Feedback Email Worm Defense System for Enterprise Networks

Cliff C. Zou*, Weibo Gong*, Don Towsley§
*Dept. Electrical & Computer Engineering
§Dept. Computer Science
University of Massachusetts, Amherst
Technical Report: TR-04-CSE-05
April 16, 2004

ABSTRACT

As email becomes one of the most convenient and indispensable communication mediums in our life, it is very important to protect email users from increasing email worm attacks. In this paper, we present the architecture and system design of a "feedback email worm defense system" to protect email users in enterprise networks. The defense system is flexible and able to integrate many existing detection techniques to provide effective and efficient email worm defense. First, in response to a "detection score" of a detected worm email and information on the possible appearance of a malicious email worm in the global Internet, the defense system adaptively chooses a cost-effective defense action that can range from simply labelling this email to aggressively deleting it from an email server. Second, the system uses "honeypot" [13] to thoroughly detect worm emails received by email servers and also to early detect the presence of an email worm in the global Internet. Third, the defense system implements a "multi-sifting detection" technique and "differential email service" to achieve accurate detection without causing much delay on most emails. Furthermore, the defense system separates email attachments from email texts and saves attachments in separate "attachment caching servers", which facilitate both email worm detection and email service efficiency.

1. INTRODUCTION

"Email worms" are malicious computer programs that propagate through email attachments: when an email user clicks and executes a worm program in an email attachment, the worm runs with the email user's privilege to compromise the user's computer; then it finds all email addresses stored on this computer and sends out worm emails to these addresses.

Email is one of the most convenient and indispensable communication mediums in our life. However, email worms keep attacking us with increasing intensity and using more advanced social engineering tricks. *Melissa* in 1999, "*Love Letter*" in 2000 and "*W32/Sircam*" in 2001 spread throughout the Internet and caused millions or even billions of dollars in damage [19][20][4]. In 2003, the "*SoBig*" series [5] attacked the Internet several times with the goal of creating spam proxies on compromised computers [24]. In January and February 2004, "*MyDoom*" infected more computers than any previous email worm by using clever social engineering techniques to lure email users to execute worm code attachments [6]. In most cases MyDoom hid its worm code in a compressed attachment file. To prevent anti-virus software from checking email attachments, the recent "*Bagle*" series email worms [7] began to use password-protected compression files in email attachments with the corresponding password hidden in email text.

Another major class of worms, "scan-based" worms, find vulnerable computers by scanning IP addresses and compromising them through a vulnerable TCP/UDP service port. In the last several years, several major scan-based worms attacked us, including Code Red [8] in 2001, and Slammer [16] and Blaster [9] in 2003. However, recent trends show that attackers use email worms to attack the Internet more frequently than scan-based worms — email worms do not require a vulnerability to compromise computers and many email users still trust most emails they receive, which make email worms easier to program while providing a larger population to infect than scan-based worms.

Current email worm defense relies on signature-based anti-virus software to filter out worm emails on email servers. Such a signature-based approach cannot defend our email system as more and more attackers generate various new email worms (or polymorphic email worms) and use password-protected email attachments [7].

This paper investigates the challenges and presents the architecture and system design of a "feedback email worm defense system" to protect email users in local networks, especially enterprise networks. The defense system is flexible and able to incorporate many existing detection techniques together to provide effective and also efficient email worm defense.

Many different defense actions exist, ranging from simply labelling detected worm emails to aggressively deleting them on email servers without notifying their recipients. The email worm defense system in this paper implements a "*feedback defense*" — it takes a more aggressive defense action on an incoming email when the email is given a higher "detection score" by the system's detection components, or when it is known that a malicious email worm is currently spreading in the global Internet.

A "honeypot" is a closely monitored decoy computer that attracts attacks for early detection and in-depth adversary analysis [13]. To accurately detect new email worms or polymorphic email worms in incoming emails, the defense system deploys one or several "honeypots" to execute suspicious email attachments to detect email worms. The honeypots are designed to not send out email in normal situations. If a honeypot begins to send out emails after running the attachment of an email, we determine that this email is an email worm. We also implement runtime detection techniques on honeypots to detect email worm attachments.

Incoming email servers save email attachments in separate "attachment caching servers" for further security check, and replace these attachments in their original emails with corresponding download links. Email servers implement "*differential email service*" to serve emails without attachments or with obviously good attachments (such as a PDF file) more quickly than emails with suspicious attachments. In addition, the defense system deploys a "*multi-sifting detection*" technique to detect obviously good or bad emails by a fast signature-based detection approach and leave all hard-to-tell suspicious emails for further more thorough security checks such as statistical-based detection and honeypot detection. In this way, only emails with suspicious attachments are delayed by the defense system without affecting most users' email communications.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 presents the architecture of the feedback email worm defense system. The major principles in designing the defense system are introduced in Section 4. Section 5 explains in detail the feedback defense of the system. We provide some discussions in Section 6 and finally conclude the paper with Section 7.

2. RELATED WORK

Zou *et al.* [25] presented a feedback dynamic quarantine defense framework for scan-based worms based on the "feedback adjustment" principle borrowed from epidemic disease control. The basic idea is to take more aggressive defense actions when the epidemic of a worm is more apparent and serious [25]. In this paper, we will implement such a feedback principle in the email worm defense system.

Gupta *et al.* [12] used network traffic based anomaly detection techniques to detect the presence of an email worm in the Internet. Although it is helpful, our objective, however, is to protect email users' computers from being infected. Thus a more useful detection is to detect malicious code in every attachment of incoming email. Bhattacharyya *et al.* [2] proposed a "Malicious Email Tracking" system to detect and filter email worms. However, it assumes that an email worm has exactly the same attachment in every email and relies on MD5 sums for worm identification, which can be easily defeated by polymorphic email worms. Schultz *et al.* [22] used a data mining approach to detect malicious executable in email attachment based on pattern learned from training data. The data mining and other statistical-based anomaly detection approaches have the advantage of being able to detect unknown email worms, but generally need more time and computational resources than signature-based approaches. Our email worm defense system incorporates both classes of detection approaches to take advantage of their strengths while overcome their weaknesses.

"Honeypot" and "honeynet" [13] are effective in Intrusion Detection and could provide in-depth examination of hackers' or worms' attacks. Recently, Levine *et al.* [15] proposed to use honeynet to detect malicious traffic in enterprise networks. However, they mainly talked about detecting scan-based worm attacks and hackers' intrusion attacks. Qin *et al.* [21] used honeypot to detect worm infection by configuring a honeypot to not generate any outward traffic in normal situations: a honeypot is infected when it sends out network traffic. In our paper, this is one of the two principles used by honeypots to detect email worms by executing email attachments. Balzer [1] proposed a wrapper to monitor the runtime behavior of opened email attachment to detect and quarantine email worms. It is effective only when worm attachments are executed, thus such a wrapper has to be used on most email users' computers. Since our defense system implements honeypots, however, such a runtime monitoring technique is an ideal technique to implement in a honeypot for email worm detection.

In the email worm propagation research area, Garetto *et al.* [11] presented analytical techniques to study email worm behavior based on *Interactive Markov Chains*. Zou *et al.* [26] used simulations to study email worm propagation in the Internet and studied the topological effect by considering power law, small world and random graph topologies. They also considered immunization effects on email worm propagation on these topologies. Briesemeister *et al.* [3] conducted a similar research of epidemic spreading in scale-free (power-law) networks.

In a static analysis of email worms, Newman *et al.* [18] presented a percolation theory for arbitrary topologies, which can be used to derive how many percentage of computers need to be immunized in order to prevent an email worm from spreading out. In the email topology study, Zou *et al.* [26] showed that Yahoo group email list has a power law distribution. Newman *et al.* [17] collected email address book data from a large university and showed that the email topology has an exponential and a stretched exponential distribution for in-degree and out-degree, respectively.

Separating email attachments from email text has been introduced in the "email attachment caching" solution by Accellion Inc. [23], which shows that 80% to 90% of email bandwidth is consumed by attachments. However, the company uses this idea for the purpose of improving the performance of an email server — fast email service and encrypted attachment delivery. It did not consider this technique for the purpose of defending email worms. On the other hand, since our email worm defense system uses this technique, the defense system can provide the service benefit of email attachment caching as explained in [23].

3. EMAIL WORM DEFENSE SYSTEM ARCHITECTURE

Fig.1 illustrates the architecture of the feedback email worm defense system for an enterprise network. Basically, it includes two units: one is the "early warning unit" that provides early warning and information of a new email worm in

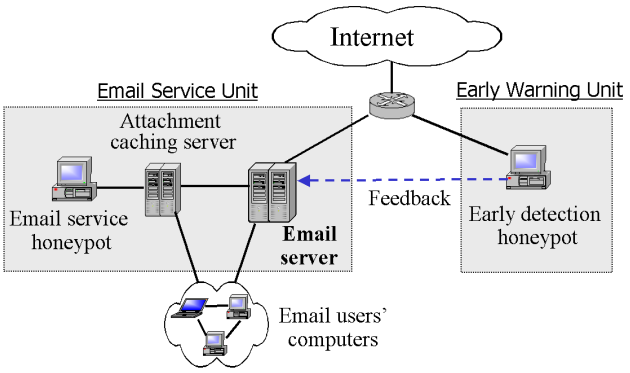

Figure 1: Architecture of the feedback email worm defense system for an enterprise network

the global Internet; another is the "email service unit" that provides secure email service to email users in an enterprise network. The defense system consists of the following components:

(1). Incoming email servers;
(2). Attachment caching servers;
(3). Email service honeypots;
(4). Early detection honeypots.

The "email servers" in Fig. 1 are incoming email servers that receive emails for users in an enterprise network. We do not need to defend outgoing email servers unless we want to prevent infected computers inside our network from sending out worm emails through outgoing email servers. To send out emails quickly and to relieve burden from incoming email servers, an enterprise could use separate outgoing email servers.

The defense system has installed basic email worm detection software on incoming email servers, such as anti-virus software and other signature-based detection software. This first-round detection is required to be fast and very strict in detection — its purpose is to quickly pass obviously "good" emails through and stop all suspicious emails for a further more thorough security check. For these suspicious emails, email servers save their attachments in separate "*attachment caching servers*" for further thorough security check and replace these attachments in their original emails by corresponding download links.

One or several "*email service honeypots*" are used to check suspicious email attachments saved on attachment caching servers for potential email worms. Several virtual operating systems can be run on each honeypot machine using virtual infrastructure such as VMware [14]. An email service honeypot executes an email attachment on its virtual operating systems to determine whether it contains worm code or not.

Email servers of an enterprise need to deal with heavy email traffic, especially during the propagation of a wide-spread email worm. Fortunately, email security checks can be executed in parallel on different emails. Therefore, the email worm defense system shown in Fig. 1 can deploy a cluster of inexpensive computers as "email service honeypots" and "attachment caching servers". We can simply increase the number of computers in the cluster in order to decrease the email service delay caused by the worm detection conducted by the defense system. No specialized hardware is needed for the email worm defense task. This is one important reason for separating email attachments from emails and using separate attachment caching servers.

In addition to dealing with incoming emails to email users in an enterprise network, the defense system deploys an "early warning unit": one or several "*early detection honeypots*" are used to actively determine whether there is an email worm spreading in the global Internet or not. A simulation program runs in early detection honeypots to simulate many email users with faked email addresses. These "robot" email users receive emails coming to their addresses and execute every email attachment to detect the presence of an email worm in the global Internet as early as possible. When detecting an email worm, early detection honeypots extract statistical patterns or signatures of the email worm and send the early warning information to the "email service unit". Such early warning information helps the "email service unit" to be well prepared (even before email servers receive any worm email) and take appropriate defense actions to received worm emails according to the "feedback adjustment" principle [25].

4. EMAIL WORM DEFENSE SYSTEM DESIGN

4.1 Separate email attachment from an email's text

Email worms rely on users to execute email attachments to infect users' computers. Therefore, we are able to defeat email worms in an enterprise network if we can prevent worm email attachments from reaching email users in this network.

The first idea of the email worm defense system is to separate a suspicious email attachment from the email's text on an incoming email server and thoroughly check the attachment on a separate machine. Incoming email servers in an enterprise network contain simple email worm detection software to quickly check all incoming emails for the first round. They allow incoming emails without attachments or with obviously good attachments (such as a PDF or a JPG file attachment) to pass through quickly to email users' mailboxes. The remaining emails with suspicious attachments are split into two parts: the email texts and the attachments. The attachments are transferred to an "attachment caching server" for further security check and are replaced by download links embedded in their email messages.

After replacing attachments with corresponding download links, email servers do not deliver these suspicious emails' texts to users' mailbox until their attachments are cleared by further security check. When an email user receives one such modified email message, the user can easily download this email's attachment from attachment caching servers by one click on the download link in the email message.

As many email servers prohibit emails with executable at-

tachments such as .exes, .scrs and .pifs, recent email worms hide worm code inside a compressed attachment file such as .zip [6][7]. Since many email users rely on compressed email attachments to transmit multiple files or to transmit a big file to their friends, email servers cannot prohibit .zip or other compressed email attachments in email communication. Furthermore, to prevent such attachments from being checked by anti-virus software, attackers have begun to use password-protected compressed attachment in email worms, such as recent Bagle email worm [7].

Our defense system successfully defeats the password-protected attacking technique due to the separation of attachments from emails. When an incoming email has a password-protected attachment and email servers are not sure that it is a worm email or not, email servers save the attachment in an attachment caching server, replace the attachment with a web link and deliver the modified email text to its recipient — attachment caching servers provide a web-based interactive service. From the link in the email message, the email recipient can upload the attachment password and then download the attachment once it is cleared by the detection system.

As explained in the "email attachment caching" solution [23], separating email attachment from email message improves the service performance of email servers (emails become much smaller) and provides secure delivery of email attachments from attachment caching servers to email users. Therefore, the email defense system presented here not only defends against email worms, but also provides the above email service benefits.

4.2 Email worm detection by "email service honeypot"

We deploy one or several "*email service honeypots*" to thoroughly check suspicious email attachments saved on attachment caching servers for email worms. Each email service honeypot runs several virtual machines (through the use of emulators such as VMware [14]); on each virtual machine, suspicious email attachments are executed one by one. Email service honeypots are designed to not send out any email in normal situations [21]. Therefore, after executing an email attachment, if a virtual machine begins to send out emails with attachments, this executed attachment is claimed to contain an email worm. To prevent infecting others, email service honeypots are designed not to send any email out.

Facing such kind of detection by honeypots, attackers might program their future email worms to not send out emails immediately after compromising an email user's computer. This dormancy makes it difficult for an email service honeypot to quickly determine whether an attachment contains worm code or not. For this reason, email service honeypots also attempt to detect worm attachments by inspecting the runtime behavior of executed email attachments. For example, Balzer [1] proposed a wrapper to monitor runtime behavior of opened email attachment to detect and quarantine email worms. Email service honeypots can implement such a wrapper or other runtime detection techniques to detect worm code contained in email attachments.

4.3 Multi-sifting detection

As described in Section 2, many email worm detection techniques exist for checking email attachments — they have varying security strengths and require different amounts of computational resources and time. For example, signature-based detection is a fast checking technique, but has trouble detecting a polymorphic email worm or variants of a known email worm. On the other hand, an email service honeypot provides a much more thorough and accurate detection, but requires more computational resources and time. In other words, no detection technique is perfect — we should use different detection techniques in different situations.

The defense system (more precisely, the email service unit shown in Fig. 1) implements a "multi-sifting detection" technique to check incoming emails for email worms. "Multi-sifting detection" means that an incoming email might need to pass through several detection procedures in order to be determined whether it is a worm email or not. Each detection procedure classifies an email into three classes: *normal email*, *worm email*, or *suspicious email*. If the i-th detection procedure ($i = 1, 2, \cdots$), which is called "*Sifting-i*", classifies an email as a "worm email", it assigns a "worm detection score" to this email (i.e., how likely this email is a worm email) and informs email servers; if the detection procedure classifies an email as a "normal email", it informs email servers to serve this email; if the detection procedure classifies an email as "suspicious", it passes the email to the next detection procedure for further more thorough security checks.

Specifically, the email worm defense system in this paper deploys three detection procedures on incoming emails. They are illustrated in Fig. 2 and described in the following:

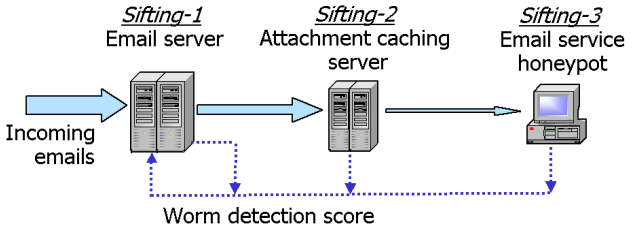

Figure 2: Multi-sifting detection (The size of each block arrow represents the number of emails)

Sifting-1: Incoming email servers conduct the first-round security check on incoming emails based on fast signature-based approach. This detection procedure is required to be fast to quickly pass good emails through, and to be accurate so as not to pass any worm email through to users. "Suspicious emails" (not classified as "normal emails" and "worm emails") are delayed and split into two parts (email text and attachment); their attachments are transferred to "attachment caching servers" for the next detection procedure.

Sifting-2: Attachment caching servers deploy various more sophisticated statistical-based email worm detection techniques (such as the data mining approach in [22]) to check email attachments. After this second-round security check, those attachments of emails that still cannot be classified as

"normal emails" or "worm emails" are transferred to "email service honeypots" for further check.

Sifting-3: Email service honeypots conduct the final check on those attachments that cannot be determined by attachment caching servers. Then return back detection score of these attachments to email servers.

After passing through the above multi-sifting detection procedure, an incoming email is classified as either a normal email or a worm email with a "worm detection score", which tells how likely this email is to be a worm email. Then email servers pass all normal emails through and take appropriate defense actions on worm emails according to their detection scores.

Most normal emails that have no attachments or have obviously good attachments (such as a PDF file) can be quickly checked and passed by the Sifting-1 detection procedure. Thus most normal emails can be served by email servers quickly without much delay. On the other hand, the multi-sifting detection has very low false negatives and false positives because of the extensive security check by Sifting-2 and Sifting-3 detection procedures. Therefore, the "multi-sifting detection" combines several detection techniques together to achieve accurate detection without causing much delay on most normal emails. In addition, because Sifting-2 and Sifting-3 need not check every incoming email, they do not impose high computer resource requirements on the email worm defense system.

4.4 Differential email service

It is a time-consuming task to thoroughly check whether an email attachment contains a worm or not, especially while using honeypots. To provide good security while maintaining email service efficiency, email servers in the defense system conduct "*differential email service*" in collaboration with the above multi-sifting detection. Emails are not served according to "first in, first out" principle; emails with a higher priority are served earlier than the ones with a lower priority. The email service priorities for normal emails, from the highest to the lowest, are:

(1). Emails without attachments;

(2). Emails with obviously good attachments (cleared by Sifting-1 detection procedure on incoming email servers);

(3). Emails whose attachments are saved in attachment caching servers and are cleared by Sifting-2 detection procedure on attachment caching servers;

(4). Emails whose attachments are saved in attachment caching servers and are cleared by Sifting-3 detection procedure on email service honeypots.

By using differential email service, most normal emails can be served quickly without delay, even when we implement a heavy-duty security check such as honeypot detection.

4.5 Early warning by "early detection honeypot"

The "email service unit" shown in Fig. 1 provides secure email service to users in an enterprise network. Its security can be further strengthened when we set up an additional "early warning unit", which attempts to actively detect the presence of a new email worm in the global Internet by one or several "*early detection honeypots*". If the "early warning unit" can detect the presence of an email worm in the Internet before this worm sends infection emails to email users in an enterprise network, the "email service unit" can be well prepared to better protect computers in this network. For example, after detecting an email worm, the early warning unit could possibly derive the email worm's statistical characteristics or signatures and feedback such information to the enterprise email service unit before incoming email servers receive any such worm email.

Each early detection honeypot runs a simulation program to simulate many email users called "robot email users". The early warning unit advertises email addresses of all robot email users to attract emails to these robot users. For example, the defense system could add their email addresses to many popular email lists. Robot email users execute all attracted email attachments they receive, but they are designed not to send out email under normal circumstances. If some of them begin to send out emails with similar pattern attachments, the early warning unit could claim that there is an email worm spreading in the global Internet.

When detecting an email worm, early detection honeypots extract worm code and try to derive the statistical patterns and signatures of the email worm. Then the email service unit (shown in Fig. 1) sets up its multi-sifting detection components according to these statistical patterns or signatures on incoming email servers, attachment caching servers, and email service honeypots. In addition, such early warning information also helps the email service unit to take appropriate defense actions on detected worm emails in incoming email servers.

Early detection honeypots directly receive emails sent to their created robot email accounts — we call such emails as "*attracted emails*" to distinguish them from "incoming emails" to real users in an enterprise network. These attracted emails will not pass through incoming email servers of an enterprise network, thus they will not affect normal email service. Early detection honeypots can easily deactivate some robot email accounts on them to control the amount of attracted email traffic, especially during the rapid spreading period of an infectious email worm.

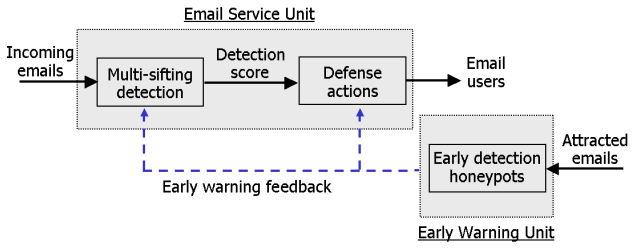

Figure 3: Logical architecture of the feedback email worm defense system shown in Fig. 1

After introducing each component of the email worm defense system, the logical architecture of the defense system is illustrated in Fig. 3, where the "multi-sifting detection" is showed in detail in Fig. 2.

5. FEEDBACK DEFENSE FOR EMAIL WORMS

5.1 Multiple defense actions

When defending against email worms, many enterprises have implemented various defense actions against incoming worm emails detected on their email servers. Instead of one single defense action, our email worm defense system deploys multiple defense actions adaptively. These defense actions are listed below in the order of their aggressiveness:

(1). Label a detected email as an "email worm" and add a warning message in the email to remind its recipent;

(2). Modify the attachment file type in addition to adding a warning message in a detected email, such as changing an executable attachment to a non-executable file;

(3). Save the attachment of a detected email on attachment caching servers, add a warning message and a download link of the email attachment in the email text. Then send only the text part of the email to its recipient. The recipient could decide whether to download the email attachment from attachment caching servers;

(4). Add a warning message in a detected email and only send the text part of the email to its recipient while deleting the email's attachments;

(5). Delete a detected worm email directly on incoming email servers without notifying its recipient.

Different defense actions have different false alarm costs and different strengths for protecting email users from email worms. For example, adding a warning message in email text or changing an attachment's file type generates a low false alarm cost because no email is deleted by email servers. However, this is not effective to defend against email worms — after receiving many such modified emails, email users may become annoyed, begin to ignore warning messages, and change back the file type of modified attachments. On the other hand, deleting detected worm emails on email servers can effectively protect email users' computers and also keep users from being annoyed by the large amount of worm emails. However, such defense action has a large false alarm cost that some email users cannot afford.

5.2 Feedback defense against email worms

Zou *et al.* [25] presented a "feedback adjustment" principle, meaning that more aggressive defense actions should be taken when the epidemic of a worm is more apparent and serious.

We implement this feedback principle in our email worm defense system. It has two feedback defense loops: one is the system-level feedback defense according to the early warning information from the "early warning unit" as shown in Fig. 1, which determines the overall defense action of the email worm defense system; another is the email-level feedback defense according to the detection score of individual incoming email, which determines what appropriate defense action should be taken on this incoming email only.

5.2.1 System-level feedback defense

When "early detection honeypots" detect an email worm, they attempt to extract its statistical pattern and signature. Then they feed back such early warning information to the email service unit as shown in Fig. 3. All detection components in the multi-sifting detector in the email service unit update their detection parameters according to this feedback to more accurately detect email worms in incoming emails. As early detection honeypots keep receiving more worm emails, they derive the worm's statistical pattern and signature more accurately as time goes on. Therefore, early detection honeypots feed back early warning information to the defense system continuously during the spreading of an email worm.

In addition, such early warning information also helps the email service unit implement more appropriate defense actions — such feedback is also shown in Fig. 3. If early detection honeypots receive a large number of worm emails in a short time, this detection tells us that the email worm is dangerous and is rapidly spreading in the global Internet. Then according to the "feedback adjustment" principle [25], the defense system would take more aggressive defense actions based on such feedback. For example, increase the checking strictness in the "multi-sifting detection", or increase the aggressiveness of defense action from simply adding a warning message in a worm email to deleting attachment from a worm email.

5.2.2 Email-level feedback defense

In recent years, email worms have become more complex and covert. In many cases, one cannot tell for sure whether an email contains a worm code attachment or a normal attachment. Instead, an email worm detection system can only provide a "worm detection score" for a checked email to say what is the possibility that this email contains a worm code attachment instead of a normal attachment. Therefore, according to the "feedback adjustment" principle, more aggressive defense action is taken on an email if this email is given a higher detection score by the multi-sifting detector in the email service unit (i.e., this email is more likely to be a worm email).

We have introduced above (in Section 5.1) five different defense actions on detected worm emails. For each detected worm email, the email-level feedback defense chooses the appropriate defense action on this email according to the email's detection score. If we are very sure that an email contains a worm attachment, the best defense action is to aggressively delete this email on email servers without notifying its recipient.

5.3 Feedback defense based on user profile

The feedback defense described above does not distinguish between email users or email accounts. In reality, different email users have different needs and different levels of security awareness. For example, email users without any

security knowledge might want the email worm defense system to take default defense actions (set up by security staff) for them; on the other hand, experienced email users might want to deal with worm emails by themselves and not have the defense system change attachment file types or delete any email attachment sent to them.

For this reason, flexibility can be provided to email users by allowing them to set up "user profiles", one for each email user in the network. Every email user can interact with the defense system (e.g., through a Web-based interface) to configure the user's "security preference", i.e., what defense actions should be taken by the defense system on detected worm emails sending to this user. In this way, the defense action on a detected worm email is not only based on the detection score of this email and the early warning from early detection honeypots, but also based on the user profile of the email's recipient.

6. DISCUSSIONS

In this section, we discuss possible future work and several open issues in email worm defense.

6.1 Detailed system design

The feedback email worm defense system presented here attempts to provide a system-level architecture and design principles for an effective email worm defense system, not a detailed defense algorithm. It is sufficiently flexible to incorporate many different detection and defense techniques together to provide effective and efficient email worm defense. For example, incoming email servers can use any existing signature-based fast detection software; "attachment caching servers" can use any statistical or anomaly detection algorithm; "email service honeypots" and "early detection honeypots" can deploy any runtime detection method to detect worm emails.

Consequently, we have not provided detailed designs for each component of the defense system. We have presented feedback design principle that can deal with false positives in a cost-effective way, but have not given detailed design on what defense action should be taken under a particular situation. In addition, we need to determine how to set up honeypots to detect worm code email attachments accurately and efficiently.

Our future work is to complete a detailed design of the system and then build a prototype to use in practice — first on the incoming email server of a department, then on the scale of a whole campus network.

6.2 Security checking on logical perimeter

The email worm defense system detects and controls worm emails on incoming email servers of an enterprise network. Incoming email servers comprise only one logical perimeter for emails coming to a network. When internal email users of an enterprise network use other email accounts, such as hotmail or yahoo email, the defense system presented here cannot check and defend email worms coming through these other logical perimeters. In this case, the security of an email user's computer is determined by the least secure email account used by the user.

For this reason, when possible, an enterprise network should have an email policy that its users should only check emails through their company's email accounts — emails in other accounts can be forwarded to users' company email accounts and be checked by the defense system. However, it is difficult to implement since web-based public email service is very popular and many people are not willing to forward their private emails to their companies' email accounts.

This problem appears to be fundamental, and cannot be solved by any network traffic based detection and defense system since many web-based email services provide encrypted HTTP communication. The good news is that most web-based public email services are relatively secure — they have great incentive and resources to beef up email security.

6.3 Email worms without attachments

The recent "Bagle.Q" email worm [10] propagates by sending emails without attachments: infected hosts create mini-http servers on them and include a web download link in its infection emails. When a user clicks on this web link in a worm email, the worm code is downloaded and then compromises the user's computer. Bagle.Q worm used predefined list of servers for worm download [10] that can be shutdown quickly by Internet community. However, a future email worm could use a web download link in its infection email pointing to the email's sender — a compromised computer that has a mini-http server installed. In this way, we cannot shutdown the worm code download channel, although this makes it easier for us to know which computers have been compromised.

We can detect such an email worm by checking URL links contained in its email. If a honeypot connects to every URL links in an email, it can detect whether a downloaded program from one of these links contains a worm. However, this is a very time-consuming security task. One possible way to quickly detect such an email worm is to check the format of URL links. If the worm uses an IP address for its URL link, it is easy to detect because most (if not all) normal servers use domain names instead of IP addresses. Many online computers that are not web or ftp servers have domain names, too; but in most cases we can tell whether they are servers or not from their domain names. For example, an ordinary DHCP computer has a domain name similar to "res-234-16.dialup.umass.edu" or "h05da7d4.ne.client.attbi.com".

7. CONCLUSIONS

In this paper, we present an effective and practical feedback email worm defense system to protect email users in enterprise networks. First, in response to a "detection score" of a detected worm email and information on the possible appearance of a malicious email worm in the global Internet, the defense system adaptively chooses a cost-effective defense action that can range from simply labelling this email to aggressively deleting it from an email server. Second, the system uses "honeypot" [13] to thoroughly detect worm emails received by email servers and also to early detect the presence of an email worm in the global Internet. Third, the defense system implements a "multi-sifting detection" technique and "differential email service" to achieve accurate detection without causing much delay on most emails. Furthermore, the defense system separates email attachments

from email texts and saves attachments in separate "attachment caching servers", which facilitate both email worm detection and email service efficiency.

The feedback email worm defense system presented here attempts to provide a system-level architecture and design principles for email worm defense, not a detailed defense algorithm. Our objective is to provide an email worm defense system that is sufficiently flexible to incorporate many existing detection and defense techniques together to provide effective and efficient email worm defense. There are still a lot detailed works to do to finalize the defense system for practical use. In addition, we need to conduct more research to deal with email worms that send out infection emails without attachments, and study how to protect users from worm emails coming from their public email accounts other than their companies' email accounts.

8. ACKNOWLEDGEMENT

This work is supported in part by ARO contract DAAD19-01-1-061, the National Science Foundation Grants ANI9980552, and by DARPA contract F30602-00-0554.

9. REFERENCES

[1] R. Balzer. Assuring the safety of openning email attachments. In *Proceedings of DARPA Information Survivability Conference & Exposition II*, June 2001.

[2] 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)*, September 2002.

[3] L. Briesemeister, P. Lincoln, and P. Porras. Epidemic profiles and defense of scale-free networks. In *Proceedings of ACM CCS Workshop on Rapid Malcode (WORM'03)*, October 2003.

[4] CERT. Cert advisory ca-2001-22 w32/sircam malicious code. http://www.cert.org/advisories/CA-2001-22.html, July 2001.

[5] CERT. Cert incident note in-2003-03: W32/sobig.f worm. http://www.cert.org/incident_notes/IN-2003-03.html, 2003.

[6] CERT. Cert incident note in-2004-01: W32/novarg.a virus. http://www.cert.org/incident_notes/IN-2004-01.html, January 2004.

[7] Symantec Corp. W32.beagle.f@mm. http://securityresponse.symantec.com/avcenter/venc/data/w32.beagle.f@mm.html, February 2004.

[8] eEye Digital Security. .ida "code red" worm. http://www.eeye.com/html/Research/Advisories/AL20010717.html, 2001.

[9] eEye Digital Security. Blaster worm analysis. http://www.eeye.com/html/Research/Advisories/AL20030811.html, 2003.

[10] F-Secure. F-secure virus descriptions : Bagle.q. http://www.f-secure.com/v-descs/bagle_q.shtml, March 2004.

[11] M. Garetto, W. Gong, and D. Towsley. Modeling malware spreading dynamics. In *Proceedings of INFOCOM*, April 2003.

[12] Ajay Gupta and R. Sekar. An approach for detecting self-propagating email using anomaly detection. In *Proceedings of Recent Advances in Intrusion Detection (RAID)*, September 2003.

[13] Honeypot. http://www.honeypots.net/.

[14] VMware Inc. Vmware: Virtual infrastruture. http://www.vmware.com/vinfrastructure/.

[15] John Levine, Richard LaBella, Henry Owen, Didier Contis, and Brian Culver. The use of honeynets to detect exploited systems across large enterprise networks. In *Proceedings of the 2003 IEEE Workshop on Information Assurance*, 2003.

[16] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Magazine on Security and Privacy*, 1(4), July 2003.

[17] M. Newman, S. Forrest, and J. Balthrop. Email networks and the spread of computer viruses. *Phys. Rev. E.*, 66(035101), 2002.

[18] M. Newman, S. Strogatz, and D. Watts. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E.*, 64(026118), 2001.

[19] CBS news. Melissa virus' author owns up. http://www.cbsnews.com/stories/1999/12/09/tech/main73910.shtml, December 1999.

[20] CNN news. Love bug costs billions. http://money.cnn.com/2000/05/05/technology/virus_impact, May 2000.

[21] X. Qin, D. Dagon, G. Gu, and W. Lee. Worm detection using local networks. Technical report, College of Computing, Georgia Tech., Feburary 2004.

[22] M. G. Schultz, E. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2001.

[23] Osterman Research white paper. The role of attachment caching in e-mail server consolidation. http://www.ostermanresearch.com/whitepapers/download07.htm, March 2004.

[24] MessageLabs Whitepaper. The convergence of viruses and spam lessons learned from the sobig.f experience. http://www.messagelabs.com/microsites/MessageLabs, 2003.

[25] C. C. Zou, W. Gong, and D. Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In *Proceedings of ACM CCS Workshop on Rapid Malcode (WORM'03)*, October 2003.

[26] C.C. Zou, D. Towsley, and W. Gong. Email virus propagation modeling and analysis. Technical Report TR-03-CSE-04, Umass ECE Dept., May 2003.