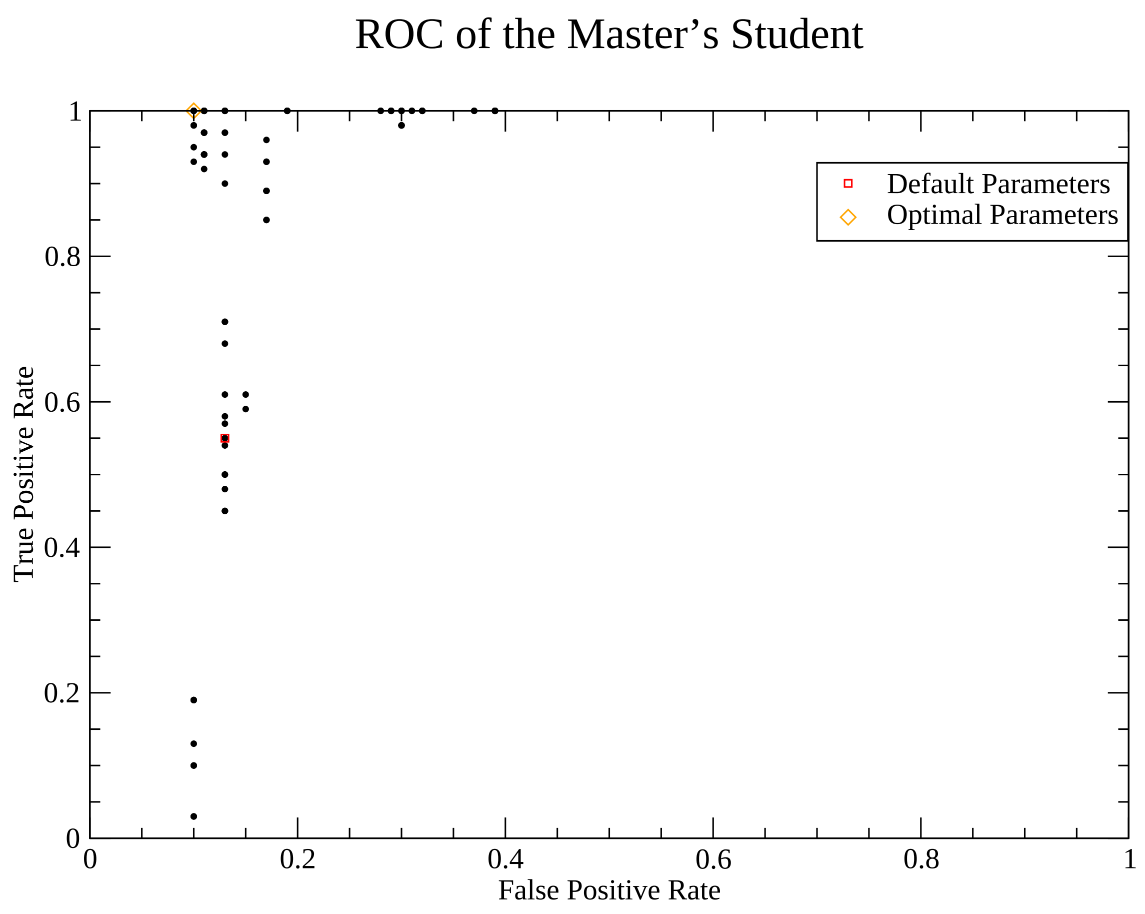

Figure 6.5: ROC plot for the Master's Student.

student; the email dispositions of the Master's Student, however, do not appear to be significantly determined by the sender of the email message. This discrepancy can be explained by the fact that this student receives a large number of automated messages for systems administrative purposes, and these messages originate from a small number of (non-human) email senders.

To further understand how these false positive rates ($f.p.$) would translate into alarms that a user would have to assess, I also analyzed the false alarm period ($f.a.$). The false alarm period ($f.a.$) represents the number of days that will pass on average

user	p_{sw}	p_{lw}	p_v	p_C	p_W	$f.p.$	$f.a.$	$t.p.$
Faculty(dp)	40	800	0	10	2.0	0.11	9	1.0
Faculty(op)	40	800	0	20	2.5	0.04	25	0.96
Ph.D Student(dp)	5	100	0	10	2.0	0.16	6	0.91
Ph.D Student(op)	5	75	1	20	2.0	0.04	25	0.90
Master's Student(dp)	30	600	0	10	2.0	0.13	8	0.55
Master's Student(op)	30	300	0	10	2.5	0.1	10	1.0

Table 6.6: The three users' results respectively based on default and optimal parameter settings. *dp*: default parameter set; *op*: optimal parameter set; $f.p.$: false positive rate; fa: false alarm period, the number of days that are passed on average when an false alarms happens; $t.p.$: true positive rate. The parameters are described in Table 6.2.

between false alarms. As the size of short-term data is set to be the average number of new messages received each day, the $f.a.$ is equal to $1/f.p.$.

Table 6.6 shows every user's false positive rate ($f.p.$), average false alarm period ($f.a.$), and true positive rate ($t.p.$) for default and optimal parameter settings. I tested all combinations of parameters in Table 6.4 for each user and chose the parameter set which produced the best performance (low false positives and high true positives) as the optimal parameter set. From this table we can see that for all users, false alarm periods are relatively long, in that a user would only be bothered once a week at most, and for the "better behaved" users, they would only be bothered roughly once a month (on average). Yet, even with these settings, over half (and generally, almost all) attacks would be detected.

Comparison of Window Variation. In this part, I further evaluate the feasibility of the user model through a direct comparison of window variations caused by users and the attack model AUD.

Figure 6.6 shows three pairs of window variation curves respectively caused by each user and the attack model AUD. There is a pair of curves in each user's figure—the bottom one shows the values of window variation caused by the user; the upper one shows the values of window variation caused by the attack model AUD. Each value of the X axis represents a window containing a short-term window and a long-term window. Each value in the X axis corresponds to two values in the Y axis: the round-form one represents user window variation; the diamond-form one represents window variation of the attack model AUD. All points are based on default parameters. The greater the separation of the two curves, the easier for the detector to distinguish between the user behavior and the attack behavior model AUD. In the Faculty's figure, the two curves are well separated, with only a short overlapping period. It has been verified that in that short period, the Faculty was away for Christmas vacation and he did not dispose of his email as usual. Curves in figures of the Ph.D. Student and the Master's Student are not as good as the Faculty's, but overlapping is infrequent.

From the two types of analysis, we can conclude that the models of the two graduate students are not as good as the Faculty's, though they are still acceptable. The results are associated with the way they used their email accounts. The Ph.D. Student only used her email account for work, and she only received an average of five new email messages per day. Therefore, the size of her short-term window is setup to five-message behavior by default. The size 5 is too small for accurate statistics, because only one message behavior variation can result in large window variation. For the Master's Student, though he received an average of 30 new mail messages per day, most messages were automatically sent by the systems in the CCSL lab. As most of those system email messages were not that important or urgent, he dealt with them when convenient, instead of in a timely way. I believe that if the two graduate students had used their email accounts as fully as the Faculty user, their models would have been better.

6.4.2 Profile Updates

One question in intrusion detection systems is how often a profile should be updated. Profile updates should be conservative to prevent attackers from slowly training the

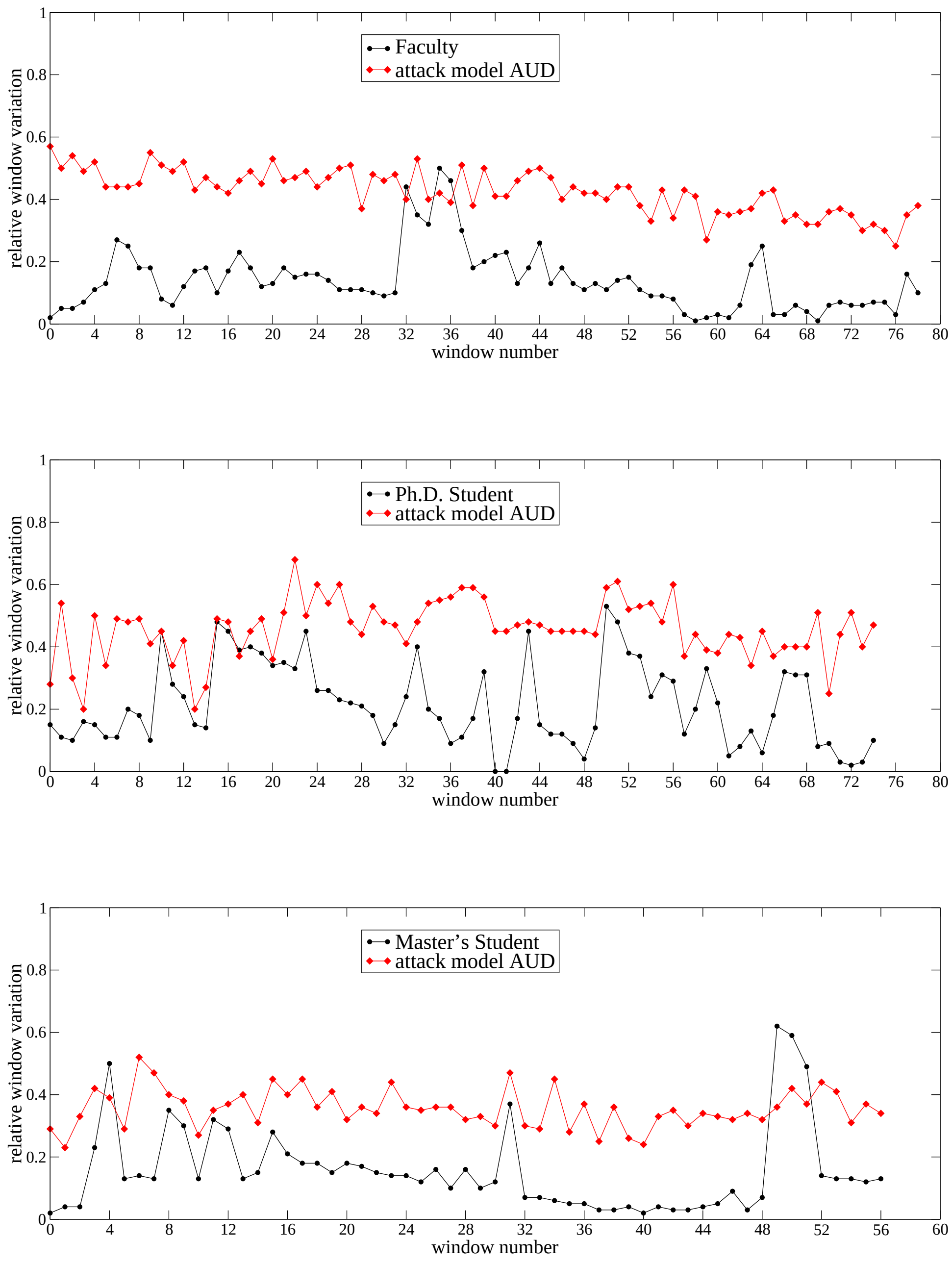

Figure 6.6: Window variation comparison between each user and the attack model *AUD*. **Top**: the Faculty; **Middle**: the Ph.D. Student; **Bottom**: the Master's Student.

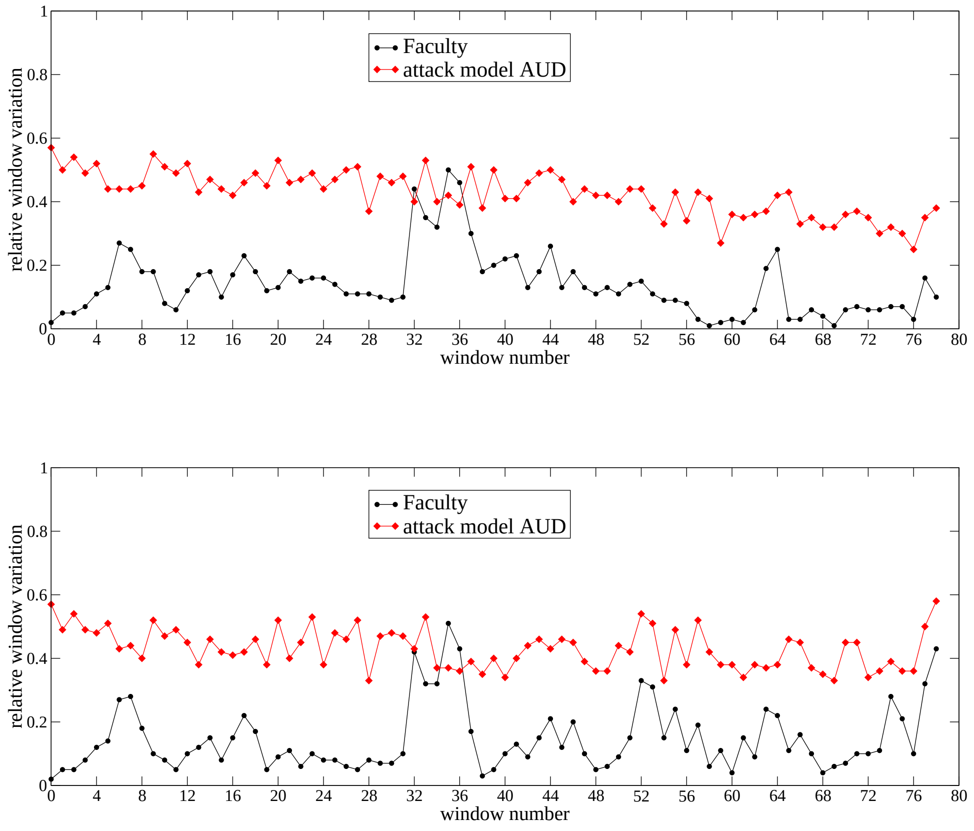

Figure 6.7: Testing profile update. **Top:** the default experiment setup—the profile is updated every day. **Bottom:** the new experimental setup—the first month data is profiled without further updates.

system. Updating a profile should only occur when a user's disposition habits change. In my default experiment setup, I assume that a profile is updated every day, and therefore with the sliding window, it is updated each time the long-term window slides past a user's average number of newly received messages in one day (size of the short-term window). In this part, using the Faculty user as a testing target, the first month's data is used as a permanent profile and his subsequent two-months' worth of data as monitored behavior data which are divided into a number of short-term data windows. The short-term windows are each compared with the profile. Figure 6.7 shows pairs of window variation curves caused by the default experiment setup (a user's profile being updated every day) and by the new experiment setup (the profile not being updated).

The figure shows that both experimental setups make the two curves (by user and by the attack model AUD) separate very well. Therefore, I can conclude that at least for some users, their profiles can be updated once every couple of months.

6.4.3 Mixed Behaviors

In my default experiment setup, I assume that the system does the detection task once a day, and therefore the size of short-term window is by default the average number of messages a user receives in one day. Because it is likely that attackers will manipulate the archive on the same day as it is legitimately accessed, the monitored data is likely mixed with behaviors of the user and the attacker. This experiment is used to assess the detection ability of the system given that the monitored data is the mixed behavior. I assume that the attacker accesses the archive and then leaves before the legitimate user gets into work. Therefore, the first half of the messages in a short-term window are assumed to be disposed by the attack model (AUD) and the next half by a user.

Table 6.7 shows that detection rates of the system decrease by half compared

user	monitored behavior	*f.p.*	*t.p.*
Faculty	default	0.11	1.00
Faculty	mixed	0.11	0.59
Ph.D. Student	default	0.16	0.91
Ph.D. Student	mixed	0.16	0.51
Master's Student	default	0.13	0.55
Master's Student	mixed	0.13	0.13

Table 6.7: Testing detection ability of the models by using monitored data (short-term data window) mixed with behaviors of users and the simulated attackers. “default”: monitored data only with attack behavior of the attack model AUD; “mixed”: monitored data mixed with behaviors of users and the attack model AUD.

with the default experiment setup. But for the Faculty and Ph.D. Student, the new detection rates are still acceptable—0.59 and 0.51, which mean that attackers can be expected to be detected after breaking in twice (two sessions). The system detects once a day, because short-term is defined as one day. For example, an attacker accesses a user's email a couple of times respectively in the first and second day, and he will be detected in the second day according to the detection rates. But for the Master's Student, the new detection rate is not acceptable. From this, I can conclude that the models of the Faculty and the Ph.D. Student are more robust, because they can still detect attackers even with mixed behavior data.

6.5 Parameter Analysis

My model consists of four parameters—the short-term window (p_{sw}), the long-term window (p_{lw}), the calculation method of message variation (p_v), and the sender confidence (p_C). Each parameter reflects part of my model. Conventionally, in the intrusion detection field, people justify a model according to false positives and true positives. Here, I consider whether we can observe a model from the parameter perspective. In this section, I test each parameter to see how different values affect false positives and true positives of each user's model.

user	p_{sw}	f.p.	t.p.
Faculty	40 (default)	0.11	1.00
	30	0.16	0.98
	20	0.11	0.98
Ph.D. Student	5 (default)	0.16	0.91
	10	0.00	0.80
	20	0.00	0.33
Master's Student	30 (default)	0.13	0.55
	20	0.13	0.64
	10	0.12	0.68

Table 6.8: Influence of different sizes of short-term data window on each user's model. Other parameters are set to the default values.

6.5.1 Short-Term Window Size p_{sw}

What is the optimal size of the short-term window (monitored behavior data)? If it is too small, it will be hard to measure in terms of statistics. If it is too big, the data will easily get mixed with user behavior and attack behavior (which has been discussed in Section 6.4.3). The mixed data will make the detection more difficult.

Table 6.8 shows the impact of different values of short-term window on false positives and true positives for each user's model. The range of values for the short-term window is set up based on actual email usage of each user (average amount of newly received email messages in one day). For the Faculty user we can see that the three sizes of short-term window (40, 30 and 20) do not make a big difference in false positives and true positives. So the size of short-term window can be set up as small as 20. For the Ph.D. Student, the value 10 is best, as it produces good values for both false positive and true positive. For the Master's Student, the value 10 is best. Therefore, I conclude that bigger values (e.g., 30, 40) do not necessarily produce better performance. However, too small a value (e.g., 5) will result in higher false positives.

user	p_{lw}	*f.p.*	*t.p.*
Faculty	800 (default)	0.11	1.00
	600	0.16	1.00
	400	0.20	1.00
	200	0.16	1.00
Ph.D. Student	100 (default)	0.16	0.91
	75	0.08	1.00
	50	0.02	0.83
	25	0.06	0.72
Master's Student	600 (default)	0.13	0.55
	450	0.11	1.00
	300	0.29	1.00
	150	0.17	0.85

Table 6.9: Influence of different values of long-term data window size on each user's model. Note: other parameters are set to the default values.

6.5.2 Long-term Window Size p_{lw}

Large long-term window size results in long training time. If it is too small, it will not be accurate in future behavior prediction. The profile with a small number of messages would be better if its prediction ability is not greatly affected. In Table 6.9, each user is assigned four different values of long-term window size according to his actual email usage. From the table, we can see that for the Faculty user, the value 800 gives better performance than the other three in terms of false positives and true positives. For the Ph.D. Student, the three values—75, 50 and 25, offer better results of false positive rates than the value 100. Considering both false positives and true positives, the value 50 is the best option for her. For the Master's Student, the value 450 offers the best results of false positives and true positives. Therefore, I conclude that a large long-term window is not necessarily better than a small one.

6.5.3 Sender Confidence p_C

For the parameter, sender confidence(p_C), all users are set up with the same range of values—20, 10, 2 and 1. Table 6.10 shows experimental results. For both the Faculty

user	p_C	*f.p.*	*t.p.*
Faculty	20	0.11	1.00
	10 (default)	0.11	1.00
	2	0.15	1.00
	1	0.17	1.00
Ph.D. Student	20	0.12	0.95
	10 (default)	0.16	0.91
	2	0.16	0.98
	1	0.19	0.93
Master's Student	20	0.13	0.55
	10 (default)	0.13	0.55
	2	0.13	0.58
	1	0.13	0.71

Table 6.10: Influence of different values of the parameter p_C on each user's model. Note: other parameters are set to the default values.

user and the Ph.D. Student, we can see that the larger the parameter value is, the lower the false positive is. Therefore, the value 20 is the best option for them or 10 for the Faculty. For the Master's Student, the parameter values only have impact on true positives. The value 1 is good for the Master's Student.

6.5.4 Calculation Method of Message Variation p_v

There are two value options for the parameter p_v—0 and 1. Table 6.11 shows that in terms of false positives, with 9% (0.16-0.07) improvement on the Ph.D. Student's model, the parameter has more impact on her model than the other two users. Therefore, for the Ph.D. Student, p_v should be setup as 1. For the other two users—the Faculty and the Master's Student, p_v does not have much impact on false positives, and however, it slightly affects true positives. Therefore, $p_v = 0$ is a little better for the Faculty while $p_v = 1$ is better for the Master's Student.

user	p_v	$f.p.$	$t.p.$
Faculty	0 (default)	0.11	1.00
	1	0.11	0.85
Ph.D. Student	0 (default)	0.16	0.91
	1	0.07	0.81
Master's Student	0 (default)	0.13	0.55
	1	0.13	0.68

Table 6.11: Influence of different values of the parameter p_v for each user. note: other parameters are set to the default values.

6.5.5 Conclusion about Parameter Analysis

From the above experimental analysis based on each single parameter, it seems that I can make the following conclusions: different parameter values have different impacts on each user's model; and a user's model might be sensitive to some parameter values while not being affected by others; in terms of model effectiveness, the system should customize parameters for each user through optimization. However, the optimization will result in extra computation cost. Because the optimization may improve effectiveness of the user model, I think that it is worth customizing each user's model with a little extra computation cost.

Chapter 7

Discussion

The past several chapters documented the user-specific design, the email user behavior model implementation, and the testing of the model. This chapter reviews these results and places them in perspective. I first explain the limitations of my work and then discuss the implications.

7.1 Limitations

There are a number of limitations in my work that must be considered.

First, only part of the user-specific design is currently implemented. The user-specific design needs further verification of its feasibility with support from mathematics theory. In order to make the design more concrete, more features need to be explored and more sub-models need to be built based on the features. Each user should be modelled not only at the level of dynamically-determined parameters but also at the level of dynamically-chosen sub-models according to the design. It is a question of balancing computation cost with fidelity to the user behavior model.

Second, my experimental results are based on a very small user population (3 users). I do not believe, however, that my model (or indeed, any simple model of user behavior) could ever apply to all users. Indeed, results on one of my test subjects (the Master's Student) show that for some users, email dispositions do not correlate well with email senders. These results are sufficient, however, to conclude that my modelling strategy has a good chance of working for some high-volume email users. In the future, more users' data will be collected to further estimate the accuracy of my model in a large population.

Third, it should also be noted that a number of assumptions were made when the data was analyzed. Users tend to receive a highly variable number of messages per day, but I assumed that users receive a constant stream of messages. In addition and more importantly, I assume that the detector checks each user's activity records once a day. As a result of these two assumptions, the size of the short-term window (representing a user's current behavior data) in the experiments was assigned as the average number of messages received in one day. In reality, attackers might access users' accounts some time during a day (e.g., one hour in the middle of a day). Therefore, the short-term data defined in that way could easily be mixed with user behavior and attacker behavior. It is difficult for the detector to recognize attacker behavior in the mixed behavior data. In the future, to avoid the mixed behavior data, the time frame for short term behavior should be reduced to a single session[1]. Also, it is possible that a user has multiple simultaneous sessions (e.g., originating from different machines) during a single time period. In this case, only one session short-term data out of the multiple sessions can not represent the user's complete behavior on new messages. Therefore, the session-based short-term data should include all of the simultaneous sessions. In summary, a practical environment is more complex than my experimental environment.

Fourth, in practice, I would not want to exclude email clients' behaviors from

[1]Actually, I collected data based on per session (see Figure 5.1). But I analyzed the data based on per day.

the IMAP request command data, as a user's email clients can be considered as part of his behavior patterns. In addition, it is not practical to extract each user's pure behaviors, because an email system can have thousands of users who would use many kinds of mail clients and the extraction algorithm used in this work requires manual translation between IMAP requests sent by a mail client and users' actual operations.

Fifth, though I believe that my approach to simulating attack behaviors is systematic, there is still a significant gap between realistic attacker behavior and the chosen attack model (AUD). Can I measure or estimate this gap, and can I improve upon it (i.e. reduce it) in a meaningful way? This is an important question for future work, both for this problem and for other approaches to user behavior-level anomaly intrusion detection.

Sixth, my modelling methods are based on Euclidean Distance. There might be better metrics to use. In the definition of M', I ever tried to put ${\Delta'_u}^2$ outside the square root and found that the experimental results based on the three users' data are worse than those inside the square root. Also, I think the operation, marking unseen, should not have more impact on the model than other operations. Therefore, I put it inside the square root. In addition, I assume that most mail senders send one message during the period of the short term, and therefore, in my model, frequencies in the short-term data are normalized to either 1 or 0. In reality, some users might often receive multiple messages from a mail sender during the short term period. I do not know whether it will matter in my models or not.

Seventh, my work is only concerned with user access patterns on new messages. Actually, attackers also likely read old messages especially in the first time that they intrude. Actually, this system can more easily detect attackers, who only read old messages and do not touch new messages, than those attackers who read new messages. It is very abnormal behavior that a user is not interested in his new messages. The experiments on the attack model NO have verified this (see Figure 6.2). However, I still think that it is necessary to monitor user access pattern on old messages in order to make the system more effective. Misuse intrusion detection is a good option to monitor user behavior on old messages because of two observations. First, if legitimate users access old messages, they most likely want to get a particular bit of information, and therefore, they more likely view those old messages from a mail sender or a couple of mail senders than each message; but attackers most likely access each message from all kinds of mail senders, especially on their first access. Second, legitimate users most likely view old messages very fast because they have read them before; however attackers will read old messages slowly because they are new for the attackers.

7.2 Implications

While I was doing this work, I found it difficult to conduct experiments and build models. I tried to look for related books or references that could provide me with theoretical guidance on these two aspects. Unfortunately, I failed to find these kinds of books or references. Therefore, I first discuss the implications of my experiments and modelling, and then discuss the applications to user-specific design.

7.2.1 Experimental Methodology

It is well-known that anomaly IDSs depend on experiments to verify their effectiveness because it is hard (or impossible) to use pure mathematical methods to verify that their models are good or not. However, experimental methodologies can have significant impact on the quality of experimental results (e.g., models' false positives and true positives). For the same model, different experimental methodologies can bring about different results. Currently, however, the anomaly IDS field is lacking in specific instructions or standards on experimental methodologies, and researchers in the field do experiments in their own ways. Based on my work, I summarize from the following three aspects.

First, for user behavior-level anomaly IDSs, in order to evaluate models' effectiveness, besides user behavior data, attacker behavior data is also needed. However, it is hard to obtain real attack behavior data. The only reliable way to obtain the data is to simulate attack behaviors. How attack behaviors are simulated is important in terms of correctly assessing a model. Careless simulation of attackers may provide results that are too good to be true. The simulated attackers not only affect detection rates but also false positives rate, because there are generally trade-offs between false positive rates and true positive rates. It is a problem about how to scientifically simulate attack behaviors so that the simulated attack behaviors are likely to represent real attack behaviors. Though many researchers work on defining intrusive behavior rules for misuse IDSs or design attack scenarios for the purpose of software vulnerability testing, to my knowledge, no researchers have worked on scientifically designing attack behavior models for the purpose of testing anomaly IDSs. However, I think that the work related to attack behavior definition for other purposes can be used to design attack models for testing anomaly IDSs.

Second, there is lack of guidance about experimental procedures such as the division of data for training and testing. I think experimental setups can also have impact on experimental results. For the same model, different experimental setups might bring about different results. Therefore, when we read experimental results in a paper, we need to consider whether the experimental setup is reasonable. It is not easy for readers, who do not have much experience in this aspect, to give a judgement. Other scientific fields such as medicine have a set of uniform standards about experimental procedures. Experiments which do not comply with the standards are not accepted regardless of the experimental results. Compared with other fields, the anomaly IDS field covers a broader scope in terms of data representation, which can be user behaviors, system calls, and so on. Therefore, it might be difficult to define a set of general standards to be applied for all categories. But we might be able to define a set of standards for each category (e.g., user behavior-level anomaly IDS) according to the characteristics of each category.

Third, I think that assessment of experimental results in the anomaly ID field is subjective, mostly depending on the researchers' own explanation. Normally, researchers use false positives (or false alarm rates), true positives and ROC to assess a model. But I believe that only if we are very sure that attack behavior data used for testing can represent real attackers can a system be assessed through the single pair of false positive rate and true positive rate. Both false positive rates and true positive rates can be affected by the attack behavior data because of the trade-offs between them. For example, if attack behavior data comes from a kind of silly attacker, who can be easily detected, researchers can easily get good experimental results (low false positives and high detection rates) by setting up an easy threshold. With that threshold, the system is only able to detect that kind of silly attacker and has nothing to do with other kinds. However, for most anomaly IDSs (not only user behavior-level IDS), it is hard (or impossible) for researchers to obtain complete attack data even though they can obtain a couple of real ones (e.g., a few types of real worms or viruses). Therefore, I think probably we should use some other assessment methods to complement the pair of false positive and true positive rates. In my experiments, besides false positives and true positives, I also use some low-level data (window variation) to present the feasibility of my models. In addition, false alarm rates are a better measure of feasibility analysis of a system than false positives. However they should be explained clearly in terms of standard units. Further, I think that the criteria of acceptable false positives and true positives should depend on a specific application, as each application might have different effectiveness requirements. Finally, how does one evaluate a ROC, by distributions of points or curve forms? Is it possible to make assessment criteria for ROC curves?

7.2.2 Modelling

In the anomaly ID field, some researchers directly apply some existing modelling methods (e.g., neural networks), which have been used in other fields; some researchers tailor modelling methods for their IDSs such as NIDES [11]; some researchers modify existed modelling methods such as Lane's work [37]. Though anomaly ID work is tightly connected with modelling methodologies, there is no reference book about how to build models for anomaly IDSs.

During the time I spent on this research, I consulted people in the statistics department and I also tried to look for help from some statistics books. However, neither of them was very helpful, because statistics is so large a field that those people working on a specific statistical field do not really know the needs of the anomaly ID field. Also I was not able to quickly get answers to my questions from the books because I do not have any special knowledge of statistics. However, statistics, as the most used modelling method in the anomaly ID field, has also been used in many fields such as economics and medicine. For each field, there are corresponding statistical subfields such as economics statistics and biological statistics. The anomaly ID field has its own requirements for modelling, such as real-time events (low computation cost). Therefore, a specialized subfield in statistics devoted to anomaly ID would be useful.

However, compared with other statistical application fields, it might be more difficult to form a set of general statistical theories or instructions for the anomaly ID field, because of the variety of data representations in the field, where data used for modelling can be system calls, network packets, user behaviors, or others. However, we can divide the field into further sub-fields according to data source and provide modelling tools for each sub-field.

The statistical-modelling tools for each sub-field can include the following information: common statistics errors which can easily occur in modelling; general instruction for threshold setup, how to deal with unscalable data sizes; and good modelling examples of anomaly IDSs.

7.2.3 The User-Specific Design

User behavior-level anomaly intrusion detection is one type of defense against insider attackers. Its key disadvantage is high false positives. In order to lower false positives, researchers work on selecting precise features, building effective models and designing domain-specific architectures. My user-specific design, with the same purpose of lowering false positives, takes advantage of not only the email domain characteristics but also human beings' strength, which has been discussed in detail in Chapter 3. Any other domain which has the same key characteristics as the email domain—one data file only belonging to one user, such as web bank application—can apply the user-specific design for its anomaly IDS against insider attackers. As for applications like databases where data files are shared and accessed by multiple users, only the first part of the design—user-tailored modelling—can be applied, because that part only derives from inherent properties of human beings' behaviors, and has nothing to do with the email domain. In fact, for any user behavior-level anomaly IDS, the ideas of the user-tailored modelling part in the design can be considered.

The anomaly ID field has existed for many years; however most work has the typical drawback—high false positives, especially the work on user behaviors. Therefore, we need to think about the anomaly ID from a different perspective. While I was doing this work I thought a lot about human beings' natural behaviors. I think that nature is the origin of inspiration. Some researchers work on associating natural things (e.g., human immune system) with IDSs. I feel that might be the right starting point for IDS evolution.

Bibliography

[1] Evolution. http://go-evolution.org/Main_Page [Accessed: June 2005].

[2] Honeypots. www.honeypots.net [Accessed: June 2005].

[3] Introduction to Genertic Algorithm. www.rennard.org/alife/english/gavintrgb.html [Accessed: Nov. 2005].

[4] Mozilla Thunderbird. www.mozilla.org/projects/thunderbird [Accessed: June 2005].

[5] Mutt. www.mutt.org [Accessed: June 2005].

[6] Norton AntiVirus. http://www.symantec.com [Accessed: June 2005].

[7] Procmail. www.procmail.org [Accessed: June 2005].

[8] Sendmail. www.sendmail.org [Accessed: June 2005].

[9] Smart Card News. http://www.smartcard.co.uk [Accessed: June 2005].

[10] The Spamhaus Project, 2005. http://www.spamhaus.org [Accessed: June 2005].

[11] D. Anderson, T. Frivold, and A. Valdes. Next-Generation Intrusion Detection Expert System (NIDES): A Summary. Technical Report SRI–CSL–95–07, Computer Science Laboratory, SRI International, May 1995.

[12] Apache Software Foundation. SpamAssassin, 2005. http://spam-assassin.apache.org [Accessed: June 2005].

[13] A. Boukerche. Security and Fraud Detection in Mobile and Wireless Networks. *Handbook of Wireless Networks and Mobile Computing*, pages 309–323, 2002.

[14] R. Buschkes, D. Kesdogan, and P. Reichl. How to Increase Security in Mobile Networks by Anomaly Detection. In *ACSAC'98: Proceedings of the 1998 Annual Computer Security Applications Conference*, pages 3–12, 1998.

[15] P. Chan and S. Stolfo. Toward Scalable Learning with Non-Uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection. In *Knowledge Discovery and Data Mining*, pages 164–168, 1998.

[16] C. Chung, M. Gertz, and K. Levitt. DEMIDS: Misuse Detection System Database System. In *Integrity and Internal Control in Information Systems (IICIS)*, 1999.

[17] Cisco Inc. Identified Internet Mail. http://www.identifiedmail.com [Accessed: June 2005].

[18] M. Crispin. Request for Comment (RFC) 3501: Internet Message Access Protocol—Version 4rev1, March 2003. http://www.faqs.org/rfcs/rfc3501.html.

[19] M. Delany. Internet Draft: Domain-based Email Authentication Using Public-Keys Advertised in the DNS (DomainKeys), March 2005. http://www.ietf.org/internet-drafts/draft-delany-domainkeys-base-02.txt.

[20] T. Dierks and C. Allen. Request for Comment (RFC) 2246: The TLS Protocol Version 1.0, January 1999. http://www.ietf.org/rfc/rfc2246.txt.

[21] W. DuMouchel. Computer Intrusion Detection Based on Bayes Factors for Comparing Command Transition Probabilities. Technical report, National Institute of Statistical Sciences (NISS), USA, 1999.

[22] E.Allman, J.Callas, M.Delany, J.Fenton, and M.Thomas. DomainKeys Identified Mail (DKIM), October 2005. http://mipassoc.org/dkim/specs/draft-allman-dkim-base-01.txt [Accessed: Nov. 2005].

[23] Email Marketer. http://www.emaillabs.com/resources_statistics.html [Accessed: June 2005].

[24] Google. Gmail, 2005. http://gmail.google.com [Accessed: June 2005].

[25] A. Gupta and R. Sekar. An Approach for Detecting Self-propagating Email Using Anomaly Detection. In *RAID 2003 Proceedings*, volume 2820 of *LNCS*, pages 55–72. Springer-Verlag, 2003.

[26] J. Hall, M. Barbeau, and E. Kranakis. Anomaly-based Intrusion Detection Using Mobility Profiles of Public Transportation Users. In *WiMob: Wirless and Mobile Computing, Networking and Communications*, 2005.

[27] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A Network Security Monitor. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1990.

[28] Hewlett-Packard. HP-Fraud Management System, 2003. http://www.hp.com.

[29] S. Hofmeyr. *An Immunological Model of Distributed Detection and its Application to Computer Security.* PhD thesis, University of New Mexico, 1999.

[30] H. Inoue and S. Forrest. Anomaly Intrusion Detection in Dynamic Execution Environments. In *NSPW '02: Proceedings of the 2002 New Security Paradigms Workshop*, 2002.

[31] Internet Security Systems Inc. RealSecure Internet. http://www.iss.net/prod/rsds.html [Accessed: June 2005].

[32] J.Thorpe and P. van Oorschot. Graphical Dictionaries and the Memorable Space of Graphical Passwords. In *13th USENIX Security Symposium*, 2004.

[33] H.-A. Kim and B. Karp. Autograph: Toward Automated Distributed Worm Signature Detection. In *USENIX Security Symposium*, pages 271–286, 2004.

[34] C. Ko, G. Fink, and K. Levitt. Automated Detection of Vulnerabilities in Privileged Programs by Execution Monitoring. In *Proceedings of the 10th Annual Computer Security Applications Conference*, pages 134–144, Orlando, FL, 1994. IEEE Computer Society Press.

[35] C. Kreibich and J. Crowcroft. Honeycomb - Creating Intrusion Detection Signatures Using Honeypots. In *Proceedings of the Second Workshop on Hot Topics in Networks (Hotnets II)*, November 2003.

[36] C. Krugel, T. Toth, and E. Kirda. Service Specific Anomaly Detection for Network Intrusion Detection. In *SAC '02: Proceedings of the 2002 ACM Symposium on Applied Computing*, 2002.

[37] T. Lane. *Machine Learning Techniques for the Computer Security Domain of Anomaly Detection.* PhD thesis, Purdue University, 2000.

[38] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman. Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition.* DARPA, 2000.

[39] M. Mahoney and P. Chan. PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic. Technical Report CS-2001-04, Florida Institute of Technology, 2001.

[40] R. Maxion and T. Townsend. Masquerade Detection Using Truncated Command Lines. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, 2002.

[41] Microsoft. Microsoft Outlook. http://www.office.microsoft.com [Accessed: June 2005].

[42] Microsoft. Hotmail, 2005. http://www.hotmail.com [Accessed: June 2005].

[43] J. Myers and M. Rose. Request for Comment (RFC) 1939: Post Office Protocol—Version 3, May 1996. http://www.faqs.org/rfcs/rfc1939.html.

[44] M. Oka, Y. Oyama, H. Abe, and K. Kato. Anomaly Detection Using Layered Networks Based on Eigen Co-occurrence Matrix. In *RAID 2004 Proceedings*, volume 3224 of *LNCS*, pages 223–237. Springer-Verlag, 2004.

[45] P. Porras and P. Neumann. EMERALD: Event Monitoring Enabled Responses to Anomalous Live Distrubances. In *National Information Systems Security Conference*, 1997.

[46] B. Ramsdell. Request for Comment (RFC) 2633: S/MIME Version 3 Message Specification, June 1999. http://www.ietf.org/rfc/rfc2633.txt.

[47] A. Roddy and J. Stosz. Fingerprint Features—Statistical Analysis and System Performance Estimates. *Proceedings of IEEE*, 85:1390–1421, 1997.

[48] M. Schonlau, W. DuMouchel, W. Ju, A. Karr, M. Theus, and Y. Vardi. Computer Intrusion: Detecting Masquerades. *Statistical Science*, 16(1):1–17, 2001.

[49] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated Worm Fingerprinting. In *SOSP03:19th ACM Symposium on Operating Systems Principles*, August 2003.

[50] S. Singh, C. Estan, G. Varghese, and S. Savage. The EarlyBird System for Real-time Detection of Unknown Worms. Technical Report CS2003-0761, University of California, August 2003.

[51] S. Smaha. Haystack: An Intrusion Detection System. In *4th Aerospace Computer Security Applications Conference*, 1988.

[52] A. Somayaji. *Operating System Stability and Security through Process Homeostasis*. PhD thesis, University of New Mexico, 2002.

[53] S. Stolfo, C. Hu, W. Li, S. Hershkop, K. Wang, and O. Nimeskern. Combining Behavior Models to Secure Email System. Technical report, Columbia University, April 2003.

[54] B. Sun, F. Yu, K. Wu, and C. M. Leung. Mobility-based Anomaly Detection in Cellular Mobile Networks. In *WiSe '04: Proceedings of the 2004 ACM Workshop on Wireless Security*, 2004.

[55] C. Taylor and J. Alves-Foss. An Empirical Analysis of NATE: Network Analysis of Anomalous Traffic Events. In *NSPW '02: Proceedings of the 2002 New security paradigms Workshop*, 2002.

[56] H. Vaccaro and G. Liepins. Detection of Anomalous Computer Session Activity. In *Proceedings of IEEE Security and Privacy*, May 1989.

[57] K. Wang and S. Stolfo. Anomalous Payload-based Network Intrusion Detection. In *Proceedings of RAID'04*, Sep. 2004.

[58] WheelGroup Corporation. Brochure of the Netranger Intrusion Detection System. http://www.wheelgroup.com [Accessed: June 2005].

[59] D. Whyte, P. van Oorschot, and E. Kranakis. Addressing Malicious SMTP-based Mass-Mailing Activity Within an Enterprise Network. Technical Report TR-05-06, Carleton University, 2005.

[60] M. Wong and W. Schlitt. Internet Draft: Sender Policy Framework (SPF) for Authorizing Use of Domains in E-MAIL, version 1, June 6 2005. http://www.ietf.org/internet-drafts/draft-schlitt-spf-classic-02.txt.

[61] P. Zimmerman. *The official PGP user's guide.* MIT Press, Cambridge, MA, 1995.