Addressing Malicious SMTP-based Mass-Mailing Activity Within an Enterprise Network

David Whyte* P.C. van Oorschot* Evangelos Kranakis*

Abstract

Malicious mass-mailing activity on the Internet is a serious and continuing threat that includes mass-mailing worms, spam, and phishing. A mechanism commonly used to deliver such malicious mass mail is an *SMTP-engine*, which turns an infected system into a malicious mail server. We present a technique that enables, in certain network environments, detection and containment of (even *zero-day*) SMTP-engine based mass-mailing activity within a single mailing attempt. Contrary to other mass-mailing detection techniques our approach is *content independent* and requires no attachment processing, statistical measures, or system behavioral analysis. It relies strictly on the observation of DNS MX queries within the enterprise network. Our approach can be used as an alternative to port 25 blocking and in conjunction with current proposals to address mass-mailing abuses (e.g. SPF, DomainKeys). Our analysis on network traces from a medium sized university network indicates that MX query activity from client systems is a viable SMTP-engine detection method with a very low false positive rate. Our detection and containment approach has been successfully tested with a prototype using a live mass-mailing worm in an isolated test network.

1 Introduction

Internet users are inundated by a steady stream of emails infected with malicious code (mass-mailing worms and viruses), unwanted product advertisements (spam), and requests for personal information from criminals masquerading as legitimate entities to enable the commission of fraudulent activity (phishing).

The use of gateway anti-virus (and per client) software and spam filters offers some measure of protection. However, these perimeter defences often fail to detect *zero-day* worms and viruses, often quarantine legitimate emails misidentified as spam, and do not address perhaps the most prevalent infection method: users unwittingly opening malicious attachments. A strong argument can be made that the best chance to detect and quarantine malicious email occurs within the enterprise network before it can be sent.

To date, the use of mass-mailing worms has been the fastest way to propagate malicious mail.[1] For example, the MyDoom mass-mailing worm (purportedly the fastest spreading mass-mailing worm to date) at its peak was responsible for one in every twelve email messages on the Internet [6]. The majority of mass-mailing worms employ the same infection delivery mechanism: a *Simple Mail Transfer Protocol engine (SMTP-engine)*. SMTP-engines turn an infected system into a malicious mail server for the worm. The effectiveness of SMTP-engines as a malicious email delivery mechanism has been noted by other malicious communities of interest. As mail server filtering techniques become more effective, spammers and phishers are resorting to hijacking ordinary PCs (called *zombies*) and using built in SMTP-engines or mail proxy programs to send malicious mail without the owner's knowledge [4, 17]. In fact, it has been estimated

*{dlwhyte, paulv, kranakis}@scs.carleton.ca. School of Computer Science, Carleton University, Ottawa, Canada.

[1] We define malicious mail as unwanted email unwittingly sent by a compromised system whether or not it contains malicious code (e.g. spam albeit unwanted typically only contains an advertisement).

that 80% of spam is sent by spam zombies [17]. A detailed description of how SMTP-engines are used by mass-mailing worms, spam zombies, and phishers is included in Appendix A.

In this paper, we show how the interaction between SMTP-engines and DNS servers offers a method to detect malicious mass-mailing activity within an enterprise network. SMTP-engine infected clients typically request Mail Exchanger (MX) records from a DNS server (either their local DNS server or DNS servers outside the network boundary) in order to locate the mail servers that can deliver the malicious mail to their intended victims. While some legitimate client systems run their own email servers locally, most enterprise environments use perimeter mail servers to send and receive email.[2] In this scenario, only the corporate mail servers within the enterprise network are generally expected to query DNS servers for MX records (see further discussion, including exceptions in Section 4.2).

We present a technique, implemented and tested with a software prototype, to detect and quarantine SMTP-engine mass-mailing based solely on the observation of a DNS MX record request from client systems. No modeling or statistical measurement of user or network behavior is required. Furthermore, it does not rely on attachment scanning, allowing detection of malicious text-based emails with hypertext embedded links to malicious websites.[4] To validate these claims, we performed tests in an isolated test network with a *live* mass-mailing worm.

Promising current techniques to address malicious mass-mailing activity (e.g. SPF [18], DomainKeys [9]) offer a way to identify forged mail as it is received by network mail servers. However, these techniques are not designed to detect or stop malicious mass-mailing that bypasses authorized mail servers (e.g. SMTP-engines) before it leaves the enterprise network. Our approach could be applied as a complementary technique to both rapidly and accurately detect and contain malicious SMTP-engine mail propagation within an enterprise network (i.e. stopped at the sending end instead of identified and stopped at the receiving end after traversal of the Internet). In contrast to port 25 blocking, our approach is flexible enough to allow mail relaying from internal users, where this practice is permitted.

Our anomaly-based approach is appealing for a number of reasons including the following:

1. *Speed*: in certain network environments the possibility to detect and contain an SMTP-engine before a single malicious email message can be sent.
2. *Detection and containment of zero-day mass-mailing worms*: possible because the approach does not rely on existing worm signatures.
3. *Impact to quarantined system*: once identified as a malicious mass-mailer, only SMTP activity (port 25) will be blocked on the system allowing all other user activity to proceed unhindered.
4. *Low-false positive rate*: empirical analysis (see Section 4.2) suggests that client MX record requests are rare for most users. [3]
5. *Ease of deployment*: the approach is network-based, runs on commodity hardware, and relies on the observation of a protocol found in all networks (i.e. DNS).

The sequel is structured as follows. Section 2 discusses related work. Section 3 outlines the basic approach of the detection technique. Section 4 presents an empirical analysis of client MX record request activity. Section 5 discusses our prototype and its performance in an isolated worm test network. Section 6

[2]This allows for gateway anti-virus software at the network perimeter and a more cost effective way (e.g. maintenance, support and troubleshooting, policy enforcement, etc.) to manage corporate email.

[4]These websites infect a system when the URL is accessed by sending malicious code through website content retrieved by the client system.

[3]Indeed, we observed a university network of approximately 300 users over a one week period and found only 5 anomalous MX record queries from client systems. Although we realize that in most corporate environments the deployed software application baseline is substantially different than a university network, the greater software diversity in the latter makes it a good test environment.

contrasts our technique with other approaches. We conclude in Section 7. Appendix A provides background on the malicious use of SMTP-engines.

2 Related Work

Zou et al. [32] developed a mass-mailing worm model by profiling the user behavior of email checking times and email attachment opening probabilities. Their analysis determined that networks, if viewed as logical networks grouped by email addresses, seem to have a heavy-tailed distribution which can be modeled as a power law network. They analyzed the impact of *selective immunization defense*, that entails making the most connected email users' systems immune to an email worm. Their results reveal that although a power law topology enables a worm to spread more quickly, it also allows for faster containment. Their work provides an email worm model that incorporates user behavior and offers some insight into worm propagation on a number of network topologies. The same authors propose [31] a multi-step feedback email defence mechanism to detect malicious email within an enterprise network. Additionally, they suggest the use of a *honeypot* to detect outgoing viruses.

Sidiroglou et al. [26] propose an architecture to detect zero-day worms and viruses. Their approach involves intercepting every email and scanning them for dangerous attachments. They employ virtual machine clusters, host-based intrusion detection, and *email-worm-vaccine* aware Mail Transfer Agents to handle malicious emails.

Hu et al. [12] present an application of the PAIDS *(ProActive Intrusion Detection System)* detection paradigm using a prototype system called BESIDES which detects mass-mailing viruses. PAIDS employs two general techniques that involve comparing a system's behavior against its security policy (*behavior skewing*) and isolating illegal system behaviors in a virtual environment (*cordoning*). Their prototype was successful in detecting a number of *real* mass-mailing worms with a low false positive rate. However, their implementation is deployed at SMTP servers which would fail to detect SMTP-engine activity. SMTP-engines bypass network mail servers (and even in some cases local DNS servers) making network-based detection techniques necessary.

Gupta et al. [11] use *specification-based* anomaly detection to detect email viruses. Their approach looks for increases in mail traffic from clients to mail servers over a threshold determined during a training period. Specifically, the statistics of send and deliver transitions in a state machine are maintained for both individual clients and the entire collection of clients within the network. Using a series of simulated experiments they were able to detect stealthy viruses (e.g. polymorphic) with a low false positive rate.

Our technique is motivated in part by previous work specifically related to DNS activity and mass-mailing worm detection. Wong et al. [30] performed an empirical study on mass-mailing worm behavior using network traffic traces from a college campus. The characteristics of two mass-mailing worms with respect to DNS activity and TCP traffic flows were studied. They found that changes in network activity from infected hosts allowed for interesting detection possibilities. They propose that a more in-depth investigation of monitoring and containing mass-mailing worms using DNS servers should be performed as it holds promise as a way to slow down propagation. One important observation made was that defences designed for monitoring SMTP servers will not work well for mass-mailing worms as they have their own SMTP-engines.

Whyte et al. [28, 29] used DNS activity to detect the presence of scanning worms within an enterprise network. The observation of connections outside the network not preceded by a DNS query was considered anomalous and a strong indicator of scanning worm activity. They hypothesized that MX queries from client systems could indicate mass-mailing worm infection, but recognized that the detection and containment of mass-mailing worms would require the collection of different network data (i.e. a data set with substantial mail activity) and a subtly different approach that was out of scope with the scanning worm detection tech-

nique. In contrast to their work we: (1) implement a new detection paradigm, (2) construct a prototype that processes DNS MX records and performs containment, (3) analyze a much larger network trace (i.e. a network that services approximately 300 clients) that includes SMTP activity, and (4) implement containment as opposed to pure detection.

Finally, closely related work on this subject was performed by Musashi et al. [22, 21, 20]. In apparently independent work from [28], they also recognized that MX query activity from client systems could indicate mass-mailing worm infection. They developed an *indirect virus detection system* (MXRPDS) that detects mass-mailing worm infection by monitoring DNS server and PC terminal interaction. In their implementation, they poll the DNS server syslog file every 10 seconds to determine client queries of A, MX and PTR records. Any client that accesses the DNS server for MX and A records without PTR records is considered to be infected with a mass-mailing worm. Clients that request a mixture of MX, A, and PTR records are considered to be spam relays. Their DNS host-based approach has a number of disadvantages compared to our network-based implementation. Specifically, a host-based approach does not address a common technique employed by SMTP-engines to obtain MX records by querying both local and remote DNS servers. Parsing of a local DNS's syslog file will not detect remote DNS accesses and introduces significant false negatives. Additionally, processing the DNS syslog every 10 seconds allows a newly infected system to remain active during this time sending potentially hundreds of malicious emails. Finally, they propose no way to quarantine the infected systems once detected.

For a discussion of alternate proposals to address malicious mass-mailing activity, see Section 6.

3 Basic Methodology and Approach

In this section, we contrast normal email delivery with email sent from a host with an SMTP-engine and present a high-level overview of our mass-mailing detection approach.

3.1 Normal Email Delivery

In the next two sections, we assume an enterprise or corporate environment. Generally, a user accesses email client software on their system to generate an email message. The client software is responsible for sending the email to the mail server specified in its configuration file. Then, the mail server sends and delivers email on behalf of the users within its domain.

In order to determine the IP addresses of the mail servers responsible for delivering mail to the intended recipients, DNS MX queries are made. An MX record identifies the mail server responsible for sending and delivering emails for a Fully Qualified Domain Name (FQDN). Figure 1 illustrates the steps required to send an email message.

1. *User to mail server interaction:* a user in the enterprise network uses their email client to compose an email for a recipient or list of recipients. Once completed, the email client forwards the email to its local mail server for delivery.
2. *Mail server to DNS server interaction:* mail servers are store-and-forward systems. Once a mail server receives an email, it accesses the recipient list to determine where it must be delivered. The recipient list contains addresses of the form *user@host.domain* as specified in RFC 822 [8]. The *user* field will be a unique identifier for the particular domain. The *host.domain* field contains the host's FQDN. DNS servers use the FQDN to locate the mail servers that service the respective domain. As shown in Figure 1, the local DNS server happens to have the MX record in its cache for the recipient's domain and sends the IP address of the mail server identified in the MX record to the local mail server.

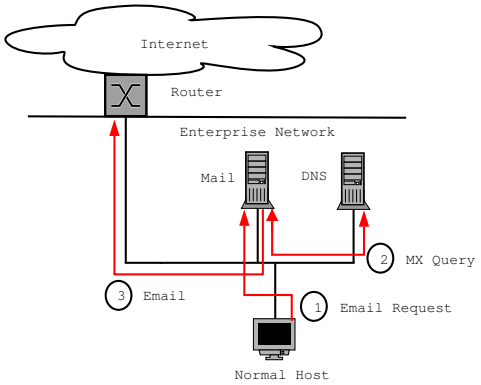
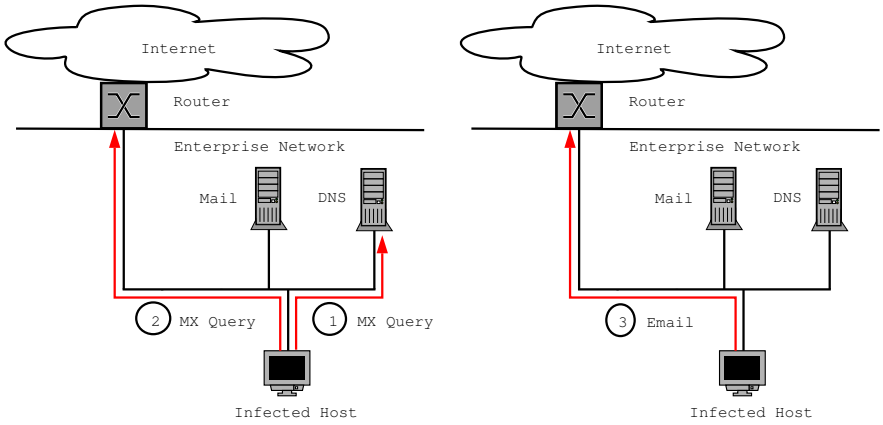


Figure 1: Normal Email Delivery.



(a) MX Record Request

(b) Malicious Mail Delivery

Figure 2: SMTP-engine Malicious Mass-Mailing Delivery.

3. *Mail server to mail server interaction:* using the IP address contained in the MX record, the local mail server sends the email to the intended recipient's mail server. In turn, the recipient's mail server sends the email to the local client of the user specified in the email address.

3.2 Malicious Email Delivery with an SMTP-engine

In contrast to a normal email generation, mass-mailing activity via SMTP-engines bypasses corporate mail servers when it attempts to send malicious mail.

Malicious mass-mailing software can either interrogate the host system to harvest email addresses (e.g. mass-mailing worm) or be supplied with a recipient list (e.g. spam) to send the malicious messages. In either case, here the SMTP-engine of the infected system is responsible for sending the malicious mail messages directly. In order to determine the mail server that services a particular recipient, the infected system, not the local mail server, queries a DNS server for an MX record associated with the email recipient's FQDN. Figure 2 illustrates the steps an infected host with an SMTP-engine performs to send an email message.

1. *Infected host to local DNS server interaction:* an internal system in the network is infected with malicious mass-mailing software that includes an SMTP-engine. To send mail, the infected system must forward MX queries to a DNS server. As shown in Figure 2(a), 1, the query is sent to the local DNS server which happens to have in its cache the MX record for the recipient's domain and sends the MX record to the infected system.
2. *Infected host to external DNS server interaction:* alternately, the infected system can query an external DNS server (i.e. Figure 2(a), 2). for an MX record.[5]
3. *Infected host to mail server interaction:* the infected system sends the malicious email to the mail server responsible for the recipient specified in the email address. The local mail server is bypassed completely.

3.3 Detection Approach

Malicious mass-mailing software that use SMTP-engines bypass local mail servers but must still rely on DNS servers to locate the respective mail servers of their intended victims. Client-based MX requests is a violation of typical DNS behavior in the network.

To detect SMTP-engine malicious mass-mailing activity we simply observe all locally generated MX queries that originate from systems other than the (known) network mail servers. These systems are regarded as potentially infected and after a certain number (configurable within our prototype) of MX queries are observed, they are quarantined from the network. Quarantining a system involves restricting it from directly performing any SMTP (i.e. port 25) network activity. Note that this differs from blocking port 25 activity of all (non-server) systems.

4 MX Record Activity Analysis

Our detection technique relies on the fact that MX query activity from ordinary client systems is distinguishable from those that perform mass-mailing. To test this hypothesis we monitored the internal client accesses of two DNS servers (a primary and secondary DNS server) for a medium sized departmental network within our university.

4.1 Network MX Record Activity Analysis

To understand the prevalence and behavior of MX record activity within a network of diverse clients, we observed a network that services a population of approximately 300 client systems used by faculty, administrative staff, and students (Department Network in Figure 4). These systems contain a variety of operating systems that include Windows platforms, Linux, BSD, and SunOS. We monitored all internal (within the department network) and external accesses (outside of the department network including the Internet) of both DNS servers over a one week period from March 18 to March 24, 2005. Table 1 shows the total DNS record activity for both DNS servers.

DNS A (authoritative) records are the most active type of DNS records observed. This is expected as DNS A records provide the mapping between the numeric IP address of a system and its FQDN. DNS A records are required for most routine connection requests between remote systems (e.g. HTTP). MX record activity is the second most requested DNS resource record. Figure 3 contrasts total DNS record activity during the week against MX record activity.

[5]For instance, during the SoBig.F outbreak, Verisign discovered DNS MX lookups from tens of thousands of systems to its root DNS server [14].

Table 1: One-week Survey of DNS Record Activity.

Record Type	Number of Records
PTR	194 140
AAAA	99 019
SOA	17 800
A	2 074 620
CNAME	72
MX	211 697
NS	4 056

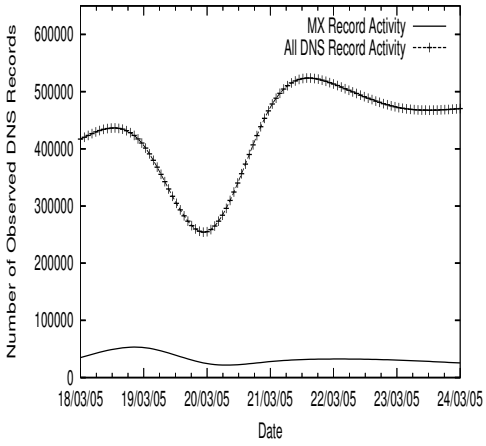


Figure 3: MX Record Activity.

4.2 Client MX Record Activity Analysis

MX record requests from external systems and internal mail servers are a normal occurrence. We analyzed the MX query activity within our network to determine if any client systems (i.e. not authorized mail servers) performed any MX queries. Table 2 shows that of the approximately 300 internal systems serviced by the two DNS servers, only five clients made MX record requests during the one week analysis period. Two of these (10.0.0.68 and 10.0.0.42)[6] made a total of 1705 MX record requests to 133 unique FQDNs. System 10.0.0.68 is owned by a network administrator who, as part of a strategy to combat spam, was testing *SpamAssassin* [1]. We confirmed that the MX request activity in question from this system was performed as part of this software testing. System 10.0.0.42 was owned by a user. A quick inspection of the system configuration determined that this activity was the result of a mis-configured *cronjob* requesting nonexistent MX records (i.e. *localhost.localdomain*) from the DNS server.

The remaining three (10.0.0.36, 10.0.0.51, and 10.0.0.83) systems were responsible for a total of 5 MX record requests over the one-week test period. As the IP addresses that corresponded to these three systems were assigned via *DHCP*, the necessary logs to perform user attribution for these IP addresses do not exist. Therefore, an analysis of the cause of these MX queries was not possible. Given the low number of *unexplained* MX queries (i.e. 5) we conjecture that these are likely caused by isolated MX lookups (e.g. perhaps evidence of mail relaying through 3rd party mail servers). The conclusion we draw from this analysis is that most client systems within this network do not perform MX queries even though it is a

[6]IP addresses have been anonymized.

Table 2: MX Record Lookups.

IP Address	MX Requests	Unique MX Requests	Reason
10.0.0.68	1691	132	System admin SpamAssassin test system.
10.0.0.42	14	1	Mis-configured cron job sending mail to *systemname@localhost.localdomain*.
10.0.0.36	3	3	DHCP system - unexplained.
10.0.0.51	1	1	DHCP system - unexplained.
10.0.0.83	1	1	DHCP system - unexplained.

heterogeneous software environment (e.g. university network). If we assume this network is representative, our technique is generally viable as there are very few false positives.

However, there may be instances in which a user at a client system needs to legitimately access SMTP services directly and request MX records (e.g. a mobile user with a laptop wanting to relay mail through their own *home* mail servers). We believe this activity is discernible from mass-mailing activity and can easily be accommodated by any containment approach.

5 Prototype and Analysis

In this section, we describe our software prototype detection and containment system as a proof-of-concept. We also discuss its performance in detecting and containing a *live* mass-mailing worm within an isolated test network.

5.1 Prototype

To validate our SMTP-engine detection and containment technique, we developed and tested a fully functional software prototype. The software was installed on a commodity PC with a Linux operating system. The prototype processes network data in *real-time* and performs two distinct functions: (1) detection of SMTP-engine mass mailing activity, and (2) containment of systems that exhibit SMTP-engine mass-mailing activity. We now discuss these two functions in turn.

Detect. The only network data feature extraction required by the prototype to detect SMTP-engine mass-mailing activity is DNS MX queries. If any client system performs a DNS MX query (local or external) this is considered potential malicious mass-mailing activity. MX queries originating from authorized mail servers (or other systems authorized to use SMTP) are exempt from the detection algorithm through the use of a *whitelist*.

Contain. Once potential SMTP-engine mass-mailing activity is detected, the prototype uses *IPTables* [2] to stop all SMTP activity from the client. IPTables software is included within the Linux kernel and provides a generic specification of rule sets that allows for stateful packet filtering. When a client system, not enumerated within the whitelist, exceeds the number of allowed MX queries, a rule is added to IPTables that restricts port 25 (SMTP) activity (both outgoing and incoming) from that client's source address.

5.1.1 Configuration Discussion

With any containment device, false positives are an important concern. A balance must be struck between rapid detection and impact to users due to unwarranted containment. Our prototype can be configured to restrict SMTP activity after the observation of any number (e.g. 1 or more) of MX queries within a given

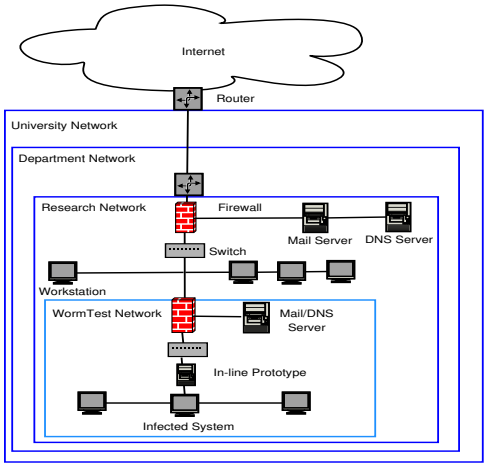


Figure 4: Mass-mailing Worm Testing Environment.

time interval. This flexibility enables the reduction of false positives (see Section 5.3.1), and the ability to allow mail relaying in the network if this activity is permitted (see discussion in Section 6.1).

Regardless, in our current implementation even if a false positive occurs a *contained* client system is only restricted from performing SMTP activity. The client is allowed unhindered access to all other network services. However, it could be argued that if we suspect a client system contains a malicious SMTP-engine, it may contain other active infection vectors (e.g. network share traversal, scanning). In this case, it may be a prudent containment decision to generate the necessary IPTables rules to restrict all network access from the client.

A further consideration for the prototype is network placement. The two most important placement considerations are (cf. Figure 2(a)): (1) enabling detection of all MX query activity (i.e. remote and local), and (2) the ability to restrict the network access of infected systems. Most SMTP-engines are configured to query the local DNS server for MX records first. In the event the local DNS server cannot be accessed, some SMTP-engines contain a list of remote DNS servers to query. The prototype should be placed where it can monitor all MX activity on the network (e.g. to also detect the use of external DNS servers). Furthermore, in order to restrict the SMTP activity of infected systems, the containment device must be placed at all egress points on the network.

5.2 Live Worm Network Testing

To conduct our prototype evaluation, we tested it against a *live* worm within an isolated network test environment. The network was used to: (1) observe the behavior of SMTP-engine mass-mailing systems, and (2) test the effectiveness of our prototype. The isolated worm test network is attached to a fully functional research network that in turn connects to a university department network. All of these networks (with the exception of the isolated worm test network) share part of our university's Class B IPv4 Internet address space. Figure 4 shows the orientation of the isolated worm test network.

To prevent inadvertent infection of systems during testing, we placed a firewall between the isolated worm test network and the research network. The firewall rules allowed only DNS traffic to enter or leave the isolated worm test network. Additionally, we physically (and logically) isolated all the worm test network IP addresses on a separate switch using a VLAN with non-routeable IP addresses (i.e. 192.168.1.0/24 [24]). To confirm the validity of our approach we infected a system within the worm test network with the NetSky.Q.mm mass-mailing worm [5] and observed its behavior for 10 minutes. Table 3 shows the network

Table 3: NetSky.Q.MM Network Activity (10 minute period).

Activity	Unique Requests	Total Requests
MX Record Queries	37	194
A Record Queries	24	24
SMTP connections	44	322

activity from the infected system. Within 10 minutes the system generated 194 MX queries to our local DNS server looking to resolve 37 unique mail server FQDNs. Additionally, after an initial burst of MX query activity, the infected system attempted to contact 44 external mail servers via SMTP.

Once the 10 minute observation period ended, we removed the worm infection from the system using anti-virus software. The prototype was then placed in-line before the firewall. The client system was re-infected with the mass-mailing worm and the network traffic was observed. Our prototype detected the first MX query and blocked all SMTP traffic from the infected host. Audit logs from both the firewall and our prototype confirm that no subsequent SMTP traffic from the infected system passed through the in-line prototype.

5.3 Discussion of False Positives and Negatives

The following two sections discuss the impact and causes of false positives and negatives on our detection technique.

5.3.1 False Positives

In our analysis of MX query activity in Section 4.2, we discovered that over a one-week period only 5 unexplained MX queries were made by client systems. Although this suggests that no widely installed client application requires MX records to access external mail servers, these queries would still be a source of false positives. Using our current configuration, our prototype would have erroneously contained three systems during the observation period.

However, the prototype could be set to restrict access after the observation of more than a single MX query within a specified time window (see Section 6.1). For instance, during the same one-week observation period if we had set the prototype to contain a client system after the observation of two or more MX queries within a 20 minute time period then no false positives would have occurred. If any client requires the use of its own SMTP server, then two options exist. First, its IP address could be added to the whitelist to exempt it from detection and containment. Second, the prototype could be configured to allow clients a certain number of MX queries within a given time interval before quarantine occurs.

5.3.2 False Negatives

The prototype permits client MX record request activity within the enterprise network through the use of a whitelist. For instance, in the department network we observed in Section 4.2 the whitelist would consist of four entries (i.e. three authorized mail servers and the SpamAssassin test system). If a system on the whitelist becomes infected with a mass-mailing SMTP-engine, its malicious mailing activity will not be detected.

5.4 Limitations and Circumvention

Limitations. Instead of using SMTP-engines to send email, compromised systems could use the resident email client software. In this case, the corporate mail server would perform the MX queries on behalf of the system and this would not be detected by our technique. However, an attacker using this strategy would not benefit from the advantages of using an SMTP-engine as discussed in Appendix A.1. A second limitation with our technique is that it does not differentiate between spam zombies and mass-mailing worms.

Circumvention. If an infected system contained a list of IP addresses for the mail servers it wanted to send mail to, then no MX queries would be required. However, this would limit the infected system to sending mail to predetermined domains. While this may be a questionable strategy for opportunistic mass-mailing worms, it may be acceptable when sending spam where a recipient's list may be available. In this scenario, although the attacker would still need to perform MX queries, these could be made offline and thus undetectable by our technique.

Another way to avoid detection would be to have the infected systems contact a system outside of the enterprise network to perform DNS queries on their behalf. Communications to this *covert proxy DNS* server could be tunneled through the network using an arbitrary non-filtered port. Our implementation of the technique currently can not detect DNS activity tunneled through the arbitrary ports (i.e. ports other than port 53).

6 Comparison with Alternate Approaches to Address Malicious Mass -mailing

A number of techniques exist that attempt to prevent malicious mass-mailing. In this section, we discuss how our technique is related to three of these.

6.1 Port 25 Blocking

Many enterprises and ISPs have instituted port 25 blocking at the network boundary to stop malicious SMTP-engine mass-mailing activity. This prohibits users running SMTP services locally. Mail relaying to 3rd party mail servers is typically allowed only through the ISP's mail servers. Port 25 blocking and the subsequent mandating of mail relaying activity to specific mail servers has proved to be both an unpopular and questionable strategy. Specifically, indiscriminate port 25 blocking has disadvantages including:

1. *Detection of infected systems*: port blocking alone will not identify malicious mass-mailers.
2. *Impact to user privacy*: mandatory mail relaying through an ISP's mail servers closely ties the user's identity to the ISP's network.
3. *Impact to user choice*: mandating how mail should be forwarded breaks a fundamental design principle of the Internet (i.e. flexibility) and takes away user options.
4. *Circumvention*: mass-mailers have begun to evade port blocking by setting up servers to use arbitrary ports (i.e. other than port 25) to relay mail from mass-mailing zombies.

Wholesale port 25 blocking is a quixotic solution. While it does address the problem of stopping malicious SMTP-engine mass-mailing activity, it does so at the expense of denying all legitimate client-based SMTP activities. In contrast to port 25 blocking, our prototype can contain SMTP traffic from a client system after any number of MX query attempts within a configurable time interval.

During our testing (see Section 5.2) we configured our prototype to restrict SMTP activity once it detected the first MX query attempt from a client system. This rule effectively restricts MX queries to the exclusive domain of the network mail servers specified in the whitelist during initialization. This decision

was directly supported by our analysis of client MX query activity presented in Section 4.2. In other network environments it may be an appropriate strategy to contain a client system after observing some number of MX queries (e.g. greater than one) within an arbitrary time interval. For instance, this strategy would be recommended for networks where it is common practice to allow users to relay mail to 3rd party mail servers. In this case, the prototype could be set to contain a client system after observing MX record activity greater than that required to resolve the FQDNs of a finite number of 3rd party mail servers. Although our approach can be as restrictive as port 25 blocking (e.g. quarantine after a single MX query is observed), it offers a means to allow a configurable amount of client-based mailing.

6.2 Preventing Email Forgery

Sender Policy Framework (SPF) [18] is a proposal designed to reduce email address forgery. Domain owners publish SPF records that specify the systems authorized to send mail for that domain. The recipient of a mail transfer agent (MTA) performs a check to determine if the sending MTA is listed in the SPF record for the domain specified in the sender's email address. If the MTA is not listed in the SPF record, it is flagged as suspicious (e.g. potential spam) or simply rejected. One drawback to SPF is that it breaks simple mail forwarding which preserves the from address within the email header as it travels through different mail servers. In order to be verified by SPF, mail forwarding will have to abandoned in favor of some form of re-mailing (i.e. as mail travels through mail servers the sender field in the email header is changed to reflect the address of the last mail server hop).

An alternate proposal to address email address forgery is DomainKeys [9]. Unlike SPF, it is designed to verify both the domain of the sender and the contents of the message. The integrity of the message is ensured by the use of a digital signature over a SHA-1 hash of the mail message (selected header fields and body). The hash is signed by the sending mail server with its private key. To verify the message, the receiving mail server performs a DNS lookup for a record that contains the sending domain's public key. If the signature verifies, there is strong assurance that the email came from the claimed domain and it has not been modified in transit. A mail server receiving an unauthenticated mail message may flag it as suspicious or simply choose to reject it. Potential issues with this approach arise from the fact the entire email message (including header) is hashed. Mail message headers are often legitimately altered by mail list servers, spam filters, and even ordinary mail servers. Legitimate modification of the email header will cause the authentication check to fail. Additionally, this approach entails both computational and administrative overhead as a result of the required cryptographic services.

Both these techniques are designed to detect spoofed email as it is received at network mail servers. However, SMTP-engines bypass corporate mail servers completely. These techniques would be ineffective in preventing mass-mailing that originates from SMTP-engines.[6] The use of SPF and DomainKeys allows spoofed mail from SMTP-engines to be stopped at the receiving end only after it has been sent. In contrast, our approach will stop SMTP-engine mass-mailing at the sending end before it crosses the network boundary. Furthermore, mail that fails SPF or DomainKeys checks may not be rejected in favor of using the failure result as an input flag to a spam filter. It is common practice for spam filters to archive mail identified as spam in a *spam directory* for the user. At the user's convenience, they can view the spam folder to look for any legitimate mail misidentified as spam. In this case, any forged mail received (e.g. like those mailed from SMTP-engines) still consumes bandwidth, network server processing, disk space, and the user's time.

Our approach can be used in conjunction with these techniques to enable both the verification of incoming mail and the eradication of mass-mailing from SMTP-engines before mail is ever sent. Stopping

[6]Once widely deployed, one could argue that these techniques may reduce mass-mailing from SMTP-engines as the generated messages would fail these checks when received at the intended recipient's mail server making this a less effective delivery mechanism. However, until there is universal adoption of these technologies, SMTP-engine mass-mailing will probably persist.

mass-mailing from SMTP-engines significantly reduces the amount of malicious mail traveling through the Internet, mail server processing, and user frustration.

7 Concluding Remarks

This paper presents a technique to detect and contain malicious mass-mailing activity from compromised systems with SMTP-engines. Being anomaly-based it has the ability to detect emerging worms and mass-mailing activities (e.g. spam and phishing zombies).

A benefit over existing mass-mailing detection techniques is that it is email *content independent*. The detection technique being based on indirect network behaviors (i.e. MX record queries) allows us to detect mass-mailing activity from client systems regardless if it is stealthy (e.g. slow spreading), polymorphic, attachment-based, or spread by embedded URLs in text messages.

Although our approach can be as restrictive as port 25 blocking for all clients in the network (i.e. quarantine occurs after a single MX query), it is flexible enough to allow a number of MX queries to occur from a client system before it is quarantined. For instance, this may allow legitimate users to relay mail through 3rd party mail servers of their choice. We believe that, in general, wholesale port blocking should only be used as an effective *stop-gap* measure until a more considered countermeasure is developed. Draconian policies like port blocking typically do not address the root problem, in this case the identification and quarantining of only malicious mass-mailing systems. Additionally, our technique could be used in conjunction with proposals to address mail forgery (i.e. SPF and Domain Keys) to offer a more comprehensive strategy to stop malicious mass-mailing.

In review, our examination of network traces from a medium-sized university network suggests that MX query activity from client systems is a viable means to detect unauthorized mass-mailing activity. We tested our prototype in an isolated test network using a real mass-mailing worm. The technique allowed us to detect and contain mass-mailing activity before a single email progressed beyond the originating enterprise network.

Acknowledgements

We thank Peter Choynowski, Anil Somayaji, Carleton University's Digital Security Group, and the anonymous reviewers for comments which significantly improved this paper. The second author is Canada Research Chair in Network and Software Security, and is supported in part by an NSERC Discovery Grant, the Canada Research Chairs Program, and MITACS. The third author is supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) and MITACS (Mathematics of Information Technology and Complex Systems) grants.

References

[1] The Apache SpamAssassin Project. http://spam assassin.apache.org.

[2] The netfilter/iptables Project. http://www.net filter.org.

[3] The SPAMHAUS Project. *SPAMHAUS*. http://www.spamha us.org/index.lasso; accessed on March 28, 2005.

[4] W32/Dumaru.w Trojan. *McAfee Inc.* http://vil.nai.com/vil/ content/v100977.htm; accessed on March 28, 2005.

[5] W32.Netsky.Q@mm. http:// security response. symantec. com/ avcenter/ venc/ data/ w32.netsky.q@mm.html.

[6] Norvag/MyDoom Worm Spreads Quickly, Targets SCO. *Security News Article*, January 2004. http://enterprise security.symantec.com/ content.cfm?articleid=3277&EID=0; accessed on Mar 20, 2005.

[7] Symantec Internet Security Threat Report Trends for July 04-December 04. Technical Report Volume VII, March 2005.

[8] D. Crocker. Standard For The Format of Internet Text Messages. RFC, August 1982. http://www.ietf.org/rfcs/rfc0822.txt; accessed March 24, 2005.

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

[10] G. Goth. Phishing attacks rising, but dollar losses down. *IEEE Security and Privacy*, 3(1):8, January-February 2005.

[11] A. Gupta and R. Sekar. An approach for detecting self-propagating email using anomaly detection. In *Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection*, September 2003.

[12] R. Hu and A. Mok. Detecting unknown massive mailing viruses using proactive methods. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, September 2004.

[13] J. Ioannidis. Fighting spam by encapsulating policy in email addresses. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium*, Feb. 2003.

[14] R. Lemos. Worm double whammy still hitting hard. *NEWS.COM*, August 2003.

[15] E. Levy. The making of a spam zombie army: Dissecting the soBig worms. *IEEE Security and Privacy*, 1(4):58–59, July-August 2003.

[16] E. Levy. Interface illusions. *IEEE Security and Privacy*, 2(6):66–69, November-December 2004.

[17] J. Leyden. Zombie PCs spew out 80% of spam. *The Register*, June 2004.

[18] M. W. M Lentczner. Sender Policy Framework: Authorizing Use of Domains in MAIL FROM. http://spf.pobox.com/.

[19] L. McLaughlin. Bot software spreads, causes new worries. *IEEE Distributed Systems Online*, 5(6), June 2004.

[20] Y. Musashi, R. Matsuba, and K. Sugitani. Detection of mass mailing worm-infected IP address by analysis of DNS server syslog. In *IPSJ SIG Technical Reports, Distributed System Management 32nd*, number 37, pages 31–36, 2004.

[21] Y. Musashi, R. Matsuba, and K. Sugitani. Detection of mass mailing worm-infected PC terminals. In *Proceedings for the 3rd International Conference on Emerging Telecommunications Technologies and Applications (ICETA2004)*, pages 233–237, 2004.

[22] Y. Musashi, R. Matsuba, and K. Sugitani. Detection of mass mailing worm-infected PC terminals for learners by observing DNS query access. In *27th IPSJ SIG Technical Reports, Computer Security*, number 129, pages 39–44, 2004.

[23] K. Poulsen. Rise of the Spam Zombies. *SecurityFocus*, April 2003. http://http://www.security focus.com/printable/news/4217; accessed on March 28, 2005.

[24] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, and E. Lear. Address Allocation for Private Internets. RFC, February 1996. http://www.ietf.org/rfc/rfc1918.txt accessed on April 10, 2005.

[25] P. Roberts. SoBig: Spam, Virus, or Both? *PCWORLD Article*, June 2003. http://www.pcworld.com/ news/ article/ 111028 00.asp; accessed on Mar 20, 2005.

[26] S. Sidiroglou, J. Ioannidis, A. Keromytis, and S. Stolfo. An Email Worm Vaccine Architecture. In *Proceedings of the 1st Information Security Practice and Experience Conference (ISPEC)*, April 2005.

[27] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, August 2002.

[28] D. Whyte, E. Kranakis, and P. van Oorschot. DNS-based detection of scanning worms in an enterprise network. Technical report, Carleton University, School of Computer Science, TR-04-06, August 2004.

[29] D. Whyte, E. Kranakis, and P. van Oorschot. DNS-based detection of scanning worms in an enterprise network. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, Feb. 2005.

[30] C. Wong, S. Bielski, J. McCune, and C. Wang. A study of mass mailing worms. In *The 2nd ACM Workshop on Rapid Malcode*, Oct. 2004.

[31] C. Zou, W. Gong, and D. Towsley. Feedback to Email Worm Defense System for Enterprise Networks. Technical report, University of Massachusetts, Amherst, TR-04-CSE-05, April 2004.

[32] C. Zou, D. Towsley, and W. Gong. Email Worm Modeling and Defense. In *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN'04)*, October 2004.

A Background on the Use of Malicious SMTP-engines

In this section, we review how SMTP-engines are used by mass-mailing worms, spammers, and phishers.

A.1 Mass-Mailing Worms

The term *mass-mailing worm* describes the general class of worms that use the medium of email to propagate. Recent examples of mass-mailing worms include SoBig, Netsky, Bagle, and MyDoom.[7] Mass-mailing worms infect systems using infection vectors including: (a) malicious attachments, (b) text links to malicious websites, and (c) message viewing in an email application preview pane. An SMTP-engine, although as short as a few lines of code, is the most crucial component of the malicious software installation during infection. SMTP-engines allow mass-mailing worms to circumvent corporate mail servers and send infected emails directly from the victim system. Furthermore, there is no requirement for the worm to have the capability to detect and then use disparate email clients on victim systems. This ensures that emails can be

[7] All of these worms use SMTP-engines to propagate.

generated and sent regardless of the email client software used by the victim, thus increasing the worm's propagation rate.

Additionally, the use of an SMTP-engine obviates the need for the worm to interact with the victim system's email client resulting in fewer signs of an active infection. For example, no copies of the infected emails will appear in the sent items folder of the infected system's email client.

A.2 Spam

Spam can be defined as unsolicited bulk email [3]. A typical spam email is sent with little or no modification to thousands or millions of recipients that have no prior relationship with the sender. Recipient email addresses can be obtained from Internet bulletin board postings, commercial databases, or by simply guessing common names and domains. It has been estimated [3] that spam comprises 75% of all email traffic arriving at most ISP mail servers.

Although novel approaches have been developed to combat spam (e.g. [13]; see also Section 6), it continues to be a serious nuisance. Professional spamming is a lucrative business and thus attracts a motivated criminal element. A common practice for spammers to conceal their identity by setting up *throw-away* email accounts at ISPs using false names, addresses, and stolen credit cards. To conceal the origin of the message spammers spoof the sender's email address through the use of third-party systems. Historically, open mail relays have been used to conceal the origin of the spam emails. However, this method of spam delivery is decreasing due to the use of blacklists (see Section A.2.1) and increased user awareness. Recently, spammers have resorted to a new method of spam delivery through the use of networks comprised of infected home PCs as discussed in the next three sections.

A.2.1 Spam Propagation Through Open Mail Relays

A mail server can be configured to either blindly accept email from any user and source address (i.e. *open relay*) or accept email based only after certain criteria (e.g. sender IP matches a specific range of IP addresses or a domain) has been met (i.e. *closed relay*).

Spammers scan the Internet searching for mail servers that can be used for open relays. Open relay mail servers allow spammers to forward thousands of email messages with spoofed source addresses (see Figure 5). It is now best practice to configure a mail server as a closed mail relay. Mail servers that continue to allow open relay connections are added to *Real-time Blackhole Lists* (RBLs). Any email sent by a mail server on an RBL is typically rejected by other mail servers.

A.2.2 Spam Propagation Through Open Proxies and Spam Zombies

The use of RBLs has served to decrease the number of open relay mail servers, causing spammers to utilize other mail delivery mechanisms such as *spam zombies*. A spam zombie is a compromised system that is covertly controlled in order to relay spam. It has been estimated that 80% of spam is sent by spam zombies [17]. Zombies are collectively grouped into virtual networks known as bot networks (*botnets*). A typical botnet consists of 2,000 to 10,000 compromised systems [19]. Zombies contain a variety of malicious software packages that allow remote coordinated covert control. Home users are attractive targets for use as zombies as they are often improperly secured, poorly maintained, utilize broadband speeds, and are constantly connected. Common methods to enable a spam zombie to send spam are: use of open proxy server software, and built-in SMTP-engines.

An open proxy is a network service that allows for indirect access to other network services. A client (in this case the spammer) simply connects to a zombie running a network proxy and commands it to connect to a mail server. Once the mail server receives the connection, it believes that the connection is coming

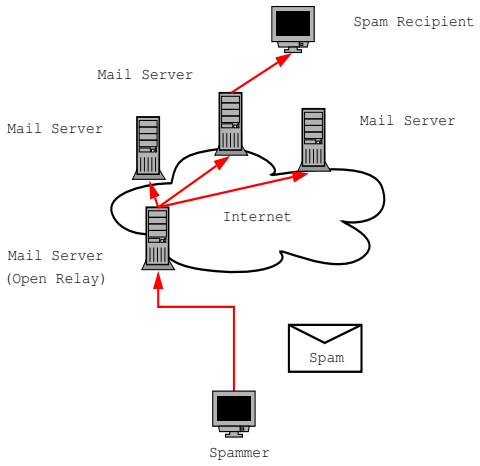


Figure 5: Spam being sent through an open relay.

from the zombie and not the client system. Although open proxy systems have many legitimate uses (e.g. allowing for anonymous web browsing, sharing network resources, etc.), spammers can use them to route mail to an ISP's mail relay server.

However, user accounts that send large quantities of spam email from their ISP accounts are usually quickly reported. In turn, the ISP will typically disable their account for breach of acceptable use policies. To avoid detection, spam zombies often make use of SMTP-engines to hide their activity. *Jeem* and *Proxy-Guzu* are two popular Trojans used by spam zombies with built-in SMTP-engines to send spam [23]. The use of SMTP-engines bypasses corporate mail servers allowing large quantities of mail to be sent avoiding any mail gateway defences (e.g. virus scanning software).

A.2.3 Convergence of Mass-Mailing Worms and Spam Botnets

The outbreak of the SoBig worm illustrates how mass-mailing worm attacks and spamming has converged [25, 15]. Analysis of its distribution patterns revealed that the initial infection resembled the spread of spam message rather than a virus. It has been theorized that the worm writer used a spam botnet to speed the initial spread of the worm variant rather than relying solely on SoBig infected systems. This method called *seeding* dramatically accelerates propagation by allowing the worm to quickly obtain a significant number of hosts, thus avoiding the typical initial slow spreading phase characteristics of non-seeded worms [27]. Verisign estimates that SoBig infected at least 100,000 systems at the height of its infection [14]. Once SoBig had completed its propagation, spammers used the *back-doors* installed by the worm to add these infected systems into their spam botnet. The zombies then used their SMTP-engines to send out advertisements rather than infected emails.

A.3 Phishing

Phishing is an activity that uses email to monetarily defraud recipients. Phishing attacks are an emerging security threat that is both difficult to defend against and increasing at an alarming rate [7, 10]. A typical phishing attack involves sending an email that appears to be from a legitimate company asking the recipient to update their account information. Any credit card information, passwords or personal information divulged in the responding email is used by criminals to defraud the sender. Phishing email is usually cleverly crafted with links to spoofed web pages of highly respected companies where this information can be requested with less suspicion [16].

Although a phisher's objective may be different than a spammer's (i.e. fraud versus advertisement delivery), a common goal is to send as much email as possible to reach more potential victims. Trojans have been designed specifically to install an SMTP-engine on systems to send phishing email [4].