Combining Behavior Models to Secure Email Systems

Salvatore J. Stolfo, Chia-Wei Hu, Wei-Jen Li,
Shlomo Hershkop, Ke Wang, Olivier Nimeskern
Columbia University
{sal, charlie, weijen, shlomo, kewang, on2005}@cs.columbia.edu

5/16/2003

Abstract

We introduce the Email Mining Toolkit (EMT), a system that implements behavior-based methods to improve security of email systems. Behavior models of email flows and email account usage may be used for a variety of detection tasks. Behavior-based models are quite different from "content-based" models in common use today, such as virus scanners. We evaluate the soundness of these techniques for the detection of the onset of viral propagations. The results achieved for the detection of the onset of viral propagations suggest email delivery should be egress rate limited - stored for a while and then forwarded - or a record of recently delivered emails should be kept in order to develop sufficient statistics to verify a propagation is ongoing. EMT can form part of a larger security platform that deals with email security issues in general. We present the variety of EMT models implemented to date and suggest other security tasks that may benefit for its detection capabilities.

1. Introduction

We introduce the Email Mining Toolkit (EMT), a system that implements behavior-based methods to improve security of email systems. Behavior models of email flows and email account usage may be used for a variety of detection tasks. Behavior-based models are quite different from "content-based" models in common use today, such as virus scanners. For pedagogical reasons, we demonstrate the utility of behavior-based models by simulating and detecting viral propagations using real email data. We evaluate the soundness of these techniques for the detection of the onset of viral propagations.

Email is a common method of choice for the propagation of viruses and worms. Typically, a virus will extract email addresses in an infected computer and send a copy of itself to some or all of these addresses. These addresses may be obtained from many sources, such as the address book, socket-layer sniffing, inbox, sent folder, and any of the stored email archives. Virus scanners cannot stop a virus in its tracks unless the signature of the virus is known a priori. Unfortunately, virus writers have demonstrated their continual cleverness by thwarting virus scanners with new viruses that escape early detection. Stopping a polymorphic virus that uses several points of entry can be a daunting task using traditional signature-based virus scanning methods alone.

Like spam email, many viruses that are propagated via email today exhibit various strategies to avoid detection by, adopting strategies such as changing the subject line, text body, and even attached file names and type. This means they will likely escape content-based filtering tools like virus scanners. Updating *virus definitions* in the future will likely be obsolete with a new generation of viruses that can mutate their payload. To complicate this matter further, some viruses look like harmless spam, with no attachments at all. Instead, a user is directed to a site and may download harmful executables without knowing so.

With these trends in mind, we propose behavior-based approaches as a solution to raise the bar of protection and detect and extinguish viral propagations as early as possible. We start by observing the following. First, viral propagations via email must involve an email being sent, with either an attachment or with something equivalent to an HTML page in the text body. In the former case, the user will have to run the executable that launches a virus directly, or invoke a program that uses the seemingly innocent data file that exploits the weakness of the program that makes use of it. In the latter case, the user may simply click on an innocent appearing URL. Second, it is highly unlikely a virus will propagate itself with only one or a few emails. This

This work was partially supported by DARPA contract F30602-00-1-0603

is because usually viruses are designed to infect as many computers as possible in a short period of time. Otherwise, they would be stopped long before they have a chance to inflict damage on many systems. Creating many copies ensures the virus will propagate quickly and widely. Finally, a virus is not intelligent, in the sense that it does not know the relationship between a user and those with whom the user is communicating with. For example, a user would be unlikely to send an email, or copies of an email, to his or her separate *social cliques.* Instead, a virus may use simple hard-coded rules in deciding whom to propagate to, violating the user's social cliques. These observations suggest that viral propagations may be detected by profiling email behavior and using behavior models to detect the onset of a propagation.

Behavior-based detection is not new. Credit card fraud detection is perhaps the best example of a widely deployed security system that depends upon profiling behavior of card users. We posit that a similar approach directed towards "email transactions" will provide comparable broad-cased security without requiring a complete overhaul of email protocols and server implementations.

A behavior-based approach would capture the essence of a user's behavior by analyzing the user's historical email data. This analysis, the subject of this paper, will generate a short summary or profile for each user and use that historical profile to detect outbound email that deviates from the user's normal email pattern. Such summarization or profiling could be done offline during the training or updating phase. The profiles themselves have to be compact and efficient so that they can be used readily in real-time. Equally important, in order to preserve privacy, minimal data from emails should be used.

In this paper, we first explore the use of *cliques* in detecting viral email. A clique is the user's common email recipients as defined by the user's own historical record. Then, on a separate front, we show how certain mathematical models can be used to capture the frequency of the user's email usage. We describe a number of experiments using real email data collected from volunteers but injected with viral emails using simulated attack strategies. (We did not run the viruses but simulated their propagation strategy.) These two behavior-based techniques are then combined to mitigate the weaknesses. Other models may be employed to possibly further improve performance. However, for the present paper we demonstrate the effectives of behavior-based models by combining only two such models. Finally, we detail an experimental system we have implemented – the Email Mining Toolkit (EMT). EMT is an offline data analysis system designed to assist a security analyst compute, visualize and test models of email behavior for use in a real-time email violation detection system, such as the Malicious Email Tracking system reported in [1]. EMT includes a variety of behavior models for email attachments, user accounts and groups of accounts. Each model computed is used to detect anomalous and errant email behavior for an individual account, or for an enclave. A behavior model computed by EMT can be used alone or in combination with other models. EMT is a collection of behavior-based tools that can be easily expanded to accommodate other useful tools to secure a computer or a network of computers.

Two specific behavior-based models are examined in detail in this paper: *user cliques* and the *Hellinger distance* model. The user cliques model profiles a user's communication groups that naturally occur (for example, colleagues, family members, etc). The Hellinger distance profiles the distribution of the frequency of communication, and the variability of that frequency, between a user and his/her correspondents. (Interestingly, the analysis we have performed on the email archives of many volunteer email users reveals that email communication behavior follows a Zipf distribution, the same distribution that models the naturally occurring frequency distribution of words in natural language.) These two models are more or less orthogonal to each other and, as we shall see later, they can be combined together to form a hybrid model that yields good detection performance. The power of these models is demonstrated with respect to the detection of the onset of a virus propagation.

Interestingly, the detection methods studied here do not use message or attachment content. We believe the approach of behavior-based modeling will provide additional evidence that when combined with existing techniques will enhance security for email and other applications.

2. Group Communication Models: Cliques

In order to study email flows between groups of users, we compute a set of *cliques* in an email archive. We seek to identify clusters or groups of related email accounts that participate with each other in common email communications, and then use this information to identify unusual email behavior that violates typical group behavior. For example, intuitively it is unlikely that a user will send a distinct message to his spouse, his boss, his "drinking buddies" and his church elders all appearing together as recipients of the same message. (Of course this is possible, but it is rather unlikely.) A virus attacking his address book at random would surely not know these social relationships and the typical communication pattern of the victim. Hence it would

violate the user's group behavior profile if it propagated itself in violation of the user's *social cliques*.

Clique violations may also indicate email security policy violations internal to a secured enclave. For example, members of the legal department of a company might be expected to exchange many Word attachments containing patent applications. It would be highly unusual, and probably unwise, if members of the marketing department, and HR services would likewise receive these attachments. We can infer the composition of related groups by analyzing normal email flows to compute the naturally occurring cliques, and use the learned cliques to alert when emails violate that clique behavior.

Conceptually, two broad types of cliques can be extracted from user email archives: *user cliques* and *enclave cliques*. In simple terms, user cliques can be inferred by looking at email history of only a single user account, while enclave cliques are social groups that emerge as a result of analyzing traffic flows among a group of user accounts.

2.1. User Cliques

For any user account, an email sent is usually not intended for everyone the user knows. Instead, it would be intended for only a small subset of his/her contacts, who may or may not appear in the address book of the user. We model the collection of recipients in a single email as a set, and summarize these sets and their dynamics. This information is used to detect abnormal emails that violate the user's clique behavior.

The recipient list of a single email can be viewed as a clique associated with the "From:" account. However, using this set directly is problematic for two reasons. First, a single user account would contain a large number of such sets and enumerating them for real-time reporting or detection tasks would be undesirable. Second, some of these sets are duplicates or subsets of one another and it would be difficult to use them directly for any purpose. For these reasons, we define a *user clique* as a set of recipients that cannot be subsumed by another set. Naturally, a single user will have a relatively small number of user cliques. As an example, suppose a user has in his/her sent-folder four emails with the following recipient lists: {A, B, C}, {A, B, C}, {A, B}, and {A, B, D}. The user cliques belong to this user would be {A, B, C} and {A, B, D}. Note that duplicate user cliques are removed, as it does not contribute useful information.

Once these sets are derived off-line, we inspect each email sent from the user's account to see if there is a clique violation – i.e. the recipient list is inconsistent with the user's cliques. The usefulness of this model depends not only on how quickly new groups of recipients form over time but also on how it is combined with other models. Installing a monitoring tool using this model on a new account or an account that is constantly communicating with new groups may cause too many false positives and thus render the model useless. However, this very behavior is indicative of user email usage patterns and thus can be turned into a feature that characterizes user behavior. Although the dynamics of clique formation (and expiration) is implemented in EMT, for the present paper we shall ignore the dynamics of clique formation. Computing the set of "static cliques" is sufficiently informative for the purpose at hand; this model provides useful evidence of a viral propagation launched from a user's account.

2.1.1. Test of Simulated Viruses

We simulate viruses by inserting dummy emails into an email archive following a propagation strategy that has been observed from numerous real viruses seen in the wild. The first 80% of emails sent from each account are used for deriving user cliques associated with that account. The remaining 20% of the emails are used during the testing phase where the dummy emails simulating the propagation are inserted. For this simulation, it is not critical exactly when and how often viral emails are sent out. This is because once the user cliques are derived, determining whether or not a recipient set violates existing user cliques is independent of the timing of the email in question. However, during the simulation/test phase, user cliques are updated on a daily basis and the timing of email is affected slightly. Such effects are still more or less negligible, as having viral emails that are sent late in time is tantamount to having a longer training phase and a shorter test phase.

In terms of modeling attack strategies, we test the effectiveness of the user clique violation model against various sizes of a viral email recipient list. For illustrative purposes, we assume that a virus would fetch email addresses from the address book of a user to propagate itself. In reality, email addresses could be obtained via others means, such as scanning the inbox, sent folder and email archives. Without loss of generality, the simulation has the virus propagating itself to recipients chosen at random. However, the usefulness of user-clique violation detection in practice depends on how a virus obtains the target email addresses. For example, a virus obtaining addresses from an inbox and replying to respective senders and everyone else in the message may not be detected easily, depending upon how compatible they are with existing user cliques. (Herein lies the reason for False Positives. The other models we explore below mitigate these mistakes.)

As we can see from the ROC curve below, the false positive rate is invariant with respect to the size of the

recipient list. This is expected, as this rate is defined as the number of false positives over the number of normal emails, and both of these quantities do not vary with respect to how viral emails are sent under our simulation setting. It is interesting to note that the rate of detection (true positive rate) increases dramatically as the size of recipient lists grow from 1 to 2 to 3 and then approaches 100% gradually as the list size further increases (figure 1). This result is intuitive; we should not expect that there would be many clique violations if a virus sends an email to only one recipient at a time. The fact that this number is not 0, as one might have thought, deserves some mention. This could happen because certain email addresses appear in an address book before any email is sent to them. While a virus may try to thwart our detection effort by sending to one address at a time, it will inevitably have to send many separate emails to achieve the same propagation speed. In doing so, it is likely a different level of threshold would be triggered by another model that is tuned to the user's outbound email frequency. Thus, we combine the user clique detection model with other methods of detection, such as Hellinger Distance described in section 3, to mitigate this error. (Alternatively, as demonstrated below - the *buffer crawling strategy* - we may delay email transmission to gather evidence of clique violations among a sequential set of similar or equivalent emails indicative of a propagation.)

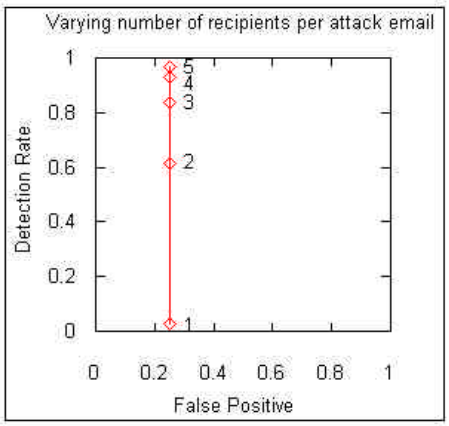

Figure 1

2.2. Enclave Cliques

When viewed at an aggregate level, email traffic flow within an enclave of accounts reveals how tightly connected individuals really are. As a consequence, we can infer clusters of social groups from the density of such links, and use the clique violation strategy to detect a broad propagation within the enclave. An enclave in this context could be a sub-domain, a department, or simply a set of user accounts of interest for inspection. Such analysis could be used for reporting, profiling, and email traffic violation detection.

Before enclave cliques can be used for email violation detection, enclave cliques have to be formed. The enclave clique model is based on graph theory. We use the branch and bound algorithm described in [4]. It finds the largest cliques (groups of users), which are fully connected with a minimum number of emails per connection at least equal to a specified parameter (set at 50 by default). For example, if clique_1 is a clique of three users, A, B and C, this means that the three email accounts must have pair-wise exchanged at least 50 emails. The clique is computed by measuring bi-directional email flow, from A to B and from B to A. With enclave cliques defined this way, any single user can be a member of multiple cliques and thus cliques are not mutually exclusive.

For concreteness, suppose a group of users, {A, B, C, D}, have exchanged a number of messages in the past, as specified in the following table,

To / From	A	B	C	D
A	N/A	20	52	23
B	33	N/A	34	24
C	14	42	N/A	79
D	5	89	37	N/A

Table 1: Number of messages exchanged among users.

Using a threshold of 50, we can immediately see that the following pairs have communicated at least a sufficient number of emails to be qualified in the same group of dyad cliques: {AB, AC, BC, BD, CD}. The triad cliques are {ABC} and {BCD}. Note that enclave cliques are not necessarily mutually exclusive.

Figure 2 shows two cliques sharing two common accounts.

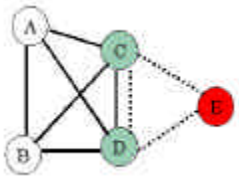

Figure 2: Two enclave cliques sharing 2 common user accounts.

2.2.2 Clique reporting

For summarization and reporting, one could simply show the enclave cliques discovered by the algorithm. However, it would be even more illuminating if the nature of the communication shared among members of such cliques is further revealed. For every clique, EMT computes the most frequently occurring words appearing in the subject line of the emails (although body text may also be used). Figure 3 displays an example enclave clique report generated by EMT.

2.2.3. Clique violation detection

Using enclave cliques to detect anomalous email traffic flow is similar to the case using user cliques, except that now we are examining all email traffic among a selected group of users, instead of focusing on one user. For every email circulated within the enclave, we check the set of users involved, including the sender, and see if this set is a subset of a known enclave clique. If it is not, an alarm is generated for further examination; this may be evidence of a viral propagation, or some security policy violation.

The usefulness of this detection algorithm depends on how well established traffic patterns are revealed by the email archive available for analysis. In an enclave that is newly formed or constantly changing its membership or communication patterns, there would be many false alarms. However, the fact that an enclave is laden with many false alarms is itself informative. It implies the fast changing nature of the communication pattern in the enclave.

In practice, an enclave defined arbitrarily by a random set of users would not be very useful, as there may not be much traffic to examine in the first place. On the other hand, a well defined enclave may not reveal as much information as we desire, using the enclaves defined thus far, as there may be a lot of external traffic in and out of the enclave. Nonetheless, the information captured by user cliques still provides evidence of a propagation. Enclave cliques remain a useful tool for summarization and email violation detection.

Figure 3: Common subject words

3. Non-Stationary User Profiles

Most email accounts follow certain trends, which can be modeled by some underlying distribution. As an example of what this means, many people will typically email a few addresses very frequently, while emailing many others infrequently. Day to day interaction with a limited number of peers usually results in some predefined groups of emails being sent. Other contacts communicated to on less than a daily basis have a more infrequent email exchange behavior. These patterns can be learned through the analysis of a user's email archive over a bulk set of sequential emails. For some users, 500 emails may occur over months, for others over days. The duration of these email transmissions is not material for the profile we now consider.

Every user of an email system develops a unique pattern of email emission to a specific list of recipients, each having their own frequency of occurrence (with respect to the number of emails). Modeling every user's idiosyncrasies enables the system to detect malicious or anomalous activity in the account. This is similar to what happens in credit card fraud detection, where current behavior violates some past behavior patterns.

3.1 Profile of a user

We analyze the account's activity in terms of recipient frequency. Figure 4 displays the frequency at which the

user sends emails to all the recipients communicated to in the past. Each point on the x-axis represents one recipient and the corresponding height of the bar measures the frequency of emails sent to this recipient, as a percentage. (The display is an actual distribution from a volunteer email account. All others have been found to follow the same type of distribution.)

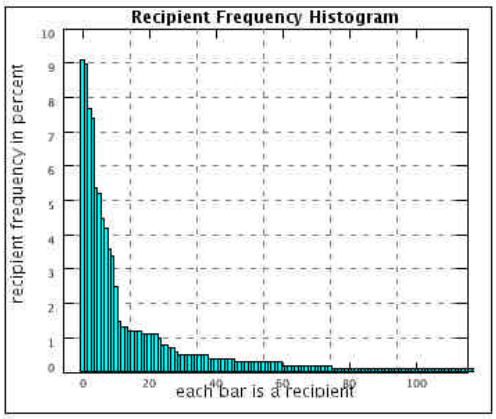

Figure 4: Recipient Frequency Histogram

This bar chart is sorted in decreasing order, and usually appears as a nice convex curve with a strong skewedness; a long low tail on the right side, and a very thin spike at the start on the left side. This frequency bar chart can be modeled with either a Zipf function, or a DGX function (Discrete Gaussian Exponential function), which is a generalized version of the Zipf distribution. This distribution characterizes some specific human behavioral patterns, such as word frequencies in written texts, or URL frequencies in Internet browsing [2]. In brief, its main trait is that few objects receive a large part of the flow, while many objects receive a very small part of the flow.

The rank-frequency version of Zipf's law states that $f(r)$? $1/r$, where $f(r)$ is the occurrence frequency versus the rank r, in logarithmic-logarithmic scales. The generalized Zipf distribution is defined as $f(r)$? $(1/r)^{?}$, where the log-log plot can be linear with any slope. Our tests indicate that the log-log plots are concave, and thus require the usage of the DGX distribution for a better fit [2].

We also analyze the number of distinct recipients and attachments. Figure 5 contains several curves that visualize the variability of the user's emission of emails. They are calculated by the number of distinct recipients and number of messages with attachments. The first type of curve uses a rolling window of 50 (or 20) emails to calculate the number of distinct recipients. What it means is, the higher its value (and thus the closer to 50), the wider the range of recipients the selected user sends emails to, over time. On the other hand, if the metric is low, it means that the user predominantly sends messages to a small group of people. We also use the moving average (using 100 records) to indicate the trend.

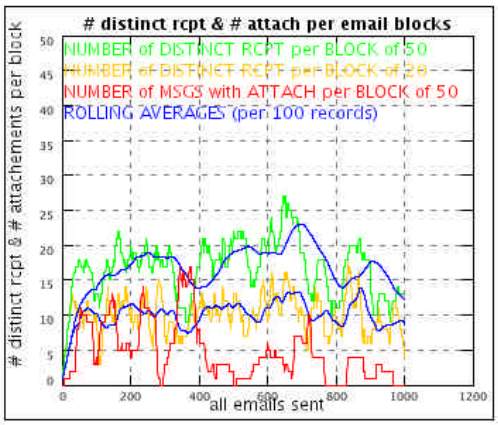

Figure 5: Recipient and attachment

We also plot a curve using 20 as the window size instead of 50. This metric has a faster reaction to anomalous behavior, while the previous one using blocks of 50 shows the longer-term behavior. The short-term profile can be used as the first level of alert, the longer-term one acting to confirm it.

Another type of curve is the number of messages with attachment(s), per block of 50 emails. It shows the average ratio of emails with attachments versus emails without attachments, and any sudden spike of emails sent with attachments will be detected on the plot.

The profile displays a fingerprint of a specific user's email frequency behavior. The most common malicious intrusion can be detected very fast by the metrics. For instance, a Melissa type virus would be detected since the curves will jump up to 50, 20 and 50 respectively.

3.2 Chi Square Test of User Histograms

We test the hypothesis that the recipient frequencies are identical over two different time frames by Chi Square. Obviously, recipient frequencies are not constant over a long time horizon, as users will add new recipients and drop old ones. It can be informative for behavioral modeling though, to analyze the variability of frequencies over two near time frames.

We compare two time periods of activity for the same user. The idea is to treat the first period as the true distribution corresponding to the user under normal behavior, while the second time period is used to evaluate whether or not the user's frequencies have changed, providing evidence that perhaps a malicious activity is taking place. Generally, we operate under the usual 1/5 - 4/5 ratio between testing and training sets. For example, we use 1000 messages as a testing set, 200

past emails are selected as the testing range, while previous 800 are the training range.

Assuming that the observed frequencies corresponding to the first, longer time frame window are the true underlying frequencies, the Chi Square statistic enables us to evaluate how likely the observed frequencies from the second time frame are to be coming from that same distribution [18]. The Chi Square formula is $Q = \sum_{i=1}^{k} (X(i) - np(i))/np(i)$, where $X(i)$ is the number of observations for recipient (i) in the testing range, $p(i)$ is the true frequency calculated from the training range, n is the number of observations in the testing range, and k is the number of recipients. There are (k-1) degrees of freedom.

The p-value represents the probability that the frequencies in both time frames come from the same multinomial distribution. In order to get an idea of the variability of the frequencies under real conditions, we used a sample of 37,556 emails from 8 users. We ran two batches of calculations. First, we used a training period size of 400 emails and a testing period size of 100 emails; for each user, we started at the first record, calculated the p-value, then translated the two windows by steps of 10 records until the end of the log was reached, each time calculating the p-value. Secondly, we reproduced the same experiment, but with a training period size of 800 emails, and a testing period size of 200 emails. We thus collected a total of 7,947 p-values, and their histogram is shown in figure 6.

Under the hypothesis that the frequencies are constant, the histogram is expected to be a flat line. On the contrary, this histogram is characterized by a very large concentration of p-values between 0 and 5%, and a large (but less large) concentration between 95 and 100%, while p-values in the range of 5 to 95% are under-represented. Our intuitive explanation of this histogram (also based on our domain knowledge) is the following: Most of the time, frequencies change significantly (in a statistical sense) between two consecutive time frames; this is why 60% of the p-values are below 5% (as a low p-value indicates a very high chance that the frequencies have changed between two time frames). Email users tend to modify their recipient frequencies quite often (at least the 8 volunteers). On the other side, there are non-negligible times when those frequencies stay very stable (as 13% of the p-values are above 95%, indicating strong stability). As the frequencies have been found to be so variable under normal circumstances, the Chi Square itself could not be used to detect an abnormal email behavior. Instead we explore a related metric, which will be more useful for that purpose.

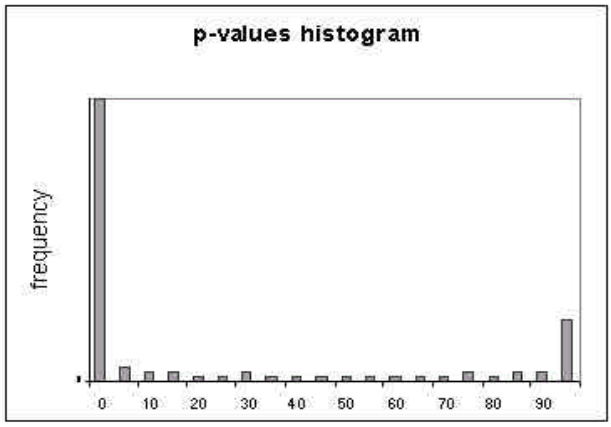

Figure 6: P-value plot

3.3 Hellinger Distance

Our first tests using the Chi-square statistic revealed that the frequencies cannot be assumed to be constant between two consecutive time frames for a given user. What is specific to every user though, is how variable frequencies are over time. We try to assess this by calculating a measure between two frequency tables.

We use the Hellinger distance for this purpose. It is defined as $HD(f_1[], f_2[]) = \sum_{i=0}^{n-1} (\sqrt{f_1[i]} - \sqrt{f_2[i]})^2$, where $f_1[]$ is the array of frequencies for the training set, $f_2[]$ for the testing set, $\boldsymbol{n}$ the total number of distinct recipients during both periods. Figure 7 displays an example for a user from our group of volunteers.

The Hellinger distance plot shows the distance between training and testing sets plotted over the entire email history of the user. For example, if a user has 2500 outbound emails, the plots starts at the 500th record, and measures the distance between the frequencies corresponding to the first 400 records, versus the emails corresponding to the next 100 records; these two windows, of 400 and 100 records, respectively, are then rolled forward over the entire email history of the user, by steps of one record. At each step, a Hellinger distance is calculated between the given training window of 400 records, and the corresponding testing window of 100 records.

What this plot tells us is that when a burst occurs, the recipient frequencies have been changing significantly. This can be either a normal event, as we know from the previous section, or a possible viral propagation.

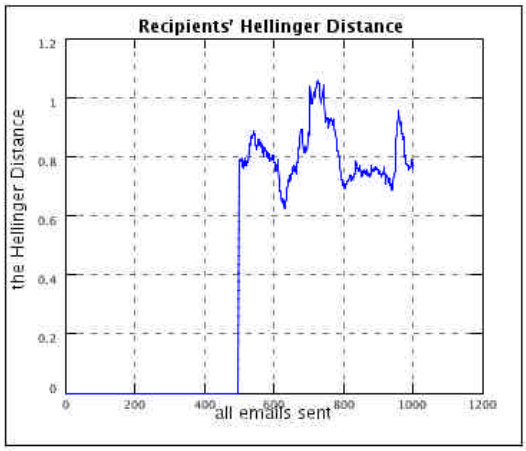

Figure 7: The Hellinger distance of a typical User

3.4 Tests using simulated viruses

As real email data with real embedded viral emails are very difficult to obtain [19], we generated outbound "dummy" viruses, and insert "virus email" records into a real email log file as described above. A set of parameters introduces randomness in the process, in order to mimic real conditions: the time at which the virus starts, the number of corrupted emails sent by the virus and its propagation rate.

For testing purposes, all the recipients of such "dummy" corrupted emails are picked randomly from the address list of a selected user. In reality, where addresses are obtained and how they are combined can be a crucial issue for a virus to successfully propagate itself without being detected. The chosen recipients can be set to be all distinct, as most viruses do. But not all viruses would send an email only once to each target recipient account. In our simulation, each "dummy" email contains one attachment, but no information about the attachment is provided or used. (Recall, our focus here is to demonstrate the value of behavior models, as an adjunct to content-based analyses.) For our purposes, we do not need to know the content of the message, its size, and the size and content of the attachments. So, these techniques may be general enough that they encompass polymorphic viruses as well (where content analysis or scanners may fail). Similarly, even though viruses can also propagate through HTML content, our techniques will handle these techniques as well.

Our experiments use a combination of three plots, "Hellinger distance", "number of distinct recipients", and the "number of attachments", as detailed in the above sections. Our intuition is that when a virus infiltrates itself, it causes each plot to rapidly grow. We use two types of thresholds to determine when a burst occurs, a threshold proportional to the standard deviation of the plots, and a heuristic formula evaluating when a change of trend occurs. We cannot say which threshold is better. The first threshold always misses the first half of the viral propagation. The second one always catches separate batches of viruses. Moreover, each of them has different false positive rates. We take the union (OR) of the models to evaluate a series of emails and to generate results.

The dataset used for this independent test is an archive of 16 users, totaling 20,301 emails. The parameters that were randomly generated at each simulation were the time of the intrusion and the list of recipients (taken from the address list of each selected user). The parameters that were controlled were the propagation rate, the number of corrupted emails sent, and the window size (Hellinger distance). In total, about 500,000 simulations were performed.

As expected, a slower propagation rate (longer inter-departure time) makes detection harder, as in such a case, each corrupted email becomes less "noticeable" among the entire email flow. As can be seen in Figure 8, the performance gets worse when the inter-departure time increases.

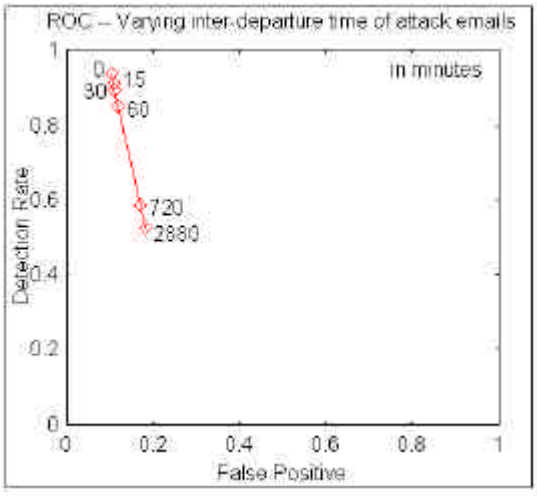

Figure 8: Varying inter-departure time

The Hellinger window size is the most important parameter. In Figure 9, the performance is best when the window size is the same as the number of dummy emails. The reason is that, for example, when the window size is 50 and there are 20 dummy emails (# of dummy emails is less than window size), the dummy emails do not occupy a very significant portion of the 50 emails. The model may not determine that they are "suspicious". On the other hand, if there are 100 dummy emails and the first 50 are not detected, these 50 dummy emails will be treated as normal emails in the next round of Hellinger training. As a consequence, the system will likely model the second 50 as normal and not be able to detect any abnormality.

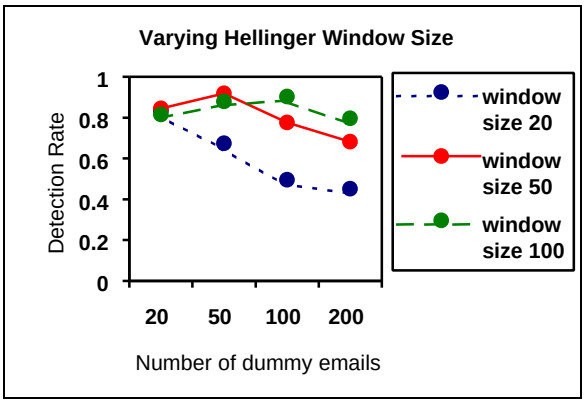

Figure 9: Varying Hellinger window size

In summary, we achieved very reasonable results with the Hellinger distance model. However, there are still three problems. First, we assumed that we have enough normal emails before and after dummy ones, and we can analyze all the emails (both dummy and normal) at the same time, which is not practical. (We cannot block a user's email for a long time, for instance, a month. However, we may store a record of the emails and detect the propagation after the fact, but perhaps still in sufficient time to forewarn the recipients that they likely have a viral email in their inbox.) Second, it's difficult to optimize the Hellinger window size, as it depends on the viral strategy used. In practice, we can overcome this by blocking all outgoing emails once we detect a virus. The question is then how can we detect the first virus. Third, the false positive rate is about 15%, which cannot be reduced in this model. Thus, to achieve a better detector, this method has to be used in combination with other models. The first two issues will be addressed in the next section.

4. Combining User Clique and Hellinger Distance

Hellinger distance is the result of inspecting the aggregate behavior of a sequence of emails. As such, it would not react immediately when a viral email appears. Similarly, it would keep setting alarms for a short while after a batch of viral emails has already been sent out. On the other hand, user cliques could detect a suspicious viral email upon its first appearance. It is worth mentioning that every time an email with a new address appears, the user clique model will treat it as a violation. In short, Hellinger analyzes the trend of users' behavior by buffering records, while the user clique method is good for single email detection. Ideally, we want to take only the best features from each method and combine them to achieve better overall performance.

4.1 Buffer Crawling

The most straightforward method to combine user cliques and Hellinger is to take the intersection (AND) of their results. A close examination shows that they have different distributions of false positives. For example, the user cliques model may generate false positives on email number 1, 3 and 5, while Hellinger may generate false positives on email number 2, 4 and 6. If we take the intersection, we can eliminate most false positives. However, a lower false positive rate is achieved at the expense of a lower detection rate (hit rate) -- down to 40%, much worse than before.

We propose an alternative strategy we call the buffer-crawling method. Emails are assumed to be buffered before they are actually sent out (or, as we mentioned, a record of the sent emails are kept for analysis). Such buffering could be hidden and unbeknownst to the user. Email is fundamentally a store and forward technology. However, an egress "*store for a while, then forward*" strategy for email delivery has a practical advantage. As far as the user is concerned, the email is sent from client to server and is delivered by the underlying communication system at some arbitrary future time. Thus, the strategy of buffering and holding emails for some period of time allows sufficient statistics to be computed by the models and also benefits mitigation strategies to quarantine viral emails, limiting exposure to the enclave. Alternatively, a record of the recently delivered emails may also benefit early detection and mitigation strategies. When the system sees an alarm triggered by both the Hellinger distance model and the user cliques model, it will examine all adjacent emails more closely, those preceding it and those newly sent by the client. Namely, it will trace (crawl) all buffered emails forward and backward (or their record of delivery), starting from the common trigger. The trace attempts to find all sequential emails that are deemed suspicious by the user cliques model and will end once a harmless email, as viewed by user cliques, is encountered. The system then marks all those emails found along the trace as suspicious.

The idea behind the backward trace is that user cliques are quick at detecting a suspicious email while Hellinger distance takes a while to trigger. The forward trace is the result of empirical observation that when a common trigger is encountered, viral emails in the neighborhood of the common trigger are detected more easily by user cliques.

Figure 10 is a simple example of buffer-crawling algorithm. Each email in the sequence is denoted by "x" or "o", depending on whether or not there is an alert associated with it. The alerts generated by user clique and the Hellinger model are in the first and second row, respectively. In the third row, the system starts checking (backward) at the first common "x" in both user clique and Hellinger alert.

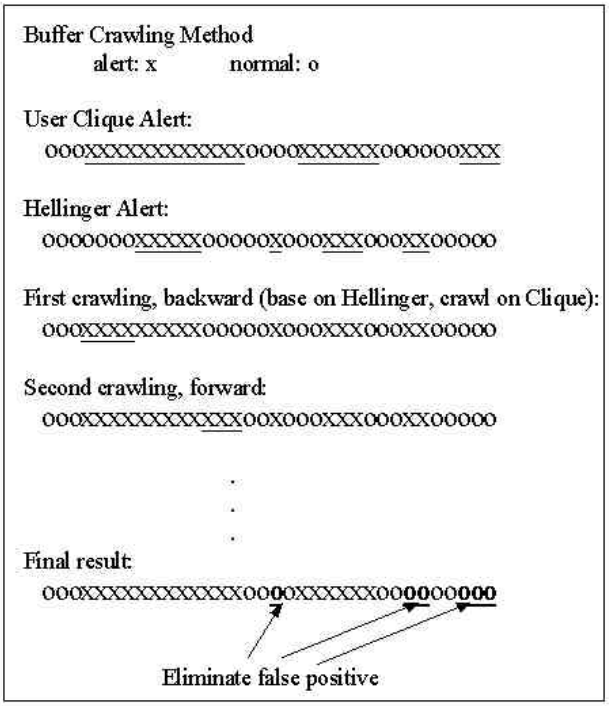

Figure 10: Buffer crawling method.

4.2 Tests of simulated viruses

The dataset for this independent test includes an entire year of email from 15 users. We treat data of the first ten months as each users' normal behavior (training data) and inject a batch of "dummy emails" into the last two months. During the test phase, we train and test the email data on a daily basis. This means that on the first day of the eleventh month, we put all the data (on that day) into the buffer. Then we use the training data to test whether they are suspicious emails depicting abnormal behaviors. We then move on day by day for testing. It is clearly not desirable to buffer and hold emails for too long. When that happens, there is bound to be a number of viral emails leaking out undetected by the Hellinger model. As we noted, however, learning that a propagation did occur is valuable information that may help mitigate broad effects within an enclave.

The parameters that are controlled are the propagation rate and the number of recipients in a single dummy email. The first parameter is one of the most important issues in the Hellinger simulation (section 3.4). The second parameter is more pertinent to user cliques. Having more recipients in a single email makes it easier for user clique to detect a violation.

Another important issue is the Hellinger window size (Hellinger Distance, see section 3.3, 3.4). Since it is impossible to a priori choose a perfect Hellinger window size, we change it by evaluating the size of data (records) each day. The window size is meaningless if it is too small or too big. If it's too small, each email has too much of an influence and each email may look like a virus. If it's too big, the training data would not be enough and a small number of viruses could easily go undetected. Generally, the window size is the average number of emails sent on an average day. If the size is less than 20 or more than 100, it is set at 20 and 100, respectively.

4.3 Results and Discussion

Varying the number of recipients in a single virus email yields a very interesting result. In Figure 11, we have what we expect from intuition. The detection rate increases with the size of the recipient list in a dummy email. It means that if a virus picks up many email addresses (for example, 9) and sends them in a single email or at the same time. We have a high detection rate (about 95%). The false positive rate also decreases when using multiple recipients.

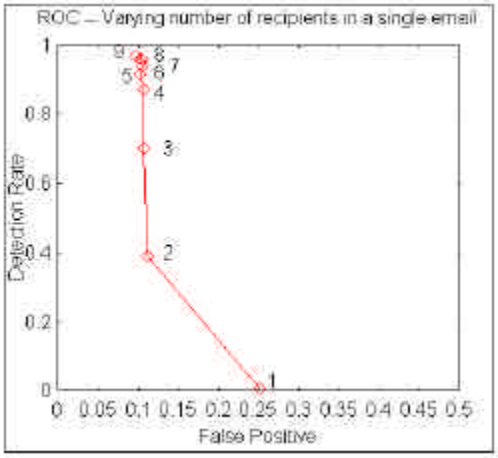

Figure 11

The first test is quite encouraging. However, this is because we set a low propagation rate in our simulated "dummy" emails. The inter-departure time used is uniformly distributed between 0 and 10 minutes in this test. The next test varies this propagation rate.

In this independent test, the number of recipients in a single email is set to 4. Similar to the propagation rate test in section 3.4, the detection rate gets worse when the inter-departure time increases (Figure 12). If this happens in the real world, once we find the first virus (with long inter-departure time), we would likely have enough time to mitigate its effects, since it propagates slowly. Thus, the issue here again is about catching the first virus. Fortunately, on average, our system in both tests can always catch the first or second dummy email.

It is important to note that the two behavior models used here, Hellinger distance and user cliques, do detect viral propagations. This is presented to demonstrate the power of behavior models. Other models are available and are described next. We believe other combinations of models, including content-based models, will

substantially raise the bar of protection against future viruses.

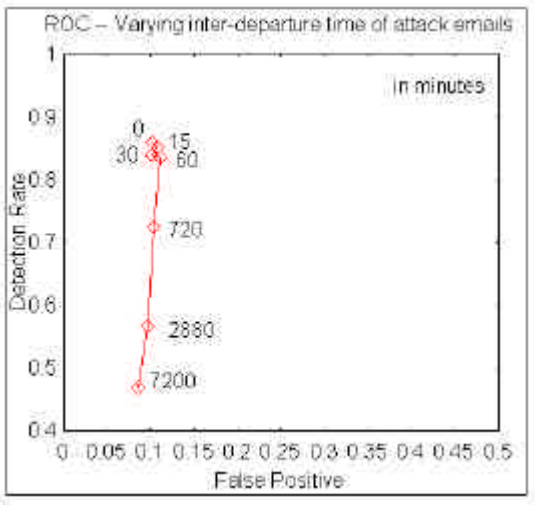

Figure 12

5. EMT – Email Mining Toolkit

In order to centralize and coordinate various behavior-based methods for email security, we have implemented EMT – Email Mining Toolkit. This toolkit is useful for report generation and summarization of email archives, as well as for detecting email security violations when incorporated with a real-time violation detection system, such as the MET (Malicious Email Tracking) system. In addition to all those techniques mentioned in the previous sections, EMT contains a myriad of features that may be combined for various detection tasks.

5.1 Attachment Statistics and Alerts

EMT runs an analysis on each attachment in the database of emails to calculate a number of metrics. These may be used to track important documents, for example. These include birth rate, lifespan, incident rate, prevalence, threat, spread, and death rate. They are explained fully in [1].

Rules specified by a security analyst using the alert logic section of MET are evaluated over the attachment metrics to issue alerts to the analyst. This analysis may be executed against archived email logs using EMT, or at runtime using MET. The initial version of MET provides the means of specifying thresholds in rule form as a collection of Boolean expressions applied to each of the calculated statistics. As an example, a basic rule might check for each attachment seen:

*If its **birth rate** is greater than a specified threshold, T, AND **sent from** at least X number of users.*

5.2 Account Statistics and Alerts

EMT provides support for email abuse detection, such as masquerader detection. Thus, EMT generates alerts based upon deviation from other baseline user and group models. EMT computes and displays three tables of statistical information for any selected email account. The first is a set of stationary email account models, i.e. statistical data represented as a histogram of the average number of messages sent over all days of the week, divided into three periods: day, evening, and night. EMT also gathers information on the average size of messages for these time periods, and the average number of recipients and attachments for these periods. These statistics can generate alerts when values are above a set threshold as specified by the rule-based alert logic section.

We next describe the variety of models available in EMT that may be used to generate alerts of errant behavior.

5.3 Stationary User Profiles

Histograms are used to model the behavior of a user's email accounts. Histograms are compared to find similar behavior or abnormal behavior within the same account (between a long-term profile histogram, and a recent, short-term histogram), and between different accounts.

A histogram depicts the distribution of items in a given sample. EMT employs a histogram of 24 bins, for the 24 hours in a day. (Obviously, one may define a different set of stationary periods as the detect task may demand.) Email statistics are allocated to different bins according to their outbound time. The value of each bin can represent the daily average number of emails sent out in that hour, or daily average total size of attachments sent out in that hour, or other features defined over an of email account computed for some specified period of time.

Two histogram comparison functions are implemented in the current version of EMT, each providing a user selectable distance function as described below. The first comparison function is used to identify groups of email accounts that have similar usage behavior. The other function is used to compare behavior of an account's recent behavior to the long-term profile of that account.

5.3.1 Histogram Distance Functions

A *distance function* is used to measure histogram dissimilarity. For every pair of histograms, h_1, h_2, there is a corresponding distance $D(h_1, h_2)$, called the distance between h_1 and h_2. The distance function is non-negative, symmetric and 0 for identical histograms. Dissimilarity is proportional to distance. We adapted some of the more commonly known distance functions: simplified histogram intersection (L1-form), Euclidean distance (L2-form), quadratic distance [9] and histogram

Mahalanobis distance [12]. These standard measures were modified to be more suitable for email usage behavior analysis. For concreteness,

L1-form: $D_1(h_1,h_2) ? \sum_{i?0}^{n?1} | h_1[i] ? h_2[i] |$

L2-form: $D_2(h_1,h_2) ? \sum_{i?0}^{n?1} (h_1[i] ? h_2[i])^2$

Quadratic: $D_3(h_1,h_2) ? (h_1 ? h_2)^T A(h_1 ? h_2)$

where n is the number of bins in the histogram. In the quadratic function, A is a matrix where a_{ij} denotes the similarity between bins i and j. In EMT we set $a_{ij} ? | i ? j | ? 1$, which assumes that the behavior in neighboring hours is more similar. The Mahalanobis distance is a special case of the quadratic distance, where A is given by the inverse of the covariance matrix obtained from a set of training histograms.

5.3.2 Abnormal User Account Behavior

The histogram distance functions are applied to one target email account. A long-term profile period is first selected by an analyst as the "normal" behavior training period. The histogram computed for this period is then compared to another histogram computed for a more recent period of email behavior. If the histograms are very different (i.e., they have a high distance), an alert is generated indicating possible account misuse. We use the weighted Mahalanobis distance function for this detection task.

The long term profile period is used as the training set, for example, a single month. We assume the bins in the histogram are random variables that are statistically independent. Then we get the following formula:

$$D_4(h_1,h) ? (h_1 ? h)^T A(h_1 ? h)$$

$$A ? B^{?1}, \quad b_{ii} ? Cov(h[i],h[i]) ? Var(h[i]) ? ? _i^2$$

$$B ? \begin{pmatrix} ?? _0^2 & 0 & \cdots & 0 ? \\ ? 0 & ? _1^2 & \cdots & 0 ? \\ ? & \cdots & \cdots & ? \\ ? 0 & \cdots & 0 & ? _{n?1}^2 ? \end{pmatrix}$$

Then we get: $D_4(h_1,h) ? \sum_{i?0}^{n?1} ((h_1[i] ? h[i])^2 / ? _i^2)$

Vector h represents the histogram of the (eg., one month) profile period, while h_1 represents the recent profile period (eg., one week). $? _i$ describes the dispersion of usage behavior around the arithmetic mean. We then modify the Mahalanobis distance function to the weighted version. First we reduce the distance function from the second degree function to the first degree function; then we assign a weight to each bin so that the larger bins will contribute more to the final distance computation:

$$D_4(h_1,h) ? \sum_{i?0}^{n?1} w_i (h_1[i] ? h[i]) / ? _i$$

Weight $w_i ? h_1[i] / \sum_{j?0}^{n?1} h_1[j]$

When the distance between the histogram of the selected recent period and that of the longer term profile is larger than a threshold, an alert will be generated to warn the analyst that the behavior "might be abnormal" or is deemed "abnormal". The alert is also put into the alert log of EMT.

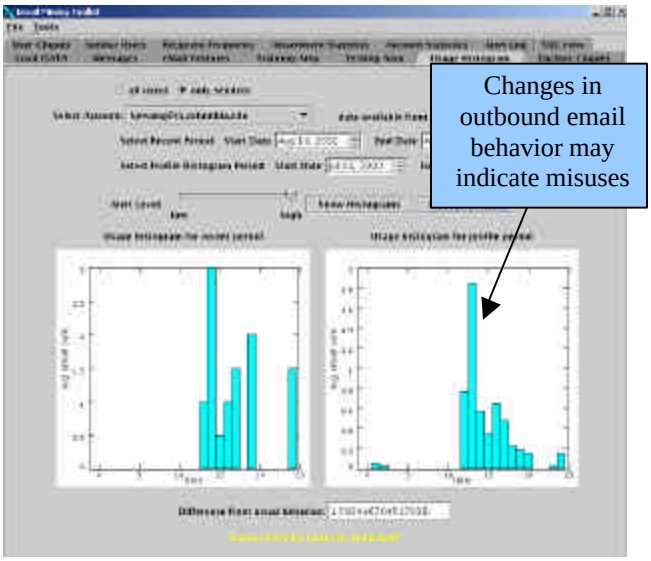

Figure 13: Abnormal Behavior Detected

The histograms described here are stationary models; they represent statistics at discrete time frames. Other non-stationary account profiles are provided by EMT, where behavior is modeled over sequences of emails irrespective of time. These models are described next.

5.3.3 Similar Users

User accounts that may behave similarly may be identified by computing the pair-wise distances of their histograms (eg., a set of SPAM accounts may be inferred given a known or suspect SPAM account as a model). Intuitively, most users will have a pattern of use over time, which spamming accounts will likely not follow. (SPAMbots don't sleep or eat and hence may operate at times that are highly unusual.)

The histogram distance functions were modified for this detection task. First, we balance and weigh the information in the histogram representing hourly behavior with the information provided by the histogram representing behavior over different aggregate periods of a day. This is done since measures of hourly behavior may be too low a level of resolution to find proper groupings of similar accounts. For example, an account that sends most of its email between 9am and 10am

should be considered similar to another that sends emails between 10am and 11am, but perhaps not to an account that emails at 5pm. Given two histograms representing a heavy 9am user, and another for a heavy 10am user, a straightforward application of any of the histogram distance functions will produce erroneous results.

Thus, we divide a day into four periods: morning (7am-1pm), afternoon (1pm-7pm), night (7pm-1am), and late night (1am-7am). The final distance computed is the average of the distance of the 24-hour histogram and that of the 4-bin histogram, which is obtained by regrouping the bins in the 24-hour histogram.

Second, because some of the distance functions require normalizing the histograms before computing the distance function, we also take into account the volume of emails. Even with the exact distribution after normalization, a bin representing 20 emails per day should be considered quite different from an account exhibiting the emission of 200 emails per day.

In addition to find similar users to one specific user, EMT computes distances pair-wise over all user account profiles, and clusters sets of accounts according to the similarity of their behavior profile. To reduce the complexity of this analysis, we use an approximation by randomly choosing some user account profile as a "centroid" base model, and then compare all others to this account. Those account profiles that are deemed within a small neighborhood from each other (using their distance to the centroid as the metric) are treated as one clustered group. The cluster so produced and its centroid are then stored and removed, and the process is repeated until all profiles have been assigned to a particular cluster.

5.4 Supervised Machine Learning Models

In addition to the attachment and account frequency models, EMT includes an integrated supervised learning feature akin to that implemented in the MEF system previously reported in [12].

5.4.1 Modeling Malicious Attachments

MEF is designed to extract content features of a set of known malicious attachments, as well as benign attachments. The features are then used to compose a set of training data for a supervised learning program that computes a classifier.

MEF was designed as a component of MET. Each attachment flowing into an email account would first be tested by a previously learned classifier, and if the likelihood of "malicious" were deemed high enough, the attachment would be so labeled, and the rest of the MET machinery would be called into action to communicate the newly discovered malicious attachment, sending reports from MET clients to MET servers.

The core elements of MEF are also being integrated into EMT. However, here the features extracted from the training data include content-based features of email bodies (not just attachment features).

The Naïve Bayes learning program is used to compute classifiers over labeled email messages that are deemed interesting or malicious by a security analyst. The GUI allows the user to mark emails indicating those that are interesting and those that are not, and then may learn a classifier that is subsequently used to mark the remaining set of unlabeled emails in the database automatically.

A Naïve Bayes [5] classifier computes the likelihood that an email is interesting given a set of features extracted from the set of training emails that are specified by the analyst. In the current version of EMT, the set of features extracted from emails includes a set of static features such as domain name, time, sender email name, number of attachments, the MIME-type of the attachment, the likelihood the attachment is malicious, the size of the body, etc. Hot-listed "dirty words" and n-gram models and their frequency of occurrence are among the email message content-based linguistic features supported.

6. Concluding Remarks

We have introduced several behavior-based methods in this paper and described how these notions can be used in detecting viral email propagations. These methods deviate from traditional approaches that handle system vulnerabilities only when the signature of an attacking virus is known. We have defined user cliques, enclave cliques, and various non-stationary user profiles. Two specific techniques were presented and tested: user cliques and Hellinger distance. They can also be combined to achieve high detection rates.

The Hellinger model can be used to confirm which alerts in user cliques should be kept, and which alerts should be eliminated. Doing so reduces the false positives and does not reduce the detection rate. When a user sends out an email with an unusual address combination, user cliques can point them out and Hellinger will either strengthen or weaken its finding and label the email as malicious or harmless.

We further find that delaying outbound email delivery for some period of time provides the means to gather additional statistics and detects viral propagations more accurately. Indeed, if detection is performed at the onset of the viral propagation, it may be entirely stopped.

The techniques can be used as part of a larger system that handles email security or system security in general. We have reported an implementation of a behavior-based email security system, EMT, which encompasses

many modeling techniques that can be combined in interesting ways. EMT includes a variety of behavior models for email attachments, user accounts and groups of accounts. Each model computed is used to detect anomalous and errant email behaviors. EMT has been deployed to several organizations that are actively testing its features for a variety of security purposes.

Simulation results based on the data obtained from a number of volunteers (who we thank) indicates these techniques are quite promising. In future work, we plan to incorporate these tools on a larger scale and explore further how these tools can be optimally combined.

7. References

1. M. Bhattacharyya, S. Hershkop, E. Eskin, and S. J. Stolfo. ``*MET: An Experimental System for Malicious Email Tracking.''* In Proceedings of the 2002 New Security Paradigms Workshop (NSPW-2002). Virginia Beach, VA, September, 2002.

2. Zhiqiang Bi, Christos Faloustos, Flip Korn. *"The DGX Distribution for Mining Massive, Skewed Data"*, 2001

3. C. Bron, J. Kerbosch, *Finding all cliques of an undirected graph*, Comm. ACM 16(9) (1973) 575--577.

4. E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. J. Stolfo. ``*A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data''*, To Appear in Data Mining for Security Applications. Kluwer 2002.

5. George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Pages 338-345, 1995

6. 6.Wenke Lee, Sal Stolfo, and Kui Mok. ``Mining Audit Data to Build Intrusion Detection Models" *In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD '98)*, New York, NY, August 1998

7. Wenke Lee, Sal Stolfo, and Phil Chan. ``*Learning Patterns from Unix Process Execution Traces for Intrusion Detection" AAAI* Workshop: AI Approaches to Fraud Detection and Risk Management, *July 1997*

8. MySQL, www.mysql.org, 2002.

9. W.Niblack et al. *"The QBIC project: querying images by content using color, texture, and shape"*. In *Proceedings of the SPIE*, February 1993

10. Procmail,www.procmail.org,2002.

11. Sendmail,www.sendmail.org,2002.

12. Matthew G. Schultz, Eleazar Eskin, and Salvatore J. Stolfo. ``*Malicious Email Filter - A UNIX Mail Filter that Detects Malicious Windows Executables*." Proceedings of USENIX Annual Technical Conference - FREENIX Track. Boston, MA: June 2001.

13. J.R.Smith. *Integrated Spatial and Feature Image Systems: Retrieval, Compression and Analysis*. PhD thesis, Columbia University, 1997.

14. M.M. Williamson, *"Throttling viruses: Restricting propagation to defeat malicious mobile code"*, Prof. ACSAC Security Conference, Las Vegas, NV, 2002.

15. M.E. Newman, S. Forrest and J. Balthrup, "*Email networks and the spread of computer viruses"*, The American Physical Society, 2002.

16. Damashek, M.. *Gauging similarity with n-grams: language independent categorization of text*. Science, 267(5199):843--848, 1995.

17. Mitchell, Tom M. *Machine Learning*. McGraw-Hill, 1997, pg. 180-183.

18. R.V. Hogg, A.T. Craig, "Introduction to Mathematical Statistics", Prentice Hall, 1994, pg 293-301

19. M. Schonlau, W. DuMouchel, WH Ju, A.F.Karr, M theus and Y. Vardi, "Computer Intrusion Detecting Masquerades", Statistical Science, Vol. 16, 2001