

Responding to Security Incidents on a Large Academic Network:

A Case Study
May 2003 – October 2005

Jamie Riden
Email: jamesr@europe.com

Abstract

This paper describes a series of security incidents on a large academic network, and the gradual evolution of measures to deal with emerging threats. I describe various techniques used and give an honest evaluation of them as implemented on a real network with tens of thousands of active users. Thanks to the relatively open nature of academic computing environments, the reader may notice that significant emphasis is given to detection and response capabilities; obviously, preventative measures are preferable when this is possible. I hope this information will be valuable when system administrators and IT security managers are evaluating preventative measures to deploy, and when they are responding to ongoing incidents.

Background

The data for this presentation was collected on a live academic network between May 2003 and October 2005. The network number is a 16-bit prefix, i.e. There are potentially around 65,000 publicly routable IP addresses, though only around 6000 are in use at any one time. The network devices are made up of Windows, Mac OS, UNIX and Linux computers, together with Cisco switches.

There is little restriction on internal communications, but between the LAN and the Internet there is a Cisco firewall, with a default deny policy for inbound traffic and a default permit policy for outbound traffic. Outbound traffic which is blocked is 25/tcp (SMTP), 80/tcp (HTTP) and 53/tcp and udp (DNS). The rationale for these blocks is as follows:

- Email (SMTP) is blocked because the institution wished to present a uniform front as far as email addresses were concerned; though this also provided a handy choke point for performing virus scanning and for prevent compromised internal systems being used for sending Unsolicited Bulk Email aka spam. All email has to be sent via the central mail server.
- Web traffic (HTTP) is restricted for several reasons. It was felt necessary to have an audit trail of web activity should the need arise to examine it, and because unchecked, web traffic could easily have saturated the Internet connection and disrupted other communications. By requiring web traffic to go through one of several caches, we save bandwidth on commonly fetched pages, ensure all requests are logged together with a user name and make it easier to throttle the flow of data if necessary.
- DNS: This was introduced after a particular trojan changed which DNS servers a handful of computers were querying, similar to the incident described at [ISC05]. Obviously, our internal DNS servers were still allowed to forward and recursively resolve queries. This is designed to protect users who will be unaware that their computer is resolving names via a malicious DNS server.

The main problems which were apparent in 2003 were: lack of mechanisms for patching desktop machines, generally weak staff passwords, limited ability to audit passwords or patch levels, insufficient staff time to maintain systems and lack of much visibility into traffic flows and network events.

Between incidents one and two, a network-based Intrusion Detection System was installed. Initially this consisted of one snort sensor which logged alerts to text files. Later on a database was created to accept the logged alerts, three more sensors were added, and the ACID[ACD] and then BASE[BAS] applications were used to sort through the alerts. The “bleeding-edge” snort signatures[BLD] were also added to the configuration.

Incident #1 – a compromised desktop machine

This machine had been a server and the firewall had been configured so the machine was effectively directly connected to the Internet. This access had been put in 'temporarily' as a third party needed access to the machine, but the access was not needed at the time of the incident. Two contributing factors to this incident were initially that the machine should not have been opened to the Internet in such at all, and that no automatic expiry was set for this 'temporary' measure. A similar temporary change to the firewall rule set allowed the SQL Slammer[SLM] worm inside the perimeter on another occasion.

Contributing factors here were lack of a properly enforced policy regarding firewall rules and the patching of machines. The lack of a proper system of tracking both meant this lack only came to light with the incident.

Incident #2 – the MSBlast/Blaster worm

After the issue of patch MS03-026 in July 2003, a request for everyone to make sure their machines were patched was sent out. Due to confusion over responsibility, lack of perceived importance and lack of perceived relevance of this notice, only around a quarter of desktop machines had been correctly patched by the time of the outbreak.

The first signs of infection by Blaster[BLS] were at 07:40h on Tuesday 12th August. These were observed as portscans across our network, from two separate hosts – one internal and the other a home machine which was connected via dial-up networking. Fortunately, an Intrusion Detection System had been installed earlier in July; in this case it was a Linux server running snort[SNT] and connected via a SPAN port to the core router at the main site. This meant it picked up portscans as soon as they had to traverse the core router, which was sufficient given the noisy nature of the worm's propagation attempts.

Detected portscan traffic from the worm increased in a roughly linear fashion from near zero at 7:40am to around 20,000 packets per minute by 8:40am. By mid-morning, the network was very sluggish and the blocking of the transmission vector of the worm – the tftp protocol on port 69/tcp – was a treatment which was almost as bad as the original problem. The extra filtering on the routers was done by relatively slow software rather than in the quick hardware, which led to very poor network performance, however, this did slow the spread of the worm.

As the network was basically unusable and the vulnerable computers kept rebooted due to the worm's constant attempts at infection, computers were patched by hand for the most part, with the IDS being used to track the infected hosts, and later on to initiate automated clean-up scripts. It was obvious that we needed to make major improvements in patching to prevent this from recurring on such a large scale, and ideally we also wanted to get the IDS to respond in some automated fashion to slow the spread of any future infection.

The worm was able to enter the network due to poor access control between the external and internal networks – remote access and laptops were both possible vectors for the worm to be carried within the security perimeter. The lack of a patch management system meant that there was a very large vulnerable population for the worm to target.

Incident #3 – the Welchia worm

A custom-written patching solution had been deployed by the time Welchia[WCH] was released, although it took a few weeks to roll a patch out to 90% of the desktop computers. However, there were still sufficient machines to crash the network for several reasons:

1. The patch solution was not 100% effective due to many misconfigured machines. Some administrators had deliberately removed domain administrators from the administrators group on their machines. Other machines had insufficient disk space to copy patches, or did not have the standard administrative shares.
2. Welchia had a more aggressive scanning routine that could exhaust the spare memory on the core router with a smaller infected population.

This time the automatic response scripts were quickly adapted from the ones used during Blaster. These basically killed off the worm's process, deleted the executable and registry keys used to start it. They were invoked after a particular host had exceeded some threshold for ICMP activity, which was the scanning method used by Welchia. Manual response, for those that could not be cleaned automatically, was to disable the Ethernet port on the nearest upstream switch.

The remote access services had been moved behind an internal firewall, to prevent Blaster being re-introduced by a careless user. However this time the worm was brought inside the perimeter on someone's laptop. This hole was much harder to plug as lots of laptops were personal equipment and there were many hundreds which were allowed to connect to the internal network. Because of this there were sporadic outbreaks of Welchia until it deactivated on January 1st 2004, though none of these were anything like as disruptive as the first Blaster infection.

The patch management system was by no means ideal, but it had drastically reduced the vulnerable population. This in turn allowed the network to remain functioning most of the time, which allowed network-based countermeasures instead of entirely manual intervention for infected computers. The preferred solution would have been to install a management system such as Altiris, HfNetChk, UpdateExpert, etc. and to disconnect any machines from the network which did not meet basic security standards. This proved problematic as no policy decision was forthcoming about whether central IT support were allowed to do such a thing, and because of financial constraints.

Incident #4 – the Sasser worm

Fortunately, I learnt of the existence of Sasser[SSS] by reading the diary on the Internet Storm Center website[ISC] one Sunday night. This gave roughly 12 hours to prepare for the probable Monday morning appearance of the worm on our internal network. Most machines had again been patched by this stage.

By now, the IDS had a patch[FLX] added which allowed it to make some attempt to disrupt TCP connections. A rule was added to apply this functionality to the FTP transfers that Sasser used as the method of copying the worm's code between hosts. The automated tracking and cleaning scripts were written from descriptions provided by ISC, Symantec and other vendors. Thanks to the early warning of this worm, disruption was minimal. Because the remediation measures were effective initially, the network remained stable and thus the remediation measures continued to be effective. Had the infection been somewhat larger, and had the network collapsed as a result, the countermeasures would not have been effective and the spread would have been much harder to contain.

Incident #5 – Agobot/SDBot/RXBot variants

```

-----
#(4 - 1329104) [2005-03-25 03:39:49.297] [snort/2001372]
BLEEDING-EDGE IRC Trojan Reporting (Scan)
IPv4: yyy.yyy.231.32 -> zzz.zzz.163.59
      hlen=5 TOS=0 dlen=168 ID=18140 flags=0 offset=0 TTL=127 chksum=56572
TCP:  port=3023 -> dport: 8000  flags=***AP*** seq=1483308911
      ack=501861482 off=5 res=0 win=64331 urp=0 chksum=51363
Payload:  length = 128

000 : 50 52 49 56 4D 53 47 20 23 61 73 74 72 6F 20 3A   PRIVMSG #astro :
010 : 5B 53 43 41 4E 5D 3A 20 52 61 6E 64 6F 6D 20 50   [SCAN]: Random P
020 : 6F 72 74 20 53 63 61 6E 20 73 74 61 72 74 65 64   ort Scan started
030 : 20 6F 6E 20 yy yy yy 2E yy yy yy 2E 78 2E 78 3A   on yyy.yyy.x.x:
040 : 34 34 35 20 77 69 74 68 20 61 20 64 65 6C 61 79   445 with a delay
050 : 20 6F 66 20 35 20 73 65 63 6F 6E 64 73 20 66 6F   of 5 seconds fo
060 : 72 20 30 20 6D 69 6E 75 74 65 73 20 75 73 69 6E   r 0 minutes usin
070 : 67 20 32 30 30 20 74 68 72 65 61 64 73 2E 0D 0A   g 200 threads...

```

Figure 1: A captured packet showing a 'bot reporting to an IRC server.

The main distinction between a “bot” and a worm is that a bot has some central control channel; often this is accomplished via IRC. These “bots” can exhibit similar behaviour to the previous worms when scanning; however, they are typically controlled via an IRC channel and will only begin scanning/exploiting on command. Again, snort signatures and port scan detection have helped pick up these machines.

Some of these bots didn't have virus definitions published for days after they had first been observed on our network – perhaps due to more limited propagation than the worms – and this required manual analysis and clean up procedures. An interesting thing about one particular attack was that it occurred on Good Friday, just after the institution had shut for five days, as if the attacker had anticipated no-one would be around to notice anything amiss.

The bots exploited the same vulnerabilities that the earlier worms had, but since they were not spreading as widely as the various worms, often our antivirus vendors did not have signatures for them. They were also only visible to the IDS when they had been specifically asked to scan a network, where as the worms were always scanning and thus always visible. The largest network we saw a command to scan was an entire class A, some 2²⁴ IP addresses.

Incident #6 – SSH password guessing

Password guessing against SSH began to get a lot of publicity quite a while ago. We took measures to minimise this risk on our servers by:

1. turning off external SSH access except where really necessary
2. moving SSH services to an alternate port
3. monitoring failed logins

Number 2 is effectively security by obscurity, which is not inherently bad, as long as it's not the only preventative measure employed. In this case, there is an implicit point number four in that we trusted that all the administrators had strong passwords. In this case, moving to an alternate port greatly cut down on the number of failed login attempts we had to review daily.

Unfortunately, other administrators around the campus did not take similar measures and several machines were compromised via password guessing attacks using SSH.

On one of these compromised machines, the attacker had copied a tar file of privilege escalation exploits and proceeded to try various ways to become root. When none of the exploit programs worked, the attacker attempted to send a phishing email relating to ebay.com. This was not successful as the exim distribution on that machine had not been configured to send mail. After this, the attacker began to use that server to guess passwords on other computers. This activity showed up very definitely on the IDS as port scans across port 22/tcp. Examination of the machine showed the attacker had been careless enough to leave all the source files around, as well as their bash history file. The sequence of actions was as follows:

The account compromised was “upload” with a password of “upload.” The first login was Sunday 23:01h local time. The attacker issued a 'wget' command to download a set of local root exploits. S/he tried the 'brk' and 'brk2' exploits[BRK] but gave up after this failed. The second action was to try and send out the eBay phishing emails, which failed as the host hadn't been configured to send email. The third and last action was to start scanning various networks for other vulnerable machines

The other machine also sparked an investigation after generating port scan alerts on port 22; i.e. it was also probably trying out SSH user names and passwords on other people's computers. This one had a copy of the 'adore' rootkit, so all of the files and directories would have been hidden from the administrator. Another directory contained a number of privilege escalation exploits, such as suid perl, do_brk. There were also some apparent attack tools for Denial of Service attacks.

Both machines were disconnected from the network immediately on discovery and booted from a Knoppix[KNP] CD which was used to examine the file system – though Helix[HLX] may be preferable to Knoppix as it has been specifically designed for forensics. A copy of the file system was taken for analysis and the server was then re-installed from scratch. SSH access from the Internet was disabled until the cause of the compromise had been identified; AusCERT[ACT] was also informed and sent a copy of the malware from the machines.

Had we been able to disseminate our knowledge more effectively to other system administrators, and to enforce it by means of audits, these problems would not have occurred. One of the machines compromised was set up in a secure manner, apart from the 'upload' account which had been created “temporarily” and then forgotten about.

Incident #7 – spam relayed via a broken formmail.pl script

In this case, a third-party contractor had used a vulnerable formmail.pl script which was abused to send spam across the net. Dire warnings about this had been circulated to employees who dealt with web pages, but the contractor was apparently unaware of the security risks of this script.

The incident came to light after a complaint was sent to the abuse handling address for the domain. The email server was immediately halted while the remaining spam emails in the queue were deleted. A keyword filter was put in place to prevent recurrences of the incident while a permanent fix was being sought regarding the removal of the broken formmail script.

This incident highlights the importance of having proper standards, such as ISO17799's system acceptance criteria, and of making sure they are requirements on all IT projects, whether implemented in-house or by external contractors.

Incident #8 – 'defaced' web pages

Due to misconfigured permissions, someone managed to upload an image to a web server. No existing content was damaged. Notification was received from AusCERT and from other parties.

There was an inadequate audit trail for this particular activity; someone at some stage had made a particular directory writeable to everyone via the WebDAV protocol, and in due course, this had been exploited. Some information was present in the logs of the reverse proxy that sits in front of the web servers, though not enough for a full reconstruction of the incident.

Incident #9 – password guessing attempts against POP3

Here, an external attack was mounted against the passwords for several user accounts by guessing at user passwords on the mail server. Since this is an on-line attack, the attacker would probably only recover the easier passwords, but that may be enough – see the SSH attacks covered earlier. Again, this was picked up by the IDS portscan detection, since a great many connections were made to the POP3 port on the mail server in a very short time.

This attack did lead to the compromise of some passwords, since the password requirements are somewhat lax. Enforcing stronger user passwords would remove a lot of the risks posed by on-line attacks such as this.

Incident #10 – an email worm

This incident was somewhat surprising; in general email worms were not an issue in this environment due to rigorous virus scanning procedures and the quarantine of executable attachments. One worm did manage to spread since the email body did not contain an attachment, but simply a URL where the worm code was located. This was used in combination with social engineering to entice the users to run the code. Once discovered, the incident was easy to contain as the offending website was quickly blocked, and keyword filters were added to our mail servers to removed the messages from circulation.

This could have been mitigated by better user education and possibly avoided by the implementation of virus scanning on all HTTP and FTP traffic, though this would still leave P2P networks as a means of spreading malicious code.

Conclusion

Design

Prevention is far easier and cheaper, for system administrators at least, than detection and response. However, any reasonably complex network is likely to experience incidents at one time or another and it's important to be prepared. As Bruce Schneier said at AusCERT 2005, don't just build systems which work, design systems which won't break. When you are designing, consider what would a malicious attacker could do to subvert your security measures.

Remember that people are part of your security system. They will forget things, cut corners and try to bypass security controls where these get in their way. Auditing configurations is essential; we have discovered several servers with "temporarily" blank administrator passwords. Reminders and automatic expiry dates are useful too – a permanent entry in the firewall may persist after the machine it refers to has been moved, assigned to a different project, or even put in a corner and forgotten about entirely. Most of our failures came from inadequate policy and procedures which were not followed and not enforced via auditing.

Using databases for configuration management reduces workload on staff and simplifies the process of auditing configurations. Keeping computer configuration in such a way greatly simplifies the process of rebuilding a system. For example, Subversion could be used to keep changes to /etc on UNIX systems, or something like Altiris can be used to store changes to Windows systems in such a way as to make applying them an automatic operation.

Prevention

All unnecessary services should be disabled. In fact, anything which should never happen is best pre-emptively blocked. Is there any reason one workstation needs to be offering file sharing services to another one? If not, why not firewall them off from one another? It will likely prevent or slow the spread of a virus throughout your entire network. Similarly, does your web server need to initiate any connections to the outside world? The firewall can be configured to prevent this, which may stop some attacks against web servers. Do you need netcat/wget/ftp/tftp/lynx binaries on your servers? These are the tools of choice for attackers to download rootkits and other malicious tools onto your machines.

Detection

For critical servers, think about using a host-based IDS such as tripwire[TRP] to check on the integrity of critical system binaries and configuration files.

If you do not have an audit trail for a particular activity, there is no way to discipline people for breaking your acceptable use rules. If a user has a particularly weak password, there is no way to be sure that they performed a particular action rather than an attacker.

There is usually no reason for any machine to be making many connections to other hosts over a sustained period. One significant advantage the system administrator has over the attacker is domain knowledge. Programmers liberally sprinkle their code with assert statements to flag up "impossible" condition. Similarly, system administrators can either outright stop things which shouldn't happen, or set up an email or pager alert. For example, denying port 53 outbound on the firewall revealed machines which had been redirected to malicious DNS servers.

You may learn of a system compromise from administrators of external systems, from your own IDS or syslogs, from your local CERT, or perhaps via looking at Dshield.org[DSH], or zone-h [ZNH] (a defacement archive). The more quickly you receive this information, and act on it, the easier it is to contain the incident and the less time you will have to spend cleaning up afterwards.

None of these attacks were carried out by particularly sophisticated attackers. If one of the SSH scans had been particularly slow – say one host per second – the attacker could have left it running for a long time to collect the data. If the IDS had not been present, or if the alerts had not been reviewed, one of the SSH compromises would have been missed. Perhaps there were other incidents which we did miss.

Incident Response

If you are examining a system you think may be compromised, do not log in to it – you could be sending the attacker your user name and password. Boot off a live CD such as Knoppix[KNP], or preferably Helix[HLX] and examine the file system and any other network logs you have. This gives you the assurance you are working on a 'known-good' system, and that no backdoors, rootkits, or trojaned binaries are in use.

You can't always rely on your antivirus vendors. In Australasia, we get Monday morning, hence the first working hours of the week, before most of the rest of the world. We can expect to see some large outbreaks of worms and viruses before other countries and often before there are good analyses or AV definitions published.

References

- [ACD] Analysis Console for Intrusion Detection, <http://acidlab.sourceforge.net/>
- [ACT] AusCERT, <http://www.auscert.com.au/>
- [BAS] Basic Analysis and Security Engine console, <http://sourceforge.net/projects/secureideas>
- [BLS] The MSBlast/Blaster worm,
<http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html>
<http://www.sophos.com/virusinfo/analyses/w32blastera.html>
- [BLD] Bleeding edge snort signatures, <http://bleeding-snort.com>
- [BOT] 'Bot signatures, <http://cert.uni-stuttgart.de/doc/netsec/bots.php>
- [BRK] Do_brk vulnerability, http://isec.pl/vulnerabilities/isec-0012-do_brk.txt
- [DSH] Dshield, <http://www.dshield.org/>
- [FLX] Flex_resp2 patch, http://cerberus.sourcefire.com/~jeff/archives/snort/sp_respond2/
- [HLX] Helix bootable forensics CD, <http://www.e-fense.com/helix/>
- [ISC] Internet Storm Center diary, <http://isc.sans.org/>
- [ISC05] Internet Storm Center diary, <http://isc.sans.org/diary.php?storyid=819>
- [KNP] Knoppix, bootable live Linux CD, <http://www.knoppix.org>
- [PST] Sysinternals pstools, <http://www.sysinternals.com/utilities/pstools.html>
- [SSS] The Sasser worm,
<http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html> ,
<http://www.sophos.com/virusinfo/analyses/w32sassera.html>
- [SLM] The SQL Slammer/Sapphire worm,
<http://securityresponse.symantec.com/avcenter/venc/data/w32.sqlexp.worm.html> ,
<http://www.us.sophos.com/virusinfo/analyses/w32sqlslama.html>
- [SNT] The Snort Intrusion Detection System, <http://www.snort.org/>
- [TRP] Tripwire, <http://tripwire.com/>
- [WCH] The Welchia/Nachi worm,
<http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html>
<http://www.sophos.com/virusinfo/analyses/w32nachia.html>
- [ZNH] Zone-h, <http://www.zone-h.org/>