Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Mass-Mailing Worms: Prevention, Detection and Response (A Case Study)

Preventing mass-mailing worms from infecting the PCs in your network is obviously the cornerstone of any reasonable defense against them, but early detection and prompt isolation and recovery of any infections which do occur should be your second line of defense. In this paper I describe the approaches to mass-mailing worm prevention, detection, and incident response that I have developed and used on a large university network. The prevention strategy has encompassed user education and awareness, desktop anti-virus pol...

Copyright SANS Institute
Author Retains Full Rights

AD

Mass-Mailing Worms: Prevention, Detection and Response (A Case Study)
Richard Gadsden
GSEC Practical v1.4b

Abstract

Preventing mass-mailing worms from infecting the PCs in your network is obviously the cornerstone of any reasonable defense against them, but early detection and prompt isolation and recovery of any infections which do occur should be your second line of defense. In this paper I describe the approaches to mass-mailing worm prevention, detection, and incident response that I have developed and used on a large university network. The prevention strategy has encompassed user education and awareness, desktop anti-virus policy, and minimally invasive server-based filtering of incoming email, while the approach to worm detection is based on detecting traffic patterns of worm behavior on the network itself, using readily available open source tools, including the argus real time flow monitor and the Perl scripting language. In this paper I present results which demonstrate the efficacy of our strategies for prevention, behavior-based (as opposed to signature-based) detection, and recovery, and I discuss future directions based on lessons learned to date.

A Brief History of Mass-Mailing Worms

I use the term *mass-mailing worm* to describe the general class of viruses, worms and other malware which use the medium of email to propagate. At minimum, infections by members of this class of malware are a significant nuisance, especially when they generate high volumes of traffic that clog networks and overload mail relays. At worst, these worms can disclose confidential information from the systems they infect, with some mass-mailing worms going so far as to install keystroke loggers and backdoor services for collecting passwords and other sensitive information from the infected system. Although they all use email to propagate, some of these worms can also spread through other means, such as by scanning for unprotected Windows shares on the infected computer's local network.

The overwhelming majority of mass-mailing worms target Windows systems. Most worms rely on an unsuspecting user to open an attachment containing the executable (and infective) payload, although some worms have also exploited defects in widely used user agents, especially Microsoft Outlook, to execute on the target system without the user even needing to open any infective attachments. Malware authors have been so successful at exploiting these two vulnerabilities - gullible email users and defective email user agent software - that the risk from email-borne malware is now considered the most significant malicious code risk by most organizations. Because of the risks that mass-

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

mailing worms create, many network administrators must invest heavily in virus scanning software on their mail servers to help prevent the spread of email-borne malware, while users must suffer with the delays in delivery these scans can cause, and even with having legitimate attachments blocked as false positives.

The Melissa virus [14], released into the wild in March 1999, was the first mass-mailing worm of global consequence. Network administrators were not prepared to deal with this new kind of malware threat, and the virus's rapid infection of millions of PC's caused widespread network disruption and economic damage. Later mass-mailing worms such as SirCam [13] and Klez [12] have demonstrated increasingly more sophisticated and polymorphic behavior, and have proved capable of infecting new victims in large numbers, long after their initial appearance in the wild. [15]

Necessity, the Mother of Invention

Our University (ouru.edu) is a large research university, where for the past few years I have served as the overall coordinator of IT security. The early mass-mailing worms such as Melissa and LoveLetter [9] did not cause any serious problems for Our University. The behavior of these early worms was so simple and predictable that we were able to protect our users' PCs from the possibility of infection by using simple filters on our SMTP servers. Moreover, the sensational media coverage these early worms received helped to raise end-user awareness of the existence of mass-mailing worms. By the time later worms such as Hybris [11] began arriving in their inboxes in large numbers, we assumed that Our University's users were well aware of the dangers posed by potentially malicious executable attachments, and could avoid being victimized. We were soon to be disabused of this notion.

In March 2001 the abuse@ouru.edu account suddenly began receiving complaints from numerous people, both inside and outside Our University, who claimed to be receiving copies of the Hybris in email messages sent from computers on our network. At the same time, the postmaster@ouru.edu address began receiving an alarming number of "bounces" containing copies of Hybris which had been sent from computers on our network (to addresses which turned out to be invalid, resulting in their being bounced to postmaster). As we began analyzing the message headers in the complaints and in the bounces, it became clear that a substantial number of our students' home computers, used to dial into Our University's PPP service, had become infected with Hybris in a matter of days. Later we learned that a few days earlier, the virus had been sent from one student's infected computer to an unmoderated mailing list which reached every student in his college. Within two days, dozens of students' computers in that college had been infected.

Needless to say, we were alarmed by this rapid and widespread infestation of our

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

network by the Hybris worm, which had until then been just a minor nuisance to our users. Based on what was known about Hybris, it did not appear to pose any overly serious, direct threats to our network. After all, it would not format the hard drives on our users' PCs, nor did it appear likely to compromise any confidential information. However, the fact that a significant number of Hybris tranmissions were being traced back to our network was causing embarrassment for Our University, and this would clearly damage Our University's reputation if we failed to respond. I was also very concerned to realize that many of our students had been so oblivious to, or naive about, the dangers of mass-mailing worms that they had infected themselves with something as obviously suspicious as an incoming Hybris-generated message should have been. It was also clear that many of our students were not using anti-virus software. I concluded that the overall level of risk to which Our University was exposed, because of the threats posed by mass-mailing worms in general (not just Hybris), was much higher than we had previously believed it to be.

Clearly, Our University needed to contain the damage from this outbreak as quickly as possible, and figure out how best to prevent similar problems in the future. We proceeded on three fronts: detection of existing infections, isolation and recovery of the infected systems, and prevention of additional infections.

Prevention

Our University's two most significant vulnerabilities to the threats posed by mass-mailing worms were obvious: (a) lack of end-user awareness, and (b) failure to use anti-virus software. If all of our students had current anti-virus protection installed and properly configured on their computers, and if all of our students understood the risk of opening potential malicious attachments, then the risk of another outbreak of a mass-mailing virus would be much lower.

Our University had already, for several years, made desktop anti-virus software available to all of our students under a site license, and had widely advertised its availability. Yet we clearly needed to do more.

To promote more consistent use of desktop anti-virus protection, I sought approval for a change to Our University's Computer Use Policy, to *require* for the first time that appropriate anti-virus protection be maintained on every computer used on Our University's network. This policy change, with the support of Student Government and the Faculty Senate, was approved by Our University's Board of Trustees in October 2001.

We also undertook an education and awareness campaign, to help our users understand both the nature of mass-mailing worms, and the broader dangers inherent in email attachments which can contain malicious code. Our goal has been to help our users understand that they have an important role to play in

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

defending our network from these threats. Because Our University's desktop anti-virus software relies on signatures, there is always a window of vulnerability between the time that a new virus or worm is released into the wild, and the time that updated signatures are developed by our anti-virus vendor and can be distributed to all of the vulnerable desktops on our network. Because these windows of vulnerability occur all the time, we depend on our users to play it safe with attachments. Our message to our users is: If an email message carrying attachments looks suspicious, don't touch it with the proverbial ten-foot pole, even if your desktop anti-virus software doesn't recognize it as malicious.

After having begun to address our two most significant vulnerabilities, we proceeded with an assessment of our residual risks. One area of concern was the low likelihood of 100% compliance with our new desktop anti-virus policy, given the highly distributed and sometime haphazard desktop computer administration model followed at Our University. Another area of concern was the difficulty of keeping all of our users educated about, and alert to, the threats posed by mass-mailing worms, given our constant influx of new students, postdocs, fellows, and other visiting scholars and research associates.

I concluded that the residual risks were sufficiently high to invest in one or more additional preventative controls which would be complementary to our desktop- and user-centered controls. Initially, we proposed implementing signature-based virus scanning and attachment blocking on our SMTP and/or IMAP servers. However, our faculty objected to this approach on the basis of its perceived intrusiveness and potential inconvenience. The costs for implementation and on-going support would also be fairly high. As a result, that proposal was shelved, and instead we implemented a much simpler control on our IMAP servers: a simple message filter called RenAttach.

RenAttach (Rename Attachments) is, according to its author Jem Berkes, "a small, efficient and surprisingly effective filter designed primarily to offer an additional level of safety to Windows users whose e-mails pass through a UNIX-like mail server." [1] Designed to filter some or all mail on an SMTP or IMAP server, RenAttach works by renaming any attachments which the implementing site considers dangerous, based on their filename extension. It also sets the MIME types of any renamed attachments to "application/unknown" to prevent their being executed by email user agents which take their dispositional cues from the attachment's MIME type (as explicitly set in the MIME header, rather than inferred from the filename extension given in the MIME header).

Sites implementing RenAttach can select one of three modes of operation: *full* mode, which renames all attachments; *goodlist* mode, which renames all attachments except those on a configurable list of extensions considered safe ("good"); or *badlist* mode, which only renames attachments if they have extensions on a configurable list which are considered dangerous ("bad"). Choosing the optimum mode for a site involves, not surprisingly, a compromise

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

between site security and user convenience. At Our University, I knew that full mode would be unacceptable, but I hoped that we could use goodlist mode, as it is inherently safer than badlist mode. To help Our University make an informed decision, I wrote a Perl script which collected statistics on the extensions of all MIME attachments stored on our IMAP servers over the course of a week. From this script, we learned that over 10,000 attachments with no filename extension were delivered during the week sampled, which constituted more than 10% of the total attachments delivered that week. When we realized that these attachments with no filename extensions were legitimate files being exchanged by Macintosh users, we elected to implement RenAttach in badlist mode. Even though this is inherently a little less secure than goodlist mode, forcing our users to put up with having 10,000 legitimate attachments renamed per week would have been too high a price to pay for the additional security. This is the current RenAttach configuration file we use at Our University:

```
# In badlist mode, only the following extensions are renamed
bad: ADE,ADP,BAS,BAT,CHM,CMD,COM,CPL,CRT,EXE,HLP,HTA,INF
bad: INS,ISP,JS,JSE,LNK,MDB,MDE,MSC,MSI,MSP,MST,OCX,PCD,PI,PIF
bad: REG,SCR,SCT,SHS,URL,VB,VBE,VBS,WSC,WSF,WSH
```

The results of our prevention efforts - a combination of desktop anti-virus policy, user education, and renaming of attachments with extensions that we consider dangerous - have been encouraging. The number of documented (detected) infections by mass-mailing worms on our network has steadily declined, despite the increasing incidence of mass-mailing worms on the Internet as a whole.

Despite all of our prevention efforts, we do still suffer from some new infections on our network. In most cases, these infections have been introduced when new or visiting users access external email accounts from a Windows workstation on our network, which either does not have anti-virus software installed, or has not been updated with the latest signatures. Given the nature of the research university environment, with a nearly constant influx of many different types of new users, many of whom bring their own computers with them, and many of whom use external email systems, the cost of achieving 100% prevention would be prohibitive. We recognize that 100% prevention is not a realistic goal, and that detection and incident response will always be important in our environment.

Detection: Behavioral Approach

Detection of mass-mailing worms and other malware at the desktop is practically a no-brainer. If desktop anti-virus software is installed and properly configured, and is updated with a signature specific to a given incoming email-borne virus, it will detect the virus and stop it in its tracks, before it can infect the workstation. User education and awareness, and using email user agent software which does not suffer from any defects which allow malware to execute without cooperation from the user, are also factors at the desktop.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

But we know that (a) not every workstation on our network is in compliance with our anti-virus software policy, (b) not every user plays it safe with attachments, and (c) not everyone on our network uses email software which is free of defects. We know that we cannot block all incoming malware before it succeeds in infecting some desktops, and so we need to be able to quickly and consistently detect the malware infections which do occur on our network, and the best way we've found for doing this is by learning to recognize evidence of worm *behavior* on our network.

By definition, the one behavior shared by all mass-mailing worms is that they try to infect other systems via email. If we have a system on our network that is infected with a mass-mailing worm, we rely principally on the SMTP traffic that the infected machine generates to reveal the worm's presence. Obviously, this requires that we be able to distinguish "abnormal" worm-generated SMTP traffic from the "normal" user-generated SMTP traffic on our network. It also requires that we monitor *all* of the SMTP traffic at strategic points on our network, both the abnormal traffic and the normal traffic, at some level of detail.

The level of email monitoring, or the degree of scrutiny to which SMTP traffic is subjected, raises obvious privacy concerns. Our University's Computer Use Policy, in its section on Privacy and Confidentiality, contains the following statements about monitoring:

> The University reserves the right to monitor user activities on all University computer systems, and to monitor communications utilizing the University network, to ensure compliance with University policy, and with federal, state and local law. Monitoring shall be performed only by individuals who are specifically authorized, and only the minimum data necessary to meet institutional requirements shall be collected.

The "minimum data necessary" standard is one which we take particularly seriously. The network monitoring infrastructure and procedures which I have developed at Our University, and which we use for detecting mass-mailing worms, were designed to meet or exceed this standard, which is clearly intended to protect the privacy of our users.

The least intrusive type of monitoring is analysis of traffic patterns at the IP and the TCP layers. The information at these layers is, relatively speaking, both anonymous and free of any user-generated data. My procedure for worm detection always begins by looking strictly at the TCP/IP layers for patterns of abnormal, potentially worm-generated behavior. At Our University, a set of argus monitors are the primary source of the data in which we look for these abnormal traffic patterns.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Argus is an open-source real time flow monitor developed and maintained by Carter Bullard at QoSient, LLC. [2] In a typical argus implementation, a network tap is used to split off a copy of the traffic on a network link whose traffic needs to be monitored. An argus(8) data collection daemon, running on a UNIX or Linux system, listens in promiscuous mode on a network interface connected to one leg of the tap.

Although argus, like tcpdump, can be used to capture and store the application layer content of the traffic stream being monitored, typically argus is used to collect and record data at the *flow* level, which is a summarization of all the packets within a logically related sequence of packets. With TCP for example, a flow consists of the set of all packets exchanged between the source and destination sockets, from the first SYN packet in the opening handshake to (normally) the final ACK in the connection termination sequence. When dealing with TCP flows, argus will monitor the entire TCP "conversation" and write out a single flow record which documents the defining features of the flow (e.g. source and destination IP addresses and TCP ports), and summary statistics about the flow (e.g. the total number of packets and bytes sent in each direction). Argus uses the concept of network flows as defined in RFC 2724 [7], and is similar in basic operation to the NetFlow feature available on some Cisco IOS platforms [4].

At Our University, we have several argus monitors connected to network taps installed at strategic points on the borders between our network and external networks, including the Internet. An argus(8) daemon creates flow records for all traffic which traverses the monitored network links, 24 hours a day, 7 days a week. We use the flow records created by these argus monitors for a wide variety of border traffic measurement, analysis and auditing applications.

Since I began battling mass-mailing worms on Our University's network in March 2001, I've learned how to use the flow metrics generated by argus to identify the traffic flows which are likely to have been generated by mass-mailing worms. Because many worms attempt, at least some of the time, to send mail directly to their intended victims' SMTP servers, network borders are good places to watch for worm-generated traffic flows. Our principal technique for identifying suspected worm-generated SMTP traffic flows makes use of the rasort(1) command to report on the data collected by the argus(8) daemon which monitors the traffic traversing our network's Internet border. To this end, a shell script runs periodically from cron with the following command, where "118.28.0.0" is the (obfuscated) netblock for Our University's internal network:

```
rasort  -n -c -s srcaddr -s startime -r $argusfiles - \
          tcp and dst port 25 and \
          src net 118.28.0.0/16
```

This rasort(1) command extracts all SMTP (tcp and dst port 25) records from the

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

argus traffic audit files covering the period of interest, and sorts them by source IP address (-s srcaddr) and time (-s startime). The results are written to an output file, which then serves as the input to an evolving collection of Perl scripts I've written, that run at regular intervals. These scripts perform our first-order traffic analysis, looking for any patterns among the flows from a given source address that are indicative of a worm infection on the computer at that source address.

Although the behavioral repertoire of mass-mailing worms has gotten richer over time, each individual species of worm has only a finite number of ways that it can vary its behavior. We have found that the worms in widespread circulation can be positively identified simply by looking at the characteristic patterns of behavior they exhibit, which are readily apparent in the flow data which argus collects.

As perhaps the simplest example, SirCam [13] is a mass-mailing worm which is easy to identify, based solely on the set of SMTP relays with which it attempts to establish connections. This virus, first discovered in the wild in July 2001, is still in wide circulation on the Internet, ranking high on the MessageLabs "Top Ten" list [8] as of June 2003. One of the ways the virus attempts to propagate is by contacting one of a short list of hard-coded SMTP relays. The domain names of these relays are documented in CERT Advisory CA-2001-22 [5], and the IP addresses (at least at the time that SirCam was written) corresponding to these domain names are listed below:

```
dobleclick.com.mx          148.243.55.3
enlace.net                 200.38.242.1
prodigy.net.mx             148.235.168.60
compuline.com.mx           209.221.219.137
```

If any machine on our network attempts to open an SMTP connection to any of these specific IP addresses, we conclude that the machine is infected with SirCam. One of my Perl scripts, which regularly scans the argus flow records collected at our Internet border, notifies our network administrators of any such connection attempts.

Of course, most worms are not as easy to identify as directly as SirCam, by using traffic analysis at the TCP/IP layer alone. With most worms, I have learned that once we have identified a suspect SMTP flow record, or set of flow records, on the basis of traffic analysis at the TCP/IP layer, we also need access to at least some data from the application layer before we can positively identify the worm.

In the case of surprisingly many worm species, looking at just the envelope sender data field within a suspicious SMTP flow is sufficient for positive identification of the specific worm which generated the flow. With some worm species, the envelope recipient data field and/or the Subject header also provide useful clues to the worm's identity. Because these three types of data are sent by worms very early in the SMTP transaction, it turns out that by inspecting just the

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

first 256 bytes or so of the suspect flow's application data, we can reliably identify most worms.

To handle this common situation, we run a second argus daemon at our network border, which captures the first 256 bytes of every SMTP flow which originates inside our network. This special argus(8) daemon runs with the following directives in its configuration file [3]:

```
ARGUS_CAPTURE_DATA_LEN=256
ARGUS_FILTER="tcp and dst port 25 and src net 118.28.0.0/16"
```

Normally, for privacy reasons, the application layer data captured by this second argus(8) daemon is retained only briefly, before being purged without ever being seen by human eyes. However, whenever routine traffic analysis at the TCP/IP layer has established sufficient justification, I use the ra(1) program to read the application data collected by this daemon, and display the first 256 bytes of application layer data which was transmitted within the TCP/IP flow.

Klez [12], in all of its variant forms, is a good example of a worm which can be reliably identified by looking at just the envelope sender data field inside a suspect SMTP flow. Klez selects addresses at random from the infected machine's local address book, and uses them to forge sender addresses. However, every variant of Klez seems to share an idiosyncratic feature: the forged envelope sender address always contains a trailing blank. Thus, whenever we see a series of different SMTP envelope sender addresses being used in SMTP flows originated on the same Windows workstation, and every envelope sender address contains a trailing blank, we know that workstation is infected with Klez.

The following example illustrates this approach. As usual, one of my Perl scripts which performs routine traffic analysis at the TCP/IP layer detected a suspicious pattern of SMTP flows from a particular dial-up workstation:

```
26 May 03 18:10:14    tcp   118.28.10.131.1029   ->
37.8.10.166.25     8        9         519          788         FIN

26 May 03 18:11:01    tcp   118.28.10.131.1044   ->
86.46.170.11.25    4        0         248            0         TIM

26 May 03 18:11:47    tcp   118.28.10.131.1046   ->
73.84.15.189.25    4        0         248            0         TIM
```

For each flow record above, represented by a row within the sample output, are printed a timestamp, the protocol type (tcp), the source IP address and port, the destination IP address and port, the source packet count, the destination packet count, the total source bytes, the total destination bytes, and the state in which the flow ended (with TIM indicating a timed-out flow, and FIN indicating that the flow ended with the normal tcp connection tear down).

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

In this instance, my Perl script flagged a series of 3 attempts, within the space of a minute or two, by a computer at a specific IP address on our network, to open connections on tcp port 25 to a sequence of seemingly unrelated external SMTP servers. When I used the ra(1) program to display the application data for the first of these flows, the trailing blank in the forged SMTP envelope sender - a very distinctive behavioral quirk of the Klez virus - was immediately apparent:

```
HELO Ive..
MAIL FROM: <kevi@kam.umd.edu >..
RCPT TO:<pekomoola@aol.com>..
QUIT..
```

This simple, two-step technique, starting with simple traffic analysis at the TCP/IP layer, if necessary followed by selective examination of the first few hundred bytes of data at the SMTP (application) layer, has proved to work well enough to reliably detect and identify, with a high level of confidence, all of the mass-mailing Windows worms which have managed to infect one or more workstations on our network, including Hybris, SirCam, Klez, BugBear, Yaha, Sobig, Sobig-B, BadTrans, and most recently, BugBear-B. Using this simple technique, we are now able not only to detect new infections promptly, but also to let the administrator of the infected system know which specific worm we are dealing with, and provide him with recovery instructions specific to that worm.

Detection: Worms Using Relays

Although most mass-mailing worms include their own SMTP protocol engine, and will attempt to connect directly to distant SMTP relays, many of them will also use a local SMTP relay if they can find one. For example, SirCam will try to use "mail.<localdomain>" if it is defined in DNS, and Klez tries "smtp.<localdomain>" as a local relay. Therefore, in addition to monitoring our border traffic for suspicious SMTP traffic patterns patterns from Windows workstations, it is equally important for us to monitor the SMTP server logs on any servers on our network which Windows workstations can use as email relays, and look for suspicious traffic patterns there as well.

On our network, all of the servers which might act as SMTP relays are UNIX or Linux servers which run sendmail. I've written Perl scripts which regularly scan these servers' sendmail logs for suspicious traffic patterns. A sample of output from one particular script we use for this kind of traffic analysis is shown below:

```
Apr 25 08:45:11,<mccottt@ouru.edu >,230326,1
Apr 25 08:45:11,<mosesdm@ouru.edu >,230276,1
Apr 25 08:45:11,<faulknel@ouru.edu >,230381,1
Apr 25 08:45:11,<kicklirc@ouru.edu >,230430,1
Apr 25 08:45:11,<chambeyg@ouru.edu >,127643,1
```

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

```
Apr 25 08:45:12,<ellisjo@ouru.edu >,127596,1
```

Each row in the output summarizes a sendmail log record, generated by a message relayed through the server from a particular workstation, on a particular day. The output shows the log timestamp, the envelope sender address, the message size in bytes, and the number of recipients. The timestamps and message sizes clearly suggest some kind of malware in action, while the trailing blank in every (forged) envelope sender address reveals the type of malware at work here: this workstation is infected with some variant of Klez.

Detection: Other Worm Behavior (Defense in Depth)

It should be noted that in addition to sending mail, some worms engage in other types of network behavior which can be detected. For example, the BugBear virus installs "backdoor" service on tcp port 36794. A simple nmap scan against this port can identify Windows machines which may be infected.

Bugbear-B [10], a virus which installs a "backdoor" service on tcp port 1080, is an example of a worm which I've detected on our network using this network port scanning technique. The first step is to run nmap as follows:

```
nmap -p 1080 -m- -iL ips.txt
```

The file "ips.txt" contains a list of all the IP addresses to scan. If a tcp service is detected on port 1080 at any of these addresses, I then use amap [6] to determine whether it's really Bugbear-B which is listening on that port, or some other application:

```
amap -d <ip> 1080
```

An observed characteristic of the service that Bugbear-B installs on port 1080 is that it responds to an amap probe with a response packet of about 300 bytes, with the response being mostly unintelligible when interpreted as ASCII, but consistently ending in a string consisting of (hex) "73 6e 3a [2d]" followed by the ASCII encoding for a string of 10 variable digits. By running regular nmap scans against our network, and then running amap against any IP addresses with unexplained services running on tcp port 1080, I've been able to detect two BugBear-B infections on our network *before* the worm began attempting to propagate itself from the infected systems.

Incident Response: Isolation and Recovery

When we detect a workstation on our network which is infected with a mass-mailing worm, we have a defined incident response policy and procedure that we

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

consistently follow. If the workstation is on campus and connected to an Ethernet switch, our network administrators are notified, and they partition the switch port from the campus network. I then notify the administrator of the workstation of the incident, and provide him with specific instructions for recovering from the infection. We require him to bring the workstation into compliance with Our University's anti-virus policy, and if applicable, provide the workstation's user(s) with remedial education on playing it safe with email attachments, before we will re-enable the workstation's switch port. For remote dial-up users, I follow a similar procedure, disabling the user's access to Our University's PPP service in order to isolate the infected comptuer during the recovery period.

Under Our University's Computer Use Policy, we have the option of pursuing disciplinary action if there is a second violation. Fortunately we have never needed to exercise this option.

Lessons Learned, Future Directions

The results we have achieved, through our current strategies for prevention, detection, and recovery, have been impressive. The infection rate on our network has declined steadily, and we detect, identify and respond to new infections so promptly that it has been almost 2 years since we last received a complaint about a mass-mailing worm infection on our network that we had not already identified and isolated.

As long as we know what "normal" SMTP traffic on our network looks like, then it should always be possible to use traffic analysis to identify sets of "abnormal" flows which are likely to have been generated by mass-mailing worms. However, we recognize that as worm behavior becomes more polymorphic, it will become harder for us to identify which specific worm species is responsible for a given set of "abnormal" flows.

As the threats posed by mass-mailing worms continue to evolve, we may at some point elect to augment our simple (but still highly effective) RenAttach filtering of incoming mail with a signature-based virus scan of all incoming SMTP flows. Similarly, we may also choose to subject all outgoing SMTP flows to some kind of signature-based application layer content scanning, in order to maintain our current ability to identify the specific worm species responsible for the internal infections that we detect using traffic analysis.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

References

[1] Berkes, J. "RenAttach for UNIX". PC-Tools.Net. 04 Nov 2002. URL: http://www.pc-tools.net/unix/renattach/ (17 Jun 2003)

[2] Bullard, C. "Argus - Home." 01 Jan 2003. URL: http://www.qosient.com/argus/index.htm (17 Jun 2003)

[3] Bullard, C. "Argus - Manuals." Argus Documentation. 2001. URL: http://www.qosient.com/argus/manuals.htm (17 Jun 2003)

[4] Cisco Systems Inc. "Cisco IOS NetFlow". 2002. URL: http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml (17 Jun 2003)

[5] Danyliw, R. et al. "CERT Advisory CA-2001-22 W32/Sircam Malicious Code". 25 Jul 2001. URL: http://www.cert.org/advisories/CA-2001-22.html (17 Jun 2003)

[6] DJ.Rev.Moon, vanHauser. "THC-Amap". THC Releases. 16 Jun 2003. URL: http://www.thc.org/releases.php (17 Jun 2003)

[7] Handelman, S. et al, "RTFM: New Attributes for Traffic Flow Measurement." Request for Comments: 2724. October 1999. URL: http://www.rfc-editor.org/rfc/rfc2724.txt (17 Jun 2003)

[8] MessageLabs Inc. "MessageLabs Top 10 Current Virus Threats". 2003. URL: http://www.messagelabs.com/viruseye/threats/ (17 Jun 2003)

[9] Network Associates Inc. "VBS/Loveletter@MM." Virus Information Library. 04 May 2000. URL: http://vil.nai.com/vil/content/v_98617.htm (17 Jun 2003)

[10] Network Associates Inc. "W32/Bugbear.b@MM." Virus Information Library. 04 Jun 2003. URL: http://vil.nai.com/vil/content/v_100358.htm (17 Jun 2003)

[11] Network Associates Inc. "W32/Hybris.gen@MM." Virus Information Library. 01 Nov 2000. URL: http://vil.nai.com/vil/content/v_98873.htm (17 Jun 2003)

[12] Network Associates Inc. "W32/Klez.h@MM." Virus Information Library. 17 April 2002. URL: http://vil.nai.com/vil/content/v_99455.htm (17 Jun 2003)

[13] Network Associates Inc. "W32/SirCam@MM." Virus Information Library. 17 Jul 2001. URL: http://vil.nai.com/vil/content/v_99141.htm (17 Jun 2003)

[14] Network Associates Inc. "W97M/Melissa@MM." Virus Information Library. 26 March 1999. URL: http://vil.nai.com/vil/content/v_10132.htm (17 Jun 2003)

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

[15] The WildList Organization International. "WildList". 2003. URL: http://www.wildlist.org/WildList/ (17 Jun 2003)

[16] Van Epp, Peter. "Pssst, Wanna Buy Some Network Insurance?" ;login: The Magazine of USENIX and SAGE Volume 26, November 2001. URL: http://www.usenix.org/publications/login/2001-11/pdfs/epp.pdf (17 Jun 2003)

© SANS Institute 2003, Author retains full rights

Last Updated: September 2nd, 2015

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Crystal City 2015	Crystal City, VAUS	Sep 08, 2015 - Sep 13, 2015	Live Event
SANS Network Security 2015	Las Vegas, NVUS	Sep 12, 2015 - Sep 21, 2015	Live Event
SANS Seoul 2015	Seoul, KR	Sep 14, 2015 - Sep 19, 2015	Live Event
Data Breach Investigation Summit & Training	Dallas, TXUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Baltimore 2015	Baltimore, MDUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Perth 2015	Perth, AU	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Tallinn 2015	Tallinn, EE	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS ICS Amsterdam 2015	Amsterdam, NL	Sep 22, 2015 - Sep 28, 2015	Live Event
SANS Bangalore 2015	Bangalore, IN	Sep 28, 2015 - Oct 17, 2015	Live Event
SANS DFIR Prague 2015	Prague, CZ	Oct 05, 2015 - Oct 17, 2015	Live Event
SANS Seattle 2015	Seattle, WAUS	Oct 05, 2015 - Oct 10, 2015	Live Event
SANS Tysons Corner 2015	Tysons Corner, VAUS	Oct 12, 2015 - Oct 17, 2015	Live Event
SOS: SANS October Singapore 2015	Singapore, SG	Oct 12, 2015 - Oct 24, 2015	Live Event
SANS Gulf Region 2015	Dubai, AE	Oct 17, 2015 - Oct 29, 2015	Live Event
SANS Cyber Defense San Diego 2015	San Diego, CAUS	Oct 19, 2015 - Oct 24, 2015	Live Event
SANS Tokyo Autumn 2015	Tokyo, JP	Oct 19, 2015 - Oct 31, 2015	Live Event
SANS South Florida 2015	Fort Lauderdale, FLUS	Nov 09, 2015 - Nov 14, 2015	Live Event
SANS Sydney 2015	Sydney, AU	Nov 09, 2015 - Nov 21, 2015	Live Event
SANS London 2015	London, GB	Nov 14, 2015 - Nov 23, 2015	Live Event
Pen Test Hackfest Summit & Training	Alexandria, VAUS	Nov 16, 2015 - Nov 23, 2015	Live Event
SANS Hyderabad 2015	Hyderabad, IN	Nov 24, 2015 - Dec 04, 2015	Live Event
SANS Cape Town 2015	Cape Town, ZA	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS San Francisco 2015	San Francisco, CAUS	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS Milan 2015	OnlineIT	Sep 07, 2015 - Sep 12, 2015	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced