Securing Email Archives through User Modeling

Yiru Li and Anil Somayaji
School of Computer Science, Carleton University
1125 Colonel By Drive, Ottawa, ON K1S 5B6 Canada
{ylic,soma}@scs.carleton.ca

Abstract

Online email archives are an under-protected yet extremely sensitive information resource. Email archives can store years worth of personal and business email in an easy-to-access form, one that is much easier to compromise than messages being transmitted "on the wire." Most email archives, however, are protected by reusable passwords that are often weak and can be easily compromised. To protect such archives, we propose a novel user-specific design for an anomaly-based email archive intrusion detection system. As a first step towards building such a system, we have developed a simple probabilistic model of user email behavior that correlates email senders and a user's disposition of emails. 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 behavior.

1 Introduction

Email is one of the cornerstone applications of the Internet, one that millions use on a daily basis. Email, unfortunately, is also an extremely insecure medium for communications. Messages lack authentication, confidentiality, and integrity guarantees while servers are extremely vulnerable to denial-of-service attacks. The spam epidemic has spread in part due to these shortcomings; in practice, however, email continues to provides "sufficient" security guarantees in that a high enough percentage of legitimate emails reach their intended destinations intact, and confidential information contained in those messages is not often used for malicious purposes.

The spread of reliable network connectivity and the declining cost of storage, though, have together changed the way that email is managed, and in so doing have changed the nature of threats facing a user's inbox. Before, email was typically downloaded from a mail server to a workgroup server or local disk. Now, however, email is more often being archived on servers that are accessible from the open Internet either via a web interface or through an online mail access protocol such as IMAP [6]. Through these servers, users can access both new and archived email from potentially any networked computer in the world. The price for this convenience, unfortunately, is that *anyone else* in the world may also access this same resource. By storing hundreds of megabytes of filtered, prioritized, and organized email, remotely-accessible email servers provide a means for an adversary, whether a business rival, criminal, or jealous lover, to learn about a target with minimal effort or risk. Most such parties do not have the technical skill or access to compromise emails while in transit; almost everyone, though, is capable of accessing an Internet-connected email archive.

Despite the increased exposure users now have with large, remotely accessible email archives, the authentication technology used to protect such resources is still extremely weak. While there exist many technologies that could be used to authenticate email users in a more secure fashion, virtually all of them would require significant changes in how users access their email. For now, we appear to be stuck with the same standard that has existed since the beginning of the Internet: reusable text passwords. Even when systems do not transmit passwords in cleartext (which they often do, even today), passwords may be compromised by malicious software, social engineering, or by simple user negligence.

To address this challenge, in our research we have focused on how to better protect email repositories even when account passwords have been compromised. One way to achieve this goal is through intrusion detection system (IDS) technology. Unlike many other IDSs, however, an email archive IDS must defend against attackers who do not exploit software vulnerabilities; instead, they will be using normal access protocols to retrieve unauthorized informa-

tion. This is the "insider threat" problem, one that was an early focus of intrusion detection research. Like systems such as NIDES [1], we propose to detect intrusions by detecting unusual uses of valid authentication credentials by maintaining statistical profiles of individual user behavior. We propose, however, that an email archive IDS should be *user specific*, in that it should be controlled by the profiled user herself, not by a systems administrator or security officer. Individual users would receive alarms and adjust parameters, giving them control over their level of security. 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.

As a first step towards such an email archive IDS, we have developed and tested a simple statistical model of user email behavior based upon the relationship between the disposition of new email messages and the senders of those emails. By limiting the scope of our model to this simple relationship, we have managed to create a model that is both surprisingly accurate and efficient to create. In testing over several months, we have found that after training for approximately one month, our system can distinguish between variations in user behavior over the next two months and simulated attacker behavior with a low rate of false positives—as low as one alarm per month, but generally not higher than one per week.

While our testing involves only a few users, and is thus far from comprehensive, we believe that our results provide sufficient evidence that our chosen modeling strategy is a viable one. Perhaps more importantly, it also demonstrates the feasibility of online anomaly intrusion detection through (suitably constrained) observable user behavior. Having said that, we do not believe that one simple model, no matter how good, will be able to accurately capture the patterns of behavior for all users. By developing multiple simple models and allowing the system to choose the most suitable one based upon its accuracy, it should be possible to accurately model the behavior of almost all users of email, and in so doing protect their archives from attack.

The rest of this paper proceeds as follows. We first discuss related work in Section 2. We then discuss the motivation and requirements of an email archive IDS in Section 3. We explain our choice of observable and modeling strategy in Section 4. Experimental setup and results are discussed in Section 5. The paper ends in Section 6 with a discussion of limitations, implications, and plans for future work.

2 Related Work

Electronic mail, or email, is a communications medium that, by the definition of its underlying protocol SMTP [12], offers almost no security guarantees. Numerous technical mechanisms have been proposed and developed in response to these limitations. Some, such as PGP [27] and S/MIME [18], use cryptography to provide a complete solution for end-to-end integrity, confidentiality, and authenticity. Most commonly-used email security enhancements, however, are more narrowly focused on the problem of unsolicited email, or spam. Some spam solutions prevent the use of forged email senders [7, 26]; others block known sources of spam through frequently updated lists of offending DNS names and IP address ranges [23]. In the end, though, many ISPs and email users resort to content-based spam filtering systems [2]. Similar content-based scanning is also used to detect and stop email viruses, either on servers or client systems. While such approaches help maintain the security of individual messages (and prevent the relaying of unwanted or dangerous ones), they do not secure messages once they have been delivered to a user's inbox.

The problem of email archive security has normally been subsumed within the general host protection problem. In that vein, both "server" and "client" protection systems have been built. For servers, we have buffer overflow mitigation systems [5], network [19] and host [21] intrusion detection systems, file integrity checkers [11], firewalls, and other protections. Client systems often employ many of these same mechanisms, but they are generally coupled with tools that prevent compromise by viruses and spyware. Whether focused on monitoring patterns in network traffic, process behavior, or file contents, though, such systems are primarily designed to prevent malicious or compromised software from circumventing access control policies—not prevent the use of compromised authentication credentials.

Unfortunately, the most commonly used authentication credentials, reusable passwords, are extremely vulnerable due to common patterns of user behavior. 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 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.

Protection against insider attacks—attacks from sources that have access to valid authentication credentials—is difficult to achieve using most commonly-used security technologies. Because insider attacks consist of "authorized"

accesses and manipulations of data, access control mechanisms are not sufficient to prevent them. It is extremely difficult to make an a-priori specification that would exclude all insider attacks while allowing legitimate accesses; similarly, insider attacks are difficult to characterize with fixed signatures.

Because of these fundamental difficulties, most insider-focused IDSs are based on some form of anomaly detection. For example, NIDES [1] monitored host audit records to detect anomalous user accounts. There has also been work on detecting insider attacks using anomalous typing patterns [3] and unusual patterns of UNIX shell commands [8, 20, 17, 15, 13]. In order to catch insider attacks, these systems build models of individual user behavior. Although they can detect a wide variety of compromises, many of these systems have been marked by high rates of false positives. In large part, these problems are not so much attributable to the failure of the systems themselves but rather to the inherent variability of human behavior. Other work in anomaly intrusion detection have avoided these problems by modeling program behavior [22] or network traffic [9, 10, 25]. While such systems often have lower false alarm rates, they are not nearly as adept at detecting insider attacks.

While (to our knowledge) there has been no work specifically on protecting email archives from attack, the problem is most analogous to that of misuse of database information by insiders. Existing proposals such as DEMIDS [4] build profiles of how individual users access a given database. While email archives have similarities to other databases, the fact that the stored information belongs to exactly one user makes the problem much more tractable. Further, the regular, well-known structure of email archives enables a simplified, specialized approach that should be feasible in practice. The fundamental challenge to developing such a system lies in determining what data to capture and how that data should be represented. The next section explains our approach.

3 Protecting Email Archives

To protect against the exploitation of email archives through disclosure of passwords, we propose that email archives be protected using a user-specific email archive intrusion detection system. Unlike host or network IDSs that are designed to protect one or more computers, we believe that an email archive IDS should be designed to protect one resource: a user's email repository. Logically, an email archive server then would actually be running multiple IDSs, with one instance per user. This design choice is largely motivated by the extremely personal nature of email; it also, however, has significant impact on our overall system architecture, modeling strategy, and the potential scalability of the system.

More specifically, we have framed our work on this problem with the following threat model. First, we assume that the attacker has access to a user's entire hardware and software environment: either the attacker uses the same platform, e.g. Microsoft Outlook running on Windows XP, or the attacker can access the user's machine itself. We also assume that the attacker has by some means obtained the user's password. Most of the data the attacker wishes to access, however, does not reside on the client machine; instead, it is stored on a remote server that is running a remote email archive access protocol such as IMAP or a web email interface[1]. Other than the targeted user's authentication credentials, the attacker has no other access to the server (i.e. no administrative/root access). The attacker wishes to compromise the archive in arbitrary ways that violate confidentiality (reading old and new emails) and/or integrity (deleting, modifying, or inserting emails). Our goal as defender, then, is to detect the attackers actions before they do significant damage.

To protect a user's email archive given these constraints, we would like an intrusion detection system that can detect the behavior of an attacker relative to that of a legitimate user. Legitimate users, however, are not all the same—being human, each will have her own patterns of archive access. Patterns that are legitimate for one user will be completely anomalous for another, e.g. one user might archive all organizational announcement emails, while another deletes them all immediately. Because the difference between legitimate and illegitimate behavior in this context is fundamentally ambiguous and related individual user behavior, we have chosen to approach this intrusion detection problem from the perspective of user behavior modeling.

Most production and research IDSs, however, do not focus on user behavior for a simple reason: false positives (false alarms). Human behavior inevitably changes over time, even using the most consistent observables. Eventually this "drift" manifests as a significant change in behavior (i.e. an alarm) unless the system takes steps to adjust to novel user behavior. Any such adjustments, however, must be very conservative, otherwise an attacker could simply "train" the system to accept her behavior as normal. The consequence of such conservatism is a persistent level of false positives in any user-specific anomaly intrusion detection system.

Clever choices of data sources, representations, and modeling algorithms can reduce the rate of false positives; the presence of false positives, however, inherently limits the scalability of any intrusion detection system. For ex-

[1] While POP [16] has some similarities to IMAP, it is primarily designed for downloading messages from an email server, not for managing a remote email archive. Because of this difference, we do not address POP further in this paper.

ample, 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). Because existing user-based anomaly IDSs tended to have higher rates of false positives than other IDS technologies, their deployments have traditionally been limited to environments where large numbers of security officers are available such as secret government agencies and military organizations.

If we make the entire system user specific (not just the models), then we have a ratio of one user to one monitoring party. With this change, user modeling becomes inherently more feasible because we can tolerate a higher rate of false positives; further, because a user understands his own behavior, he is the best equipped to determine whether or not a given alarm reflects a genuine security violation. This choice also puts the party with the most at stake—the user—in a position to make appropriate trade-offs regarding their level of protection vs. the amount of work required for monitoring the IDS.

We have found, though, that it is difficult for a simple model of user behavior to apply equally accurately to all users (see Section 5). While we cannot be certain that all simple models have this limitation, the inherent variability of human behavior suggests that this will be the case. To overcome this problem, a production email archive IDS system would need to incorporate multiple models of potential user behavior, with some of these models based upon different sets of observables. The system would then decide which model(s) to use to capture normal behavior during the system's initial training period. Thus, the model proposed in Section 4 should not be seen as applying to all users, but merely as a proof-of-concept model that applies to a reasonable fraction of potential users.

Because an email archive IDS would interact 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 alternate communication channel to transmit alarms such as instant messaging, mobile phone text messaging, or an alternative email account. (In this context, the IDS in effect filters side-channel messages to reduce user fatigue.) Another, potentially simpler alternative would be a clear alert in the mail reading client/web browser that cannot be removed by the user; instead, it would persist on-screen for a fixed period of time (e.g. a few days or a week). 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.

Of course, before we can consider the feasibility of building an email archive IDS, we first need to determine whether user behavior within this domain is consistent enough to form the basis of an anomaly-based IDS. To address this question, we need at least one candidate method for modeling user behavior. The next section, Section 4, explains our chosen modeling approach in detail, and Section 5 presents the results of offline experiments that test the performance of our model.

4 Modeling User Email Behavior

In this section we explain our approach to modeling email archive access behavior. Section 4.1 explains the intuitions that underly our model design. Section 4.2 presents the features used by our behavioral model. Section 4.3 describes the model itself. To evaluate this model, we need attacker behavior data in addition to normal user behavior; Section 4.4 explains our approach to modeling attackers for testing purposes.

4.1 Intuition

To design an anomaly-based email archive IDS, we first need to decide what to observe about email archive access behavior and how to model those observations. We have many possible observables, such as the time of day of archive accesses, the amount of data transferred, and the program used to access the archive. Rather than look at such incidental features, we have chosen to focus on the pattern of email disposition, i.e. whether an individual user chooses to read an email, delete it, forward it, etc. Because there are only a small set of possibilities for email disposition, this restriction greatly simplifies the task of modeling user behavior. At the same time, though, these operations represent the fundamentals of user decisions regarding email.

In this context, we 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 emails, however, are frequently read, deleted, or responded to soon after they have arrived. Because of this observation, we have chosen to build our model of user behavior using only the disposition of newly arrived emails.

The other key observation we have is that there is a naturally high correlation between the sender of an email message and its final disposition. Some email originating from an announcement email list might be read and deleted; email from an important colleague, though, would typically

be responded to and/or archived. Our model of user behavior, thus, correlates new email disposition with the sender of individual messages.

To test our model, we need to understand how an attacker might choose to mask her behavior. If an attacker is familiar with the habits of an individual, it might be possible for her to disguise her attack by imitating the targeted user's archive access behaviors. The problem for the attacker, though, is that typical message disposition actions change the state of the archive: email messages are moved, deleted, or are marked as being read. If an attacker attempts to do exactly the same operations as those of the normal user, because email accounts are normally not shared, the user herself will notice the unauthorized changes (messages marked as read that have never been seen by the user, mysteriously moved messaged, etc.). An attacker thus must both conceal her actions from the IDS *and* from the targeted user. This dilemma greatly simplifies the design of an email archive IDS system; we explore this idea further in Section 4.4 and Section 5.2.

4.2 Feature Extraction

There are a number of user disposition operations related with new email messages. Rather than build a model that incorporates all of these possibilities, we have instead chosen to focus on the following common operations: *reading, deleting, copying, moving, marking a message as being unread (setting the unread flag)*. If none of these actions are taken, we assume that the message was left unread in the user's inbox. For each newly received email message for a given user, we record the sender of the email (from the "From:" line of the message) along with which (if any) of the selected disposition options was performed. Note that to reduce data storage requirements and simplify our model, we ignore the parameters of these operations, e.g. the destination folder for a move operation.

For the reported experiments, we gathered this information by monitoring the IMAP requests that were sent from a user's email client to an IMAP server. Because clients can send different patterns of IMAP requests in response to the same basic user action, we tested various email clients and created a translation table that maps specific IMAP requests to the intended user action. This translation actually makes the detection task harder, because without it (i.e. if we built our model using IMAP commands directly) an attacker could be detected if he used a different email client from the targeted user's. Our intent, however, has been to evaluate the specific feasibility of modeling user interactions with an email archive. Because the same basic operations would also be used when accessing an archive via a web interface or other access protocol, our results potentially translate to these other systems in addition to IMAP-based email archives.

sender #	k (# msgs)	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	0.0
3	20	0.0	1.0	0.0	0.0	0.0

Table 1. Example of long-term behavior data.

4.3 The Model

Each user has a profile consisting of two parts, a long-term record of the frequency of different email dispositions for each email sender, and a set of dynamically-determined general model parameters. 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 behavior. When those changes exceed a fixed threshold, that short-term behavior is considered to be anomalous.

A user's short-term behavior patterns are assessed using a fixed number of the most recent messages received by that user. By default, the size of short-term data window is set to a user's average number of new messages received in one day. Long-term behavior patterns are stored using a larger sliding window that hold a user's email dispositions for a larger set of past messages. By default, this second window is twenty times the size of the short-term data window, or equivalent to approximately the number of messages received in twenty days. Both of these windows store frequency information for each monitored email disposition (reading, deleting, copying, moving, setting the unread flag).

To assess the difference between short-term and long-term trends in behavior, we define three measures, message variation M, sender confidence C, and window variation W. They are defined as follows.

4.3.1 Message variation M

The *message variation* M is the Euclidean distance between the disposition of a given message in recent short-term data and the average disposition (within the long-term profile) of past messages from that same sender. More specifically, we place dispositions within a five dimensional space, with each dimension corresponding to a particular type of message disposition: read r, delete d, move m, copy c, set unread flag u. A single recent message s in short-term data is represented in this space as a point with a 1 for each chosen disposition and 0 otherwise. The long term behavior record for a user, then, is simply the set of points $\{l\}$ representing the average disposition over k past messages for each email sender. The *message variation* M is defined in terms

of the distance between the short term behavior s (a single message's disposition) and long term behavior l as:

$$\Delta = s - l$$

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

For example, assume that a user's long term behavior profile is described in Table 1. This hypothetical user has received 38 messages from three distinct email senders. For the first email sender, based upon a sample of ten messages, this user generally reads messages (80% of the time), but sometimes deletes (20%) or just marks a message as unread (10%). Next, assume that this user receives a new message from sender #1 which the user decides to read and delete. Then, the message variation M for this message is:

$$\begin{aligned} \Delta &= \langle (1-0.8)_r, (1-0.2)_d, (0-0)_m, \\ &\quad (0-0)_c, (0-0.1)_u \rangle \\ M &= \sqrt{0.2^2 + 0.8^2 + 0^2 + 0^2 + (-0.1)^2} \\ &\approx 0.8307 \end{aligned}$$

In addition to this definition of M, we also tested an alternative definition for M that used a modified Euclidean distance metric; we found, though, that this alternative metric was generally much less effective than the one outlined above. When $p_v = 1$, then this alternative metric was used; otherwise, the default definition for M was used. For further information on this alternative M, see [14].

4.3.2 Sender confidence C

The numbers 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 modeling requires an adequate number of samples; to determine whether we have seen a sufficient number of messages for predictive purposes, we need a per-sender measure of our confidence in the model.

To measure this confidence, which we refer to as *sender confidence* C, we divide the number of messages k received from an email sender by a fixed user-specific threshold p_C. We also set the maximum possible C value to be 1. In other words, we define C as:

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

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

Model	Description
AU	Read all new msgs., mark as unread.
AD	Read all new msgs., delete.
AUD	Read all new msgs., delete or mark unread.
AN	Read all new msgs. only, no evasion.
IU	Read important msgs., mark as unread.
ID	Read important msgs., delete.
IUD	Read important msgs., delete or mark unread.
IN	Read important msgs. only, no evasion.

Table 2. Description of the eight models of attacker email reading behavior.

user	email usage	days	msgs.	senders
Fac.	work, personal	85	3997	930
Ph.D.	work	84	484	202
M.S.	work, sysadmin	65	2340	73

Table 3. Description of the three users' data sets, listing number of days of data collection, total number of messages received, and the number of distinct email senders.

4.3.3 Window variation W

Window variation W represents the difference between short-term behavior data and long-term behavior data. It is defined in terms of the values of message variation M of all messages within the short-term behavior window and their corresponding sender confidence C. Specifically, we define W as:

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

Here, i ranges over all messages in the short-term window.

To detect anomalies, the W of short-term data is compared to $\overline{W}$, the average window variation during long-term training. If $W > p_W \overline{W}$, then an anomaly is signalled. p_W is a user-specific parameter that by default is set to 2.

4.4 Simulated Attack Behaviors

Because it is extremely difficult to obtain attack data on specific email accounts, we have tested our model using simulated attacker behavior. As there is generally a trade-off between false positives and true positives in anomaly detection systems, the choice of attacker simulation method directly affects the interpretation of our results.

As part of developing our simulation strategy, we defined fourteen types of attacker behavior models based upon four attack scenarios [14]. Here, though, we focus on the eight attack models that were determined to be the most difficult

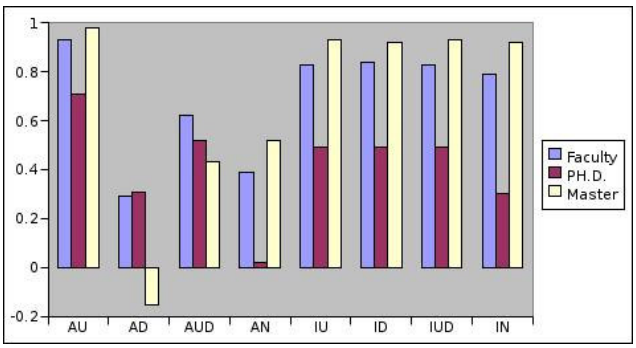

Figure 1. *Window variation difference* ΔW between the eight attack models and users. Each group of bars represents an attack model marked on X axis. Y axis represents ΔW. The three bars in each group represent ΔW between the given attack model and each user.

for users to detect on their own. These models are described in Table 2. When an attacker must choose between deleting or marking a message as unread (attack models AUD and IUD), we choose either option for each message with probability 0.5.

In the first four models, 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 a few correspondents. To partially account for this scenario, we divide email senders into two categories, important and non-important. More specifically, we define *important email senders* as those correspondents with whom a user has a significant social or work connection. We 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 our experiments, we determined these sets after discussions with each monitored user.

While real attacker behavior will generally be much more complex than that outlined in these models, an attacker's options are greatly constrained by need to avoid detection by both the IDS and the targeted user. We discuss this issue further when we present attack simulation results in Section 5.2.

5 Experiments

This section presents the results of the experiments used to develop and analyze our model of user email disposition behavior. The first part explains our experimental setup. The next section describes our empirical evaluation of the attacker models described in Section 4.4. The final part presents our overall evaluation of our model's feasibility in terms of true and false positives.

parameter	values
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 member)	40
p_{lw} (Faculty member)	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 4. Values defined for each parameter. lw and sw are the number of messages in the long-term and short term message windows, respectively. The other parameters are defined in Section 4.3.

5.1 Experimental Setup

There are fundamental privacy concerns that arise in any situation where email activity is monitored and analyzed. Ideally, analysis should be done automatically by programs that will not store or expose confidential user information. In practice, though, we needed to analyze manually user email behavior in order to develop our simple model.

To resolve this dilemma, we developed and tested our model using a small user population that would give consent to this type of monitoring and who could provide useful feedback on detected anomalies. More specifically, we monitored user email disposition behavior of three users on the IMAP server running in the Carleton Computer Security Laboratory (CCSL). For these users, we logged three months worth of IMAP server activity using a modified version of the University of Washington's IMAP server [24]. As mentioned earlier, the chosen five email dispositions were extracted from this IMAP data such that variations in email client IMAP behavior was excluded. The collected data sets are outlined in Table 3. Our model was initially developed and tested using data from the faculty user; it was then further tested on 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 these results appear sufficient for an initial evaluation of our approach.

To simplify analysis, we assumed that an equal numbers of messages were received on each day; thus, we could divide up the data set into a set of equal-sized "days" for analysis purposes. The short-term window (p_{sw}) was set to be equal to one day's worth of messages, and the long-term window (p_{lw}) was set to be 20 days worth of messages. Anomalies were then detected by comparing the behavior on one day to the user's average behavior over the 20 immediately preceding days through the use of the window variation W value explained in Section 4.3. Note that we did not attempt to prevent days with unusual behavior from being included in a user's long-term behavior data.

We used the first third of each data set (approximately one month of data) to determine the value of $\overline{W}$, which is needed to determine the attack detection threshold (see Section 4.3.3). The latter two-thirds of the data were then used to determine true and false positive rates.

5.2 Attack Simulations

Before we analyze the feasibility of our model, we first need to study the eight attack models presented in Section 4.4 in order to understand which one would be the most difficult for our system to detect under realistic attack conditions. In so doing, we assume that if our system can detect the most evasive simulated attacker, it can also detect the others. We also assume, though, that the attacker will attempt to avoid detection from the targeted user as well. Note that this means that the attacker cannot simply imitate normal user behavior, because this would entail making changes to the archive (e.g. reading a message and leaving it marked as read) that would cause a user to be suspicious.

We evaluate the difficulty of detection by comparing the *window variation difference* ΔW caused by the eight attack models and users. In each window, the calculation of ΔW is as:

$$\Delta W = W_a - W_u$$

where W_a is the window variation caused by the simulated attacker and W_u is the window variation of a user's normal behavior. To calculate both of these values, we use our default model parameters (see Table 4) and the profiled behavior calculated by sliding our long-term window across second two-thirds of user data (the first third is used to establish $\overline{W}$). To calculate W_u, we also 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.

Figure 1 shows ΔW for the eight 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; thus, these attack models are the hardest to distinguish from normal user behavior. According to these results, AD and

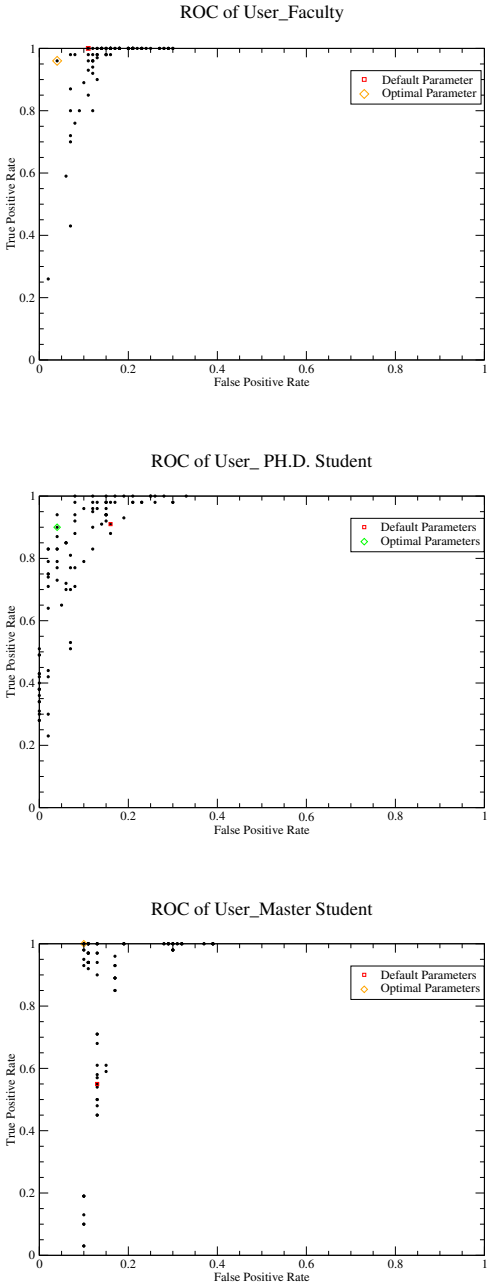

Figure 2. ROC plots for three users: Faculty Member, Ph.D. Student, and Master's Student.

User	p_{sw}	p_{lw}	p_v	p_C	p_W	fp	fa	tp
Fac. (dp)	40	800	0	10	2.0	0.11	9	1.0
Fac. (op)	40	800	0	20	2.5	0.04	25	0.96
Ph.D. (dp)	5	100	0	10	2.0	0.16	6	0.91
Ph.D. (op)	5	75	1	20	2.0	0.04	25	0.90
M.S. (dp)	30	600	0	10	2.0	0.13	8	0.55
M.S. (op)	30	300	0	10	2.5	0.1	10	1.0

Table 5. Model performance using default (dp) and optimal (op) parameter settings. fp: false positive rate; fa: false alarm rate; tp: true positive rate.

AN are closer to user behavior than AUD; these two attack models, however, are of minimal concern, as they can be detected by users themselves (and as such, we wouldn't expect an attacker to behave in these ways). In contrast, it is harder for users on their own to discover attackers behaving according to model AUD. Thus, we chose to use model AUD as the attack model for evaluating our model's ability to detect attacks.

5.3 Feasibility Analysis of the model

To date we have focused on using offline experiments to evaluate feasibility of the model for distinguishing legitimate users from simulated attacker behavior (attack model AUD). In particular, we've focused on testing how different parameter settings affect the trade-off between false positives and true positives. In order for our approach 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.

Figure 2 is an ROC curve for each user based on 96 sets of parameters, with one point per parameter set. These parameter sets are selected combination of the values listed in Table 4. For the faculty and Ph.D. student users, 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%. For the master's student, however, the points are spread much more evenly across the range of false positives. These plots show that our model is a relatively accurate representation of the behavior of faculty and Ph.D. students; the disposition of the master's student emails, 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 would translate into alarms that a user would have to assess, we also analyzed the false alarm rate fa. fa represents the number of days that will pass on average between false alarms. As the size of short-term data is set to be the average number of new messages received each day, the fa is equal to $1/fp$.

Table 5 shows every user's false positive rate, per-average day false alarm rate, and true positive rate for default and optimal parameter settings. Optimal parameter settings are those producing the best performance in terms of low false positives and high true positives. From this table we can see that for all users false alarm rates are relatively low, 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.

6 Discussion

Although we are greatly encouraged by our results, there are also a number of limitations that must also be considered. First, perhaps most importantly, our results are based on a very small user population. We do not believe, however, that our model (or indeed, any simple model of user behavior) could ever apply to all users. Indeed, results on one of our test subjects (the master's student) show that for some users, email disposition does not correlate well with email sender. These results are sufficient, however, to conclude that our modeling strategy has a good chance of working for some high-volume email users. Further, it also shows that user behavior modeling can be performed simply in the email archive domain through a judicious choice of observables and a suitable model representation.

It should also be noted that a number of assumptions were made when analyzing the results for true and false positives, ones that were not necessarily realistic. Users tend to receive a highly variable number of messages a day and not a constant stream of messages, and typical email archive attackers would not read new email messages every day. While such complications would need to be addressed by an online implementation, we do not believe that the addi-

tion of such complications would change our basic results.

Also, in practice we would not want to exclude email client behavior from normal access behavior (if we monitored IMAP commands) given that it can be difficult to distinguish between intended user behavior and program-generated incidental IMAP commands. Because program-level behavior will be relatively repetitive, though, such an addition shouldn't have much of an impact on our false positive rates and in fact it should improve our true positive rates (if attackers choose to use different email clients).

It is true that our detector can only detect an attack once per day; further, we are uncertain what the effect of interleaved attacker and normal behavior (e.g. at different times of day) would have on our model's accuracy. Though we believe that our approach to simulating attack behaviors is relatively systematic, there is still a significant gap between realistic attacker behaviors and the attack model AUD. Can we measure or estimate this gap, and can we improve upon it in a meaningful way? This is an important question for future work, both for this problem and for other approaches to user-level anomaly intrusion detection.

Though user-level anomaly intrusion detection has natural weaknesses, from our current work we think it is possible to build a practical email archives intrusion detection system by taking advantage of inherent features of the domain, selecting good features, and by using simple and efficient modeling methods. Given the vulnerabilities and sensitivity of email archives, we hope this work encourages the development of email archive intrusion detection systems.

Acknowledgements: The authors would like to thank the members of the CCSL for their participation, encouragement, and feedback. This work was supported by the Canadian government through an NSERC Discovery Grant and MITACS.

References

[1] 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.

[2] Apache Software Foundation. SpamAssassin, 2005. http://spamassassin.apache.org.

[3] F. Bergadano, D. Gunetti, and C. Picardi. User Authentication through Keystroke Dynamics. *ACM Trans. Inf. Syst. Secur.*, 5(4):367–397, 2002.

[4] C. Chung, M. Gertz, and K. Levitt. DEMIDS: Misuse Detection System Database System. In *IICIS*, 1999.

[5] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang, and H. Hinton. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In *Proc. 7th USENIX Security Conference*, Jan 1998.

[6] M. Crispin. Request for Comment (RFC) 3501: Internet Message Access Protocol—Version 4rev1, March 2003.

[7] 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.

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

[9] 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.

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

[11] G. Kim and E. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. In *ACM Conference on Computer and Communication Security*, 1994.

[12] J. Klensin. Request for Comment (RFC) 2821: Simple Mail Transfer Protocol, April 2001.

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

[14] Y. Li. Email archive intrusion detection systems. Master's thesis, Carleton University, 2005.

[15] 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.

[16] J. Myers and M. Rose. Request for Comment (RFC) 1939: Post Office Protocol—Version 3, May 1996.

[17] 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.

[18] B. Ramsdell. Request for Comment (RFC) 2633: S/MIME Version 3 Message Specification, June 1999.

[19] M. Roesch. Snort—Lightweight Intrusion Detection for Networks. In *Proceedings of LISA '99: 13th Systems Administration Conference*, Seattle, WA, November 7–12 1999.

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

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

[22] A. Somayaji and S. Forrest. Automated Response Using System-Call Delays. In *Proceedings of the 9th USENIX Security Symposium*, Denver, CO, August 14–17, 2000.

[23] The Spamhaus Project. http://www.spamhaus.org, 2005.

[24] University of Washington. IMAP information center. http://www.washington.edu/imap/, 2005.

[25] K. Wang and S. J. Stolfo. Anomalous Payload-based Network Intrusion Detection. In *RAID 2004 Proceedings*, volume 3224 of *LNCS*, pages 203–222. Springer-Verlag, 2004.

[26] 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.

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