Global Information Assurance Certification Paper

Network Security Analysis, Unseen University
5/19/2002 through 5/22/2002

GCIA Practical Assignment
Version 4.0
Andrew Magnusson
10/24/2004

Table of Contents

Table of Contents 2
Abstract 3
I. Executive Summary 4
II. Detailed Analysis 6
1. Chosen Scenario 6
2. Relationship Analyses 6
3. Network Detects and Analyses 7
i. Overview of Detects 7
ii. Detect I 10
iii. Detect II 12
iv. Detect III 16
4. Network Statistics 19
5. Correlations 22
6. Internal Dangers or Anomalies 23
7. Defensive Recommendations 23
III. Analysis Process 25
Appendix A. Perl source listing 27
References 30

Abstract

The following document is a security analysis for an unknown university, henceforth 'Unseen University'. I analyze four days' of Snort RAW logs from http://isc.sans.org/logs/raw/ and describe the various attacks found therein. Three of these attacks are pulled out for further study, then I describe some general statistics of the logfiles, and do a small amount of research on the three most suspicious external IP addresses. Finally, I describe for the university staff some internal anomalies that should be investigated more thoroughly, and provide a series of recommendations to improve their network's security standing.

In a final section I describe my analysis procedure, and provide a Perl source listing for a script I wrote to aid my analysis.

I. Executive Summary

Introduction

This paper is a first attempt at a comprehensive network security audit of Unseen University[1]. I was provided with the intrusion detection alert logs for four days in May, 2002, and carefully analyzed them in order to get a clear idea of the strengths and weaknesses of the University's network security framework. This paper details my findings, and concludes with a list of substantive recommendations to improve overall network security at the University.

The remainder of this executive summary is divided into two parts: an overview of the vulnerabilities discovered in my analysis and a description of the specific technical recommendations that I have assembled to address these and other vulnerabilities.

Findings

For this security review I analyzed four days' of Snort intrusion detection system (IDS) logs to compile a list of true attacks and false positives. The logs provided were 'raw' logs generated by the Snort IDS; these logs preserve the exact network traffic tha caused alerts but do not record any other information, such as which Snort rule captured them. Thus, some of my conclusions are unavoidably speculative, but I believe that I've managed to get a fairly accurate picture of the attacks during this period. This section describes some of the more important attacks I've analyzed in Part II.

First, and perhaps most importantly, there were a number of attacks indicating that several machines on the University network may be infected with a Trojan, which enables an external attacker to connect remotely and control the system. If this Trojan is in fact on any of the University's systems, it represents a significant danger to the network and must be addressed immediately.

Second, there were a large number of reconnaissance attempts detected. There were a number of 'nmap' scans, a well-known reconnaissance utility. It is likely that these reconnaissance attempts succeeded and responses were sent by the targeted machines. Likewise, there were a number of reconnaissance attempts by several sources looking for open proxies, which are services that allow an attacker to 'bounce' his or her network connections (usually web, but sometimes other services) through the proxy in order to mask their true source. While these recon packets made it through the firewall, there is no danger unless any of these destinations are in fact running proxy software. In both cases, however, the firewall can be configured to block reconnaissance attempts like this. (This will be discussed further in the 'recommendations' section.)

A third category of attacks I discussed were web application attacks. For instance, a large number of requests were made to the unseen.edu webserver, looking for an older version of the 'Formmail' CGI script. If your server is running a vulnerable version of Formmail, it will allow attackers to relay spam through your servers, which could create a PR nightmare for Unseen University. (The remainder of the web attacks appear

[1] In this paper, I will refer to the institution as Unseen University, and replace all actual DNS references with 'unseen.edu'.

directed at Windows servers, while the unseen.edu server is Linux.) I've recommended carefully checking all software versions, including the formmail script, to ensure they are all up to date in order to mitigate the risk of an attack like this succeeding.

The final type of anomaly I analyzed in detail was an unusual packet coming from what appears to be a VPN server on the Internet. It was picked up because it is destined to UDP port 0, which is an illegal destination port, and packets of this sort are known to cause Checkpoint Firewall-1 firewalls to reboot. Whether or not this is a real attack is immaterial; if the University is using a Checkpoint firewall this packet may have caused a crash. Newer versions of FW-1 have resolved this particular vulnerability, so a simple upgrade will render your FW-1 firewall, if you have one, immune.

By no means is this an exhaustive list of the attacks I looked at; for a more comprehensive discussion please see II.3.i below.

Recommendations

While few major attacks were detected in this four-day period, the breadth of recorded alerts indicates many weaknesses in the University's network security standing. The University's external firewall, if any, is not configured to block many sorts of illegitimate network traffic, and the IDS that generated these alerts needs further configuration and tuning before it will be truly effective in recording legitimate attacks while keeping false positives to a minimum. Finally, some University servers could be hardened somewhat against future attacks. This section summarizes the recommendations that I have compiled for the University's technical staff.

For the various hosts on the University network, I've recommended a three-pronged approach. First, a host-based firewall should be installed, if possible, and configured to allow in only legitimate traffic to services known to be running on the system. Second, network and LAN staff should ensure that every machine is running the most recent versions of their software and OS to minimize the effects of a network attack. (A patch mangement system such as Patchlink might be very useful here, especially for Windows-based systems.) They should also have up-to-date virus protection to protect against most viruses and Trojans. In places where this is impossible to ensure, like student workstations, a policy of restrictive firewall rules and careful IDS monitoring will have to suffice to minimize the effect of any intrusion on those systems. Finally, University servers can be further configured to give out less information about their running configuration, making an attacker's job more difficult.

The University firewall should have its ruleset reviewed and tuned for maximum protection. While the most secure method is a 'default deny' policy, this may be infeasible to implement in a university setting. In this case, I've described for network staff several of the most important things to block through the firewall. At the very least, however, the firewall should be configured to deny traffic that is by definition illegal.

When the firewall is in a 'default allow' mode, effective IDS functionality is more important than ever. The current IDS setup appears fairly untuned, as it generates inappropriate alerts (for instance, alerting on a Windows webserver attack when the attacked machine is Linux and not vulnerable) and may not be capturing all of the malicious traffic aimed at the University. I described a series of steps that the network staff can take in order to tune the IDS and improve the IDS infrastructure, by tuning the ruleset and, if possible, adding more IDS sensors. This will allow each sensor to be

customized to the network segment it's listening to, and lead to a more effective IDS alerting procedure.

II. Detailed Analysis

1. Chosen Scenario

I have chosen to analyze a four-day period covered by the RAW files 2002.4.19, 2002.4.20, 2002.4.21, and 2002.4.22. These files were downloaded from http://isc.sans.org/logs/Raw in August 2004. Notwithstanding the filenames, the data contained therein actually covers the period of 5/19 through 5/22 (presumably still from 2002). These files were generated from Snort running in binary logging mode, which records all packets that trigger one or more of its rules. This format is particularly useful because it preserves all of the information of the original packet, allowing the analyst to verify the event independently of Snort's signature-based detection

The downside to this format is twofold, however. First, an external analyst is given no hint of what rules triggered for each packet to be tagged. Second, there is no context given for any packet. While there is certainly a large stream of data surrounding each of these detects, none of this ancillary traffic is included. These two problems lead to a difficulty in interpretation for the analyst, who is forced to look at packets with no context and no hint why they were tagged. However, even in this situation it is possible to come to a reasonably clear view of the network's overall health, and such a thing I have attempted. Interested parties may find a more complete description of my analysis procedure in Section III of this paper.

2. Relationship Analyses

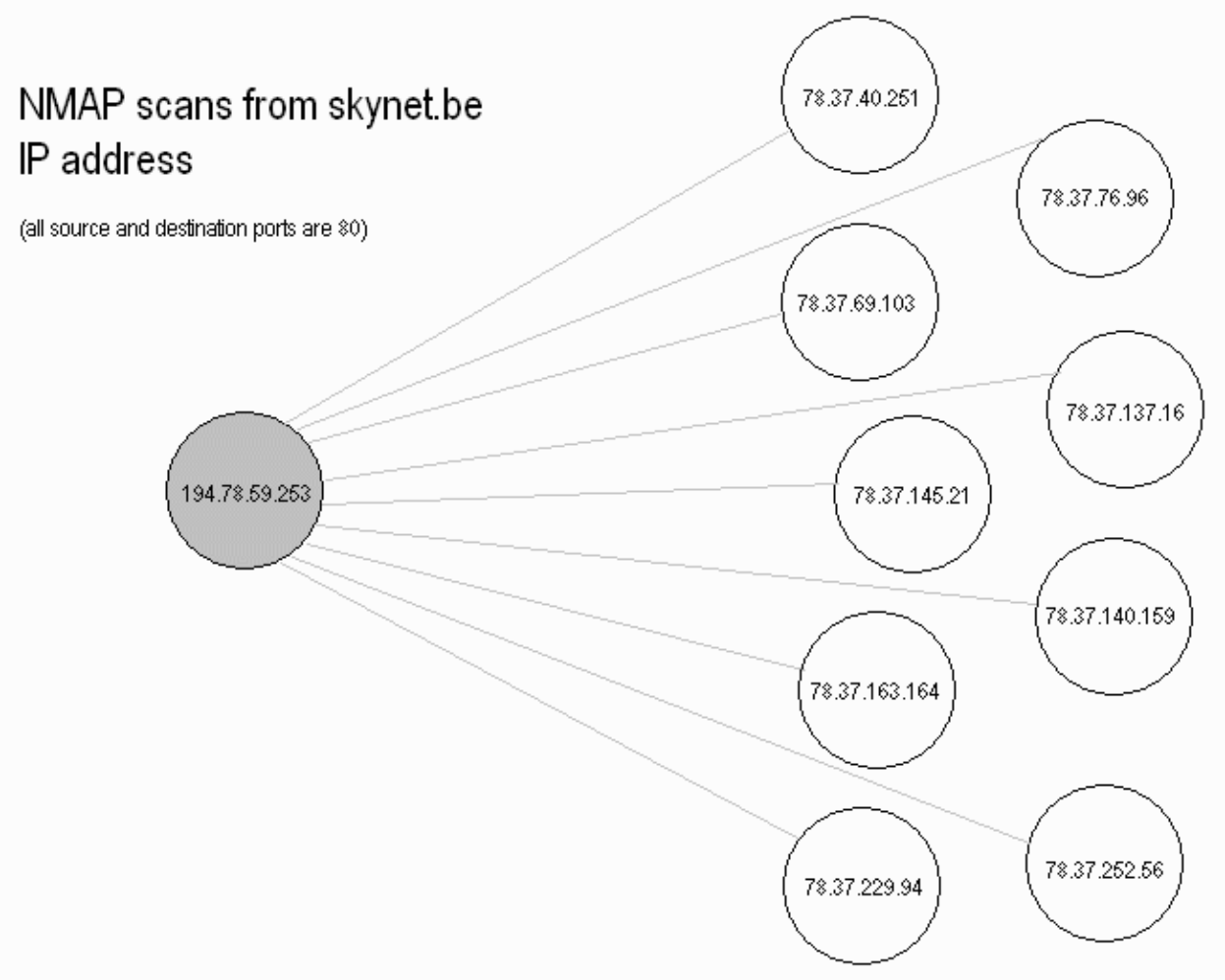
The internal network in this scenario is 78.37.0.0/16. Only two Ethernet addresses appear in these packets, 00:00:0c:04:b2:33 (inside) and 00:03:e3:d9:26:c0 (outside). As such, it appears as though the Snort IDS is listening to a wire between an internal router (or firewall) and an external router (or firewall) with an Internet uplink somewhere upstream. Both MAC addresses are Cisco devices, according to the IEEE Organizationally Unique Identifier (OUI) list[2].

Based upon the types of traffic observed in the captured packets in these files, the function of a number of internal IP addresses can be determined. 78.37.212.165 is an Apache web server (1.3.12) running on Redhat Linux, according to its server response headers (probably version 6.2, as that version of Redhat included a prebuilt RPM of Apache 1.3.12), and also appears to have a running FTP server of unknown provenance. 78.37.212.173 is the mail server for unseen.edu. 78.37.212.28 is something of a mystery; while there is a significant amount of web and Gnutella traffic from this IP address, indicating a user's workstation, it also may be listening on port 53 TCP, indicating a running DNS server. (Some packets were logged that are addressed to TCP 53 on this machine with the ack flag set, indicating either an ongoing conversation or an attempt to confuse the host or an intermediate device.) Perhaps this IP address is a NAT device; it is impossible to say for sure.

[2] IEEE OUI codes, http://standards.ieee.org/regauth/oui/oui.txt

The remaining addresses on this network that exist in these capture files are unknown devices. From the TTLs, the environment appears to be mixed Windows/Linux. (Further analysis of several hosts on this network will be performed in the three in-depth detects.)

Following is a link graph demonstrating one relationship discussed further on: an NMAP scan sequence from 194.78.59.253 to various internal addresses.



3. Network Detects and Analyses

i. Overview of Detects

Following is a brief description of each network detect that appears to be an accurate alert or at least worth brief discussion. For clarity, I have grouped closely related alerts together, for instance listing scans for three different types of proxy under the same item. Three of these detects are analyzed further in sections ii, iii, and iv.

- 166 packets were pulled from the alert logs which were indicative of the Q Trojan. For more information, please see Detect 1 in Section ii below.

- A large number of http packets – 798, to be exact – were recorded from 78.37.212.28. (“bare byte unicode encoding” – 771, all seem to be harmless, though 740 of those are hits to hitbox.com, a spyware provider. 16 packets were “apache whitespace”, and 11 packets “non-rfc http delimiter”, all of which to be spyware or ad-related false positives as well.)
- 1195 Gnutella connect and data packets (search results, etc) were picked up. Nearly all of this traffic is from or to 78.37.212.28, but a couple are from 78.37.250.214. While generally benign, Gnutella traffic can cause network congestion and may be involved in copyright-infringing filesharing.
- 1 packet of UDP port 0 from 159.75.232.253 to 78.37.212.28. For more information, please see Detect 2 in Section iii below.
- 156 UDP DNS packets were noted, all of which were BIND version queries to various hosts on the Unseen University network. This was an untargeted scan, and does not appear to have hit any real DNS servers.
- There were five TCP port 53 packets from various outside hosts to 78.37.212.28; all were zone transfer queries for unseen.edu. This IP does not appear to be a DNS server (in fact, its traffic pattern indicates that it is a workstation), so unless it is actually a NAT device or firewall of some sort then this was a completely un- or mis-targeted attack.
- 277 packets; TCP 21 to 78.37.212.165. All of these were FTP ‘anonymous’ user login attempts. It is unknown whether or not these logins succeeded. These login attempts were spread over all four days and from many source addresses. 164.164.60.11 (on two occasions) and 61.144.60.18 were particularly persistent in their connection attempts.
- 85 packets attacking the webserver at 78.37.212.165 were logged. Three of these were Code Red II probes, and the rest were attempted privilege escalations and other nastiness with FrontPage extensions (82 packets). Because this is a Redhat web server, these IIS-reliant attacks are not particularly concerning.
- Two packets from 24.217.114.34 to 78.37.233.40 port 80 were picked up, both attempted directory traversal attacks against a Windows OS webserver. It is unknown whether or not this target runs Windows, but since the packets detected appear to be from a real TCP connection, it is certainly running a webserver on port 80. If this is a Windows host, then it should be checked to see whether its webserver software allows directory traversal commands like these.
- 34 packets containing Formmail script exploit attempts against 78.37.212.28 were noted. If Formmail is running on this server, it should be checked to verify that it does not allow mail relaying in order that Unseen University not become an unwitting spam relay. Please see Detect 3 in Section iv below for further analysis of this automated attack.
- 110 packets in toto were recorded that appear to be NMAP scans. These scans (generally picked up by the consistent ACK value of 0) were targeted at 78.37.161.181, 78.37.212.30, 78.37.11.210, 78.37.250.214, 78.37.212.165 (web server), and 78.37.212.173 (mail server). These scans came from multiple sources, and all apparently reached their destination. While it is troubling that these packets were not blocked by the firewall, no harm was done beyond

reconnaissance. The recommendations that I outline below will be useful in blocking this sort of traffic in future.

- One FIN scan packet was recorded, from 80.212.202.187:1638 to 78.37.11.210:6346. No further traffic was recorded in the alert files to or from either of these IP addresses. The destination port is typical of Gnutella connections, so this may just be a broken Gnutella packet, not an actual scan attempt. If so, however, we would expect to see more Gnutella packets recorded to or from this internal host.
- One packet from 192.16.19.42 to 78.37.100.225 matched the Snort rule 'IP reserved bit set' (1:523). This is a badly out-of-spec packet with the reserved IP bit set and a missing TCP header. Its fragment offset is very high (17184). This is most likely a crafted reconnaissance packet of unknown provenance. No further packets were recorded to or from the source of this alert.
- One packet from 61.125.134.88 to 78.37.250.214 matched the Snort rule 'misc tiny fragment' (1:522). This is a small fragmented packet with the 'more fragments' flag set, which is unusual traffic to say the least. Generally only the last packet in a fragment train is particularly small. It also has a very high fragment offset (61816), so either this is traffic fragmented over an exceedingly tiny link, or, far more likely, this is a packet crafted for reconnaissance or attack. Unfortunately, since the rest of the traffic was not recorded it is impossible to say for certain what the situation is here.
- Four truncated packets were noted, from two external sources (3 from 193.100.10.5, 1 from 210.15.18.8) to the internal network. These may be reconnaissance attempts, or may be harmless, but the external university router should be configured to drop packets like this regardless.
- A total of 111 packets were found that are scans for open proxies. 67 of these were scans for SOCKS proxies on port 1080, 10 for Squid on port 3128, and 34 for a generic proxy on port 8080. These are all SYN packets, so most likely it's a series of untargeted scans.
- 46 packets were found in the capture logs, all http replies from a webserver at 78.37.212.165. Specifically, all of these packets are '403 forbidden' notices. Most of the 403 replies were for PDF and Powerpoint files, and a few requests for the server root. While it might be informative to follow up on these denied requests if time permits, this is not a high priority.
- 162 packets were picked up by Snort as potential shellcode exploits. All but one of them are to or from 78.37.212.28 and seem to be all either gnutella or FTP data false positives. (All of the seeming FTP traffic to this host is to legitimate FTP servers on the Internet.) The exception is 202.140.159.50:3579 -> 78.37.212.173:25. The NOOP sled found in this packet, part of an email, appears to be a false positive, since there doesn't seem to be a payload following the NOOPs. As it is part of an email, though, it is possible that there is an exploit in the next packet, to be triggered when the mail server reassembles the message for processing. Because no other packets from this session were captured, however, it's impossible to say.

ii. Detect I

Description

Over four days, there were 166 packets detected with the source address of 255.255.255.255 and multiple destination addresses on the local (78.37.0.0/16) network. The packets all have a source port of 31337, and a destination port of 515, which is typically used for printer communication. Two representative packets follow:

```
20:26:35.754488 IP (tos 0x0, ttl  13, id 0, offset 0, flags [none], length: 43,
bad cksum 5fcb (->1884)!) 255.255.255.255.31337 > 78.37.71.37.515: R [bad tcp ck
sum 12f3 (->cbab)!] 0:3(3) ack 0 win 0 [RST cko]
        0x0000:  4500 002b 0000 0000 0d06 5fcb ffff ffff  E..+......_.....
        0x0010:  4e25 4725 7a69 0203 0000 0000 0000 0000  N%G%zi..........
        0x0020:  5014 0000 12f3 0000 636b 6f00 0000       P.......cko...
21:48:05.794488 IP (tos 0x0, ttl  13, id 0, offset 0, flags [none], length: 43,
bad cksum 499d (->256)!) 255.255.255.255.31337 > 78.37.93.83.515: R [bad tcp cks
um fcc4 (->b57d)!] 0:3(3) ack 0 win 0 [RST cko]
        0x0000:  4500 002b 0000 0000 0d06 499d ffff ffff  E..+......I.....
        0x0010:  4e25 5d53 7a69 0203 0000 0000 0000 0000  N%]Szi..........
        0x0020:  5014 0000 fcc4 0000 636b 6f00 0000       P.......cko...
```

Please note that although these packets were picked up by Snort's 'BACKDOOR Q' rule, it is impossible to say for certain that they were generated by the Q Trojan. Certainly Q is capable of creating TCP packets of this sort, but the rule (described below) is vague enough that it could be almost any sort of crafted traffic. The following discussion is predicated on the assumption that this is indeed Q traffic, but the defensive recommendations are the same no matter the provenance of these packets.

Reason selected

This detect was selected due to the shamelessly crafted nature of the packets in question. Additionally, these packets are likely to be control commands for the Q Trojan which, if present on any systems on the local network, represents a significant intrusion that must be further investigated and resolved by the local network administrators.

Generated by

This alert was generated by Snort running in binary logging mode. The configured ruleset is unknown, however the following rule is in the default Snort ruleset and was the most likely trigger:

```
alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q access"; dsize:>1; flags:A+;
flow:stateless; reference:arachnids,203; classtype:misc-activity; sid:184; rev:6;)
```

This rule is in the backdoor.rules file.

Probability spoofed

These packets were certainly spoofed. First, 31337 is a very suspicious source port. It is often associated with the BackOrifice Trojan, and in 'leet speak' (script kiddie slang) '31337' is supposed to mean 'elite'. Second, 255.255.255.255 is not a valid source address. RFC 919, "Broadcasting Internet Datagrams", describes this address as "a

broadcast on a local hardware network, **which must not be forwarded**."[3] (emphasis mine) 255.255.255.255 is not a valid source address; it is used only as a broadcast destination. What's more, broadcasts of this sort are used only with UDP (stateless) traffic, as the TCP protocol does not allow for simultaneous broadcast connections. As such, this packet is by definition invalid and therefore crafted[4]. Finally, if any more evidence of crafting were needed, the TTL (13), IP ID (0), and acknowledgement numbers (0) are all fairly suspicious. A TTL (time to live) of 13 means either that the source host is very far from the destination or it has a very low initial TTL. While some systems have initial TTLs as low as 30, most are 64 or 128. The IP ID field helps the TCP/IP stack uniquely identify the packet. While 0 is valid, it should be very rare, and all of these have that value. The same goes for the acknowledgement number – 0 isn't illegal, but shouldn't be in all of the packets either.

Mechanism

The Q Trojan is a family of backdoor processes written starting in 1999 by Mixter as 'proof-of-concept' backdoor code. Notwithstanding his stated intentions, however, they were quickly adopted by the script-kiddie contingent and have been plaguing networks ever since[5]. Q is a client-server pair, both of which are available on multiple platforms including most versions of Unix and Windows. The server, 'qd', is installed on the system to be controlled remotely, and the client program 'qs' sends commands to it via raw IP packets which are crafted to look like TCP or UDP. Q is capable of encrypting its commands, though the current packets don't appear to contain any encrypted data. In fact, the only apparent payload is the string 'cko', which is interpreted by tcpdump as data on reset, a legitimate option to a RST packet. It's unknowable what, if anything, the attacker has configured the Q daemon to do upon receiving this command.

Correlations

The Q Trojan has been extensively discussed by Gordon[6] in his SANS FAQ on the Trojan. The Q source code is available from Mixter's site, mixter.void.ru[7]. Elsewhere on his site, Mixter has papers discussing Trojan detection and removal[8], and crafting raw IP packets in C[9]. The 'correlations' section below also lists several GCIA practicals that also discuss this alert activity.

---

[3] RFC 919, 5

[4] As a side note, these packets originate from outside the local network – the Ethernet address on all of these packets matches the external router address – so the local router broke the broadcast RFC by even allowing it in.

[5] This section owes much to the excellent analysis of this family of Trojans by Les Gordon. (see References)

[6] ibid.

[7] http://mixter.void.ru/Q-2.4.tgz

[8] http://mixter.void.ru/trojans.txt

[9] http://mixter.void.ru/rawip.txt

Active targeting?

There is some evidence of active targeting of these packets. They are sent to multiple addresses on the internal network, in no apparent pattern. With no other information available, and no knowledge of the payload that may be contained in these connections, it is impossible to tell whether this is an attempt at subtlety, and other ranges will be scanned later, or if it is commands sent only to hosts on the university network that are known by the attacker to be running the Q Trojan.

Severity

(2 + 5) – (1 + 2) = 4

Criticality: 2. None of these systems appear to be critical to network functionality. Further information about the network architecture might be cause to revise this number in either direction.

Lethality: 5. If this attack is against systems already infected with this Trojan, the attacker could cause them to do anything, from opening a shell to ping-flooding a third party. A workstation or server running the Q Trojan is completely controllable by the attacker.

System Countermeasures: 1. No defensive measures are known to be running on the target systems. If this attack is targeted, as speculated above, then the systems are already compromised. Otherwise, it is possible that the target machines are running some sort of host-based firewall; it is impossible to say. If host-based protection exists, this number may be revised upward.

Network Countermeasures: 2. If the routers to this point have not stopped this clearly RFC-breaking traffic, it's unlikely that any others before the target system will. What's more, if these packets are part of a targeted attack, then the attacker has already had access to these machines at least once, to find that they were running Q daemons. It has, however, been detected as an attack by Snort.

iii. Detect II

Description

One packet was found in the analyzed alert files with a UDP destination port of 0. Port 0 is reserved, and should not be a valid destination. The source port is 10000, which is often used by Cisco VPN devices. The packet follows:

```
15:04:47.094488 IP (tos 0x0, ttl  48, id 49454, offset 0, flags [none], length: 464, bad cksum
62ac (->1d64)!) 159.75.232.253.10000 > 78.37.212.28.0: [no cksum] UDP, length: 436
        0x0000:  4500 01d0 c12e 0000 3011 62ac 9f4b e8fd  E.......0.b..K..
        0x0010:  4e25 d41c 2710 0000 01bc 0000 4f86 426f  N%..'.......O.Bo
        0x0020:  0000 0001 0101 0101 0101 0101 a955 be5c  .............U.\
        0x0030:  2e53 4dff 34b3 9b11 3fd9 1a0b 8668 a8f0  .SM.4...?....h..
        0x0040:  9a8e 9b25 c889 daaf 34ba c5c4 bdf4 b6d7  ...%....4.......
        0x0050:  5bc6 2f6d a9a5 b9d8 eb73 fe2a 31d2 3434  [./m.....s.*1.44
        0x0060:  0c62 e540 17ba 00bb 4ef3 b810 e10e 9e6b  .b.@....N......k
        0x0070:  7fc9 12a1 3e2a 9bd2 dbf7 1a05 3e89 dbf9  ....>*......>...
        0x0080:  6257 9b7d 9c60 6415 70bf 338c f97d 3765  bW.}.`d.p.3..}7e
```

```
0x0090:  4f5d 4829 3d52 4bab 6ec9 bb4c 5562 9dd8  O]H)=RK.n..LUb..
0x00a0:  faa8 ff3c 8bc2 bf9f 519a ca29 244f 1230  ...<....Q..)$O.0
0x00b0:  1632 2869 b426 b7ba aee1 ac24 1e4a e11b  .2(i.&.....$.J..
0x00c0:  98b9 6cd6 bc28 d820 2ba6 4474 779e bbcc  ..l..(..+.Dtw...
0x00d0:  a1db 1ab7 405f 7fea 6555 9c43 5d7d 6151  ....@_..eU.C]}aQ
0x00e0:  b437 ae10 5f89 a9e3 f25f 35da 1a29 9d70  .7.._...._5..).p
0x00f0:  1b6d a9c4 e142 361a 201d 1ce0 f010 4f46  .m...B6.......OF
0x0100:  9382 e3b5 f6c7 5f4a 598d 0b27 8d60 8536  ......_JY..'.`.6
0x0110:  f2da 81a8 cc0a 0282 d648 1351 68a1 990a  .........H.Qh...
0x0120:  9a1e 1796 0d6d 1a96 18d8 37ac 7a47 1352  .....m....7.zG.R
0x0130:  fa50 9089 aab1 5ee8 281e 0917 590b b4fb  .P....^.(...Y...
0x0140:  bc3d da0e e303 e3db 83ce ed1d c059 a5d9  .=...........Y..
0x0150:  a042 13bf c847 2ac9 c95a ced7 5e99 0ea8  .B...G*..Z..^...
0x0160:  7615 be80 38e5 f136 36dd dc98 b411 9b38  v...8..66......8
0x0170:  1ab7 4c4a 5690 7c01 096d 08cd 35f4 cf64  ..LJV.|..m..5..d
0x0180:  10e5 2753 bbef 62a1 fd59 72b8 d963 bf42  ..'S..b..Yr..c.B
0x0190:  884a d09e 43dc 2ba2 cbd7 db24 16a3 3e2e  .J..C.+....$..>.
0x01a0:  58f0 b5f4 b2a2 026b 4e74 816f 70a8 5f81  X......kNt.op._.
0x01b0:  baee ce17 0d8c 6279 b8c8 12ef dbe6 ba83  ......by........
0x01c0:  dd59 2352 8e0c 8cd8 3218 e58a f128 ee2a  .Y#R....2....(.*
```

There is one attack known that utilizes VPN packets on UDP port 0 to attack Checkpoint Firewall-1 devices and cause a denial of service (DoS) condition. While it is not known whether Unseen University uses FW-1 firewalls, this potential attack deserves further investigation.

Reason selected

This detect was selected because it is potentially a very interesting attack and, while it is more than likely a false positive (more on this later) the fact that it arrived through the external defenses into the network points out at least one important configuration change that needs to be made to Unseen University's defenses.

Generated by

This alert was generated by Snort running in binary logging mode. The configured ruleset is unknown, however the following rule is in the default Snort ruleset and was the most likely trigger:

```
alert udp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD-TRAFFIC udp port 0 traffic";
reference:bugtraq,576; reference:cve,1999-0675; reference:nessus,10074; classtype:misc-activity;
sid:525; rev:9;)
```

This rule is in the bad-traffic.rules file.

Probability spoofed

The probability that this packet was somehow spoofed is fairly low, but not insignificant. The source address, 159.75.232.253, resolves to ww-vpn.ltx.com, which appears to be a legitimate VPN device at LTX, a corporation that provides electronics testing equipment. Other traffic from the destination address, 78.37.212.28, reveals a pattern of traffic to electronics-related websites, so it seems reasonable that this user might have a business relationship with LTX requiring a VPN connection. UDP port 10000 is a typical Cisco VPN source port, lending more credence to the belief that this packet is part of a VPN connection. However, the destination port of 0 is not a valid port, which means that either this VPN is misconfigured, the packets are getting mangled in transit, or this is a crafted packet.

If there is a Firewall-1 firewall on the network, however, then this packet is likely to have caused it problems whether or not it was crafted expressly for that purpose.

Mechanism

The attack under discussion occurs when an ISAKMP (VPN) packet is sent through a Checkpoint FW-1 firewall with a destination UDP port of 0. The final destination does not matter; as long as it goes through the FW-1 device then the attack succeeds. When this packet passes through an affected implementation of the FW-1 firewall, it causes the firewall to reboot itself, bringing about a temporary denial of service condition while the system reloads. While the specific packet we are examining may be legitimate, it will in any case still succeed in rebooting affected FW-1 systems.

It is believed that Solaris systems running FW-1 will reboot, however, details are sketchy for other host operating systems. I have seen no further study on this vulnerability to determine whether or not other operating systems are as vulnerable. Versions 3 and 4 of FW-1 are believed to be vulnerable to this attack.

Though the initial Bugtraq report (see below, in Correlations, for details on this posting) did not specify a mechanism, nor did any of the follow-ups, it is possible to speculate about how this sort of packet is capable of crashing firewalls running FW-1 software. Lance Spitzner has an article describing Firewall-1's state table functionality, and his following discussion of UDP state tables is illuminating:

> When a UDP packet is allowed through the firewall (based on the rulebase) a entry is added to the connections table. Any UDP packet can return within the timeout period (default 40 seconds) as long as both the SRC/DST IP addresses and SRC/DST ports match. For example, below is a DNS query.
>
> ```
> Src_IP         Src_Prt Dst_IP         Dst_Prt IP_prot Kbuf    Type    Flags          Timeout
> 192.168.1.10   1111    136.1.1.20     53      17      0       16386   ff01ff00       34/40
> 192.168.1.10   1111    136.1.1.20     0       17      0       16386   ff01ff00       34/40
> ```
>
> Here you see the system 192.168.1.10 doing a dns query to the server 136.1.1.20. For 40 seconds (Timeout) that system can return as many UDP packets as it wants, as long as both the SRC/DST IPs match, and the SRC/DST ports match. Notice how there is [sic] two entries, both are identical except for the Dst_Prt, which is 53 and 0. I do not know why FW-1 creates a second entry for a Dst_Prt of 0. However, this is common for most, if not all UDP traffic that FW-1 filters.[10]

As such, then, a FW-1 system encountering a packet with an actual destination port of 0 will probably attempt to create two identical state table entries, which may be the cause of the crash.

I was unable to find any discussion of this vulnerability on Checkpoint's support site, however a recent knowledge base article claims[11] that FW-1 drops all traffic with source

[10] Spitzner

[11] Checkpoint Knowledge Base, https://secureknowledge.checkpoint.com/sk/public/idsearch.jsp?id=sk27109

or destination port 0, so as of at least 9/16/2004 this vulnerability may be resolved by the simple expedient of ignoring all packets like this.

Correlations

This attack was first described on August 9, 1999 on the Bugtraq mailing list[12]. There isn't a whole lot of information available about this exact vulnerability, but some discussion can be found at http://www.securityfocus.com/bid/576 and http://cve.mitre.org/cgi-bin/cvename.cgi?name=1999-0675. The discussions on those pages, however, seem almost completely based upon the original Bugtraq posts by 'Malikai', and don't contain much further information.

Active targeting?

Because this vulnerability is against a firewall, not an endpoint, the destination address of this packet is not the actual destination of the intended attack, if any. If Unseen University uses a firewall running the FW-1 system, then the attacker may know this and is actively attempting to break it. If the University is not running FW-1, or if the attacker doesn't know, then if this is an attack it's a shot in the dark.

Severity

(4 + 2) – (1 + 2) = 3

Criticality: 4. The attacked device is a firewall. If it is brought down or caused to reboot, it may cause widespread network outages within the organization.

Lethality: 2. If Unseen University has a Checkpoint FW-1 device, and if it is running an affected version of the operating system, then this packet would likely cause a DoS. While dangerous, however, it would cause no lasting damage to the network environment. It is also by no means certain that there is a FW-1 firewall on this network, hence the lowered lethality score. This number may be revised upward or downward once the make and model of firewall are known.

System Countermeasures: 1. The Checkpoint firewall, if any, has permitted this packet and hence has no defense against it if in fact it is an attack.

Network Countermeasures: 2. This packet has not been blocked thus far, and if it has made it this far into the network, then whatever damage it may be capable of doing to the perimeter firewall has already occurred. It has, however, been detected as an attack by Snort.

[12] Malikai

iv. Detect III

Description

Over four days, thirty-four attempts were made to exploit the Formmail perl script. Formmail is a widely installed script that is used to power online 'contact' pages, sending site visitors' messages in email to a server-defined recipient. While the most recent version of this script is secure, older versions (1.9 and earlier[13]) allowed an unscrupulous web client to use the script to send email anywhere. Not surprisingly, this discovery was quickly put to very effective use in sending spam from various unsuspecting websites. These attempts come from many locations, but they all seem to use small variations of the same exploit tool, which apparently connects to vast numbers of websites and attempt to send email to a predefined address. If the command succeeds, then the attacker merely has to check his email to find a list of vulnerable Formmail scripts ready to exploit in a spam-sending campaign. Unfortunately, since only Snort-tagged packets show up in the logfiles, it is impossible to tell whether any of these attacks succeeded in causing the webserver to send email to any of the attackers' addresses.

One representative packet follows (identifiable host information redacted):

```
01:37:43.854488 IP (tos 0x0, ttl 112, id 33715, offset 0, flags [DF], length: 420)
65.58.238.117.4176 > 78.37.212.165.80: P [tcp sum ok] 243206 7773:2432068153(380) ack 2893207797
win 8160
            0x0000:  4500 01a4 83b3 4000 7006 3326 413a ee75  E.....@.p.3&A:.u
            0x0010:  4e25 d4a5 1050 0050 90f6 68bd ac72 d8f5  N%...P.P..h..r..
            0x0020:  5018 1fe0 fb52 0000 4745 5420 2f63 6769  P....R..GET./cgi
            0x0030:  2d62 696e 2f66 6f72 6d6d 6169 6c2e 706c  -bin/formmail.pl
            0x0040:  3f72 6563 6970 6965 6e74 3d66 6f72 6d6d  ?recipient=formm
            0x0050:  6169 6c69 6e66 6f40 7961 686f 6f2e 636f  ailinfo@yahoo.co
            0x0060:  6d26 7375 626a 6563 743d 6874 7470 3a2f  m&subject=http:/
            0x0070:  2f78 7878 2e78 7878 782e 7878 782f 6367  /xxx.xxxx.xxx/cg
            0x0080:  692d 6269 6e2f 666f 726d 6d61 696c 2e70  i-bin/formmail.p
            0x0090:  6c26 626f 6479 3d4a 7570 5a26 656d 6169  l&body=JupZ&emai
            0x00a0:  6c3d 6574 7540 616f 6c2e 636f 6d20 4854  l=etu@aol.com.HT
            0x00b0:  5450 2f31 2e31 0d0a 4163 6365 7074 3a20  TP/1.1..Accept:.
            0x00c0:  696d 6167 652f 6769 662c 2069 6d61 6765  image/gif,.image
            0x00d0:  2f78 2d78 6269 746d 6170 2c20 696d 6167  /x-xbitmap,.imag
            0x00e0:  652f 6a70 6567 2c20 696d 6167 652f 706a  e/jpeg,.image/pj
            0x00f0:  7065 672c 202a 2f2a 0d0a 4163 6365 7074  peg,.*/*..Accept
            0x0100:  2d4c 616e 6775 6167 653a 2065 6e2d 7573  -Language:.en-us
            0x0110:  0d0a 4163 6365 7074 2d45 6e63 6f64 696e  ..Accept-Encodin
            0x0120:  673a 2067 7a69 702c 2064 6566 6c61 7465  g:.gzip,.deflate
            0x0130:  0d0a 5573 6572 2d41 6765 6e74 3a20 4d6f  ..User-Agent:.Mo
            0x0140:  7a69 6c6c 612f 342e 3020 2863 6f6d 7061  zilla/4.0.(compa
            0x0150:  7469 626c 653b 204d 5349 4520 352e 303b  tible;.MSIE.5.0;
            0x0160:  2057 696e 646f 7773 2039 383b 2044 6967  .Windows.98;.Dig
            0x0170:  4578 7429 0d0a 486f 7374 3a20 7878 782e  Ext)..Host:.xxx.
            0x0180:  7878 7878 2e78 7878 0d0a 436f 6e6e 6563  xxxx.xxx..Connec
            0x0190:  7469 6f6e 3a20 4b65 6570 2d41 6c69 7665  tion:.Keep-Alive
            0x01a0:  0d0a 0d0a                                ....
```

[13] While Snort's rule description claims that only 1.6 and prior are vulnerable, the changelogs for 1.8 and 1.9 note several more anti-spam functions, indicating that at least 1.7 and 1.8 are also vulnerable. Coincidentally, the most recent version of Formmail, 1.92, was released on the third day of the present analysis range. See http://www.scriptarchive.com/readme/formmail.html#history for more information.

Reason selected

While attacks of this sort have been widespread since at least early 2001, many webservers are still running vulnerable versions of Formmail. Due to the significant potential damage, both technical and political, of Unseen University sending large amounts of spam, it is important to verify that the targeted server is not running a version of the Formmail script that is susceptible to being controlled in this way.

Generated by

This alert was generated by Snort running in binary logging mode. The configured ruleset is unknown, however the following rule is in the default Snort ruleset and was the most likely trigger:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI formmail access";
flow:to_server,established; uricontent:"/formmail"; nocase; reference:arachnids,226;
reference:bugtraq,1187; reference:bugtraq,2079; reference:cve,1999-0172; reference:cve,2000-0411;
reference:nessus,10076; reference:nessus,10782; classtype:web-application-activity; sid:884;
rev:14;)
```

This rule is in the web-cgi.rules file.

Probability spoofed

It is very unlikely that this traffic is spoofed, since making a valid web request requires at the minimum an active TCP connection. While TCP sessions can sometimes be hijacked by a determined attacker, it seems like overkill to do so in this situation, where the attackers are merely trolling for open spam relays.

Mechanism

The attack is performed by a specially formatted request to the Formmail script on the webserver. One example follows:

```
GET /cgi-bin/formmail.pl?recipient=formmailinfo@yahoo.com\
&subject=http://ftp.smsc.com/cgi-bin/formmail.pl\
&body=JupZ&email=etu@aol.com
```

This is an http GET request that passes four variables to the formmail.pl script: recipient (formmailinfo@yahoo.com), subject (URL of the script attacked), body (“JupZ”), and return address (etu@aol.com). While an attacker can manually create URLs like this, it is much more likely that this one is using an automated tool of some sort to quickly send these URLs to a long list of websites, in the hopes that one or more will allow the request and send the email. At this point, the attacker just has to watch his email box (formmailinfo@yahoo.com or perhaps etu@aol.com if he is being a little bit sneakier) for the emails to come in, with the subject lines being the exact URLs that are vulnerable to the exploit.

This request takes advantage of the fact that older versions of the script had no mechanism for verifying the ‘recipient’ field in the GET request for the script. While under normal circumstances the web developer would hard-code the recipient’s name into the form that called Formmail, an attacker could simply call the script manually and pass any values to it, including any recipient at all. Thus, spam can be relayed through a formmail script simply by encoding it into a GET request like the one above. Later versions of Formmail introduced an ‘allowed recipients’ configuration value which

mitigates this risk, but it took several more versions before the script was protected from more subtle attacks.

Correlations

A good overview of the 1.6 Formmail vulnerability can be found at http://www.net-security.org/article.php?id=503. There is a detailed paper circulating online describing in detail several attacks against Formmail 1.9; an example is at http://www.monkeys.com/anti-spam/formmail-advisory.pdf. (Versions 1.91 and 1.92 appear to have resolved these issues.) Automated attacks against Formmail scripts, much like the one apparently responsible for the present alerts, can be found in many places, such as http://www.ezgoal.com/security/f.asp?fid=61962. Finally, Formmail exploitation was third (of ten) in the SecurityFocus Top Attacks for the 1st Quarter 2002[14], approximately the time period of these alert logs.

Active targeting?

There is no evidence of active targeting, beyond the strong likelihood that these attackers are working from a list of known webservers. The attack was aimed at a URL, not an IP, which strengthens this hypothesis significantly. The various scripts that hit the webserver are looking for different names (FormMail.pl, formmail.PL, Formmail.cgi and so on) and sending to different addresses, so it appears to simply be a large number of 'script kiddies' blasting these attacks at large numbers of websites in an attempt to find some servers willing to relay anonymous email.

Severity

(4 + 3) – (2 + 2) = 3

Criticality: 4. The attacked system is a webserver, apparently the main webserver for unseen.edu.

Lethality: 3. The attack will do no harm to the server beyond eating up bandwidth with the floods of spam that are sure to follow a successful attack. As mentioned above, however, there may be significant PR consequences if the University is shown to be a source of spam.

System Countermeasures: 2. This attack will only succeed if the server is running a vulnerable Formmail with one of the names that the attackers used in their various attacks. Once this is determined, this value may be revised upward or downward. Another mitigating factor, if it is installed, might be an Apache module like Mod_security which can be configured to block queries of this sort.

Network Countermeasures: 2. As this is a valid web request, it is not surprising that the perimeter firewall, if any, has permitted it in to the web server. It has, however, been detected as an attack by Snort.

---

[14] SecurityFocus, "Top Attacks for the 1st Quarter 2002", http://www.securityfocus.com/corporate/research/top10attacks_q1_2002.shtml

4. Network Statistics

The list of top five attacked ports was determined purely by packet count. However, 'attack' is a subjective term here. For instance, the number two port on the list, 64507, appears to be a very extended false positive in the form of a large binary file (perhaps an image) downloaded from an IEEE site.

Top five attacked ports, by packet count

Port	Service	Count	Data
80T	http	9366	13275994
64507T	unknown	654	912636
6347T	Gnutella	549	46116
21T	FTP	277	12850
1863T	MSN Messenger	180	35221

The 'top talkers' list was also determined by total packet count. While the most important list here is those five most talkative sources, it is also useful to know the top five listeners, that is, the top five destinations for this suspect traffic. Like above, the second IP address is almost certainly the victim of mistaken identification, as it is an IEEE IP address and is the source of the aforementioned large binary file.

Top five source and destination IPs, by packet count

Sources	Count	Data	Destinations	Count	Data
78.37.212.28	10454	13340815	64.154.80.51	6192	8909262
63.84.220.222	660	921008	64.154.80.50	1606	2939079
255.255.255.255	166	5478	78.37.212.28	1394	1940530
128.9.176.20	112	166552	78.37.212.165	440	69425
61.144.60.18	99	5127	206.132.132.199	210	104142

Suspicious external hosts

61.144.60.18 was the source of 97 anonymous login attempts to the University webserver between 2:38 and 2:55 AM on 5/21, and two attempts to access Frontpage extensions on the server at 3:47 that same morning. While nothing seems to have come of these access attempts, this is still a suspicious source. The TTL of packets arriving from this IP is 104, so the initial TTL is likely 128, which indicates that this IP is probably running Windows 9X, NT, or 2000[15]. The window size, 17457, makes it most likely that the source is running Windows 2000. This IP address has no reverse DNS configured, but here are the most specific portions of a whois lookup. This host is an IP address in Guangzhou, China, and almost certainly has no legitimate reason to be attempting to access the University's webserver:

```
# ARIN WHOIS database, last updated 2004-10-23 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
% [whois.apnic.net node-2]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html
```

[15] Northcutt/Novak, Table 11.1, p. 211

```
inetnum:      61.144.60.0 - 61.144.60.255
netname:      GUANGZHOU-JS-BUILDING-CO-
descr:        JINSHAN BUILDING
country:      CN
admin-c:      GL217-AP
tech-c:       GL217-AP
mnt-by:       MAINT-CHINANET-GD
status:       ASSIGNED NON-PORTABLE
changed:      ipadm@gddc.com.cn 20011024
changed:      hm-changed@apnic.net 20040927
source:       APNIC

person:       GZ LEE
address:      NO.11,RO.TIYUDONG,GUANGZHOU
country:      CN
phone:        +86-20-38883360
fax-no:       +86-20-38883360
e-mail:       ipuser@gddc.com.cn
nic-hdl:      GL217-AP
mnt-by:       MAINT-CHINANET-GD
changed:      ipadm@gddc.com.cn 20011024
source:       APNIC
```

200.181.137.39 was the most persistent of the proxy scanners that hit the University network, probing 78.37.186.120 for SOCKS, Squid, and generic 8080 web proxies. Why it only hit this IP address is unknown. Its TTL upon hitting the IDS is 44, so it most likely had an initial TTL of 64, which is likely a Linux or other Unix system[16]. While its window size of 5808 is indicative of an HP JetDirect device, this isn't likely to be the case. Packets from this source also have DF set, and a TOS of 0, which matches Linux, Solaris, and OS/400. Of these, Linux is by far the most likely culprit. This IP resolves to 1-039.ctame701-1.telepar.net.br, which is probably a dynamic IP address from an ISP in Brazil. Relevant portions of its WHOIS record follow:

```
% Copyright LACNIC lacnic.net
%  The data below is provided for information purposes
%  and to assist persons in obtaining information about or
%  related to AS and IP numbers registrations
%  By submitting a whois query, you agree to use this data
%  only for lawful purposes.
%  2004-10-24 13:36:15 (BRT -03:00)

inetnum:     200.128/9
status:      allocated
owner:       Comite Gestor da Internet no Brasil
ownerid:     BR-CGIN-LACNIC
responsible: Frederico A C Neves
address:     Av. das Na??es Unidas, 11541, 7? andar
address:     04578-000 - S?o Paulo - SP
country:     BR
phone:       +55 11 9119-0304 []
owner-c:     CGB
tech-c:      CGB
inetrev:     200.128/9
nserver:     NS.DNS.BR
nsstat:      20041023 AA
nslastaa:    20041023
nserver:     NS1.DNS.BR
nsstat:      20041023 AA
nslastaa:    20041023
nserver:     NS2.DNS.BR
nsstat:      20041023 AA
nslastaa:    20041023
remarks:     These addresses have been further assigned to Brazilian users.
```

[16] ibid.

```
remarks:     Contact information can be found at the WHOIS server located
remarks:     at whois.registro.br and at http://whois.nic.br
created:     19950104
changed:     20020902

nic-hdl:     CGB
person:      Comite Gestor da Internet no Brasil
e-mail:      blkadm@NIC.BR
address:     Av. das Na??es Unidas, 11541, 7? andar
address:     04578-000 - S?o Paulo - SP
country:     BR
phone:       +55 19 9119-0304 []
created:     20020902
changed:     20020902
```

Finally, **194.78.59.253** was one of the most active nmap scanners during this time period, scanning nine internal hosts between 3:08 AM and 6:43 PM on 5/19. Unfortunately little can be determined from the packet information since nmap crafts almost every field, but it is likely a Unix host since nmap, while available on Windows, is far more commonly found on Linux and other Unix systems. This IP address has no reverse DNS mapping, but a WHOIS query reveals that it is an IP from the Skynet network in Belgium:

```
# ARIN WHOIS database, last updated 2004-10-23 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
% This is the RIPE Whois secondary server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/db/copyright.html

inetnum:      194.78.0.0 - 194.78.255.255
org:          ORG-BS2-RIPE
netname:      BE-SKYNET-960213
descr:        PROVIDER
descr:        Skynet Belgium
country:      BE
admin-c:      BIEC1-RIPE
tech-c:       BIEC1-RIPE
status:       ALLOCATED PA
mnt-by:       RIPE-NCC-HM-MNT
mnt-lower:    SKYNETBE-MNT
changed:      hostmaster@ripe.net 19960213
changed:      hostmaster@ripe.net 19980916
changed:      hostmaster@ripe.net 19990301
changed:      hostmaster@ripe.net 20040830
source:       RIPE

route:        194.78.0.0/16
descr:        SKYNETBE-CUSTOMERS
origin:       AS5432
notify:       noc@skynet.be
mnt-by:       SKYNETBE-MNT
changed:      jef@interpac.be 19960506
changed:      jfs@skynet.be 19990420
source:       RIPE

organisation: ORG-BS2-RIPE
org-name:     Belgacom Skynet
org-type:     LIR
address:      Belgacom Skynet SA/NV
              Rue Carli 2
              B-1140 Brussels
              Belgium
phone:        +3225407507
fax-no:       +3225135425
e-mail:       ripe@skynet.be
admin-c:      JFS1-RIPE
```

```
admin-c:      PDH16-RIPE
admin-c:      PG471-RIPE
admin-c:      PD448-RIPE
admin-c:      MN1190-RIPE
mnt-ref:      SKYNETBE-MNT
mnt-ref:      RIPE-NCC-HM-MNT
mnt-by:       RIPE-NCC-HM-MNT
changed:      hostmaster@ripe.net 20040415
changed:      bitbucket@ripe.net 20040830
changed:      bitbucket@ripe.net 20040830
source:       RIPE

role:         Belgacom Internet Expertise Center
address:      Belgacom SA de droit public
address:      ANS/ROC/RNO/IEC - Batiment TGX
address:      Boulevard du Roi Albert II, 27
address:      B-1030 Bruxelles
address:      Belgium
phone:        +32 2 202-4111
fax-no:       +32 2 203-6593
e-mail:       noc@skynet.be
admin-c:      MN1190-RIPE
admin-c:      PD448-RIPE
tech-c:       PDH16-RIPE
tech-c:       NV179-RIPE
tech-c:       SVDS1-RIPE
tech-c:       PD756-RIPE
tech-c:       PG471-RIPE
nic-hdl:      BIEC1-RIPE
remarks:      ----------------------------------------
remarks:      Network problems to: noc@skynet.be
remarks:      Peering requests to: peering@skynet.be
remarks:      Abuse notifications to: abuse@skynet.be
remarks:      abuse requests sent to another address
remarks:      will be ignored.
remarks:      ----------------------------------------
notify:       noc@skynet.be
mnt-by:       SKYNETBE-MNT
changed:      jfs@skynet.be 20040806
source:       RIPE
```

5. Correlations

Under most circumstances, the alert histories available at the Internet Storm Center (isc.sans.org) are particularly useful in correlating local attacks with more global trends. Unfortunately, however, the ISC doesn't keep port histories as far back as May 2002. More detailed correlation information can be found in the relevant analyses above, but following are a few notes regarding these attacks as viewed by other GCIA candidates.

These same raw logfiles, along with many others, have been pored over by a large number of other GCIA students, and several of those analyses have dealt with the same attacks from the same or different alert files. The Q Trojan was ably discussed by Craig Baltes[17], Al Maslowski-Yerges[18], and Rob McBee[19], among others. Formmail is a perennial favorite subject for analysis, described in at least several dozen practicals, including Thomas Harbour[20] and Barbara Morgan[21].

[17] http://www.giac.org/practical/GCIA/Craig_Baltes_GCIA.doc
[18] http://www.giac.org/practical/GCIA/Al_Maslowski-Yerges_GCIA.pdf
[19] http://www.giac.org/practical/GCIA/Rob_McBee_GCIA.pdf
[20] http://www.giac.org/practical/GCIA/Thomas_Harbour_GCIA.pdf
[21] http://www.giac.org/practical/GCIA/Barbara_Morgan_GCIA.doc

Strangely enough, it appears that nobosy has discussed the FW-1 DoS vulnerability in a GCIA practical exam. The relevant Snort rule was mentioned in several, but always in the context of a bad *TCP* packet, not UDP. This is likely because this attack is rare, and because the detect discussed above was more than likely a false positive. Nonetheless, as I argued above, it is worth analyzing if only for the danger that even an innocent port 0 packet might cause to a Checkpoint firewall.

6. Internal Dangers or Anomalies

There are two main internal anomalies that deserve some follow-up. First, as discussed in Detect 1 above, the list of Q Trojan destination addresses might not be a random scan for hosts, but in fact a command to already-infected machines. The systems corresponding to these IP addresses should be carefully checked to verify that none of them are in fact running the 'qd' process.

Second, in performing this analysis I noticed a large amount of unusual http traffic recorded from 78.37.212.28. All, or nearly all, http packets from this source appear to be doubled, albeit imperfectly. First a packet comes from this host with a TTL of 128, then less than a second later an almost identical packet is seen, but with a TTL of 240, an IP ID of 0 (no matter what the original ID was), usually a TOS (type of service) of 0x10, and a different sequence number. The final byte in the packet is also always truncated. I've been entirely unable to come up with a consistent explanation for this behavior, though on its own it appears harmless. This IP address was also by far the highest-trafficked IP in the alerts file, with a large amount of Gnutella and MSN Messenger traffic as well as Hitbox and Gator spyware. As such, this system deserves a further look.

7. Defensive Recommendations

I have compiled a list of recommendations for the network and hosts at Unseen University, divided into three main categories: Host, Firewall, and IDS.

Host

- For hosts running network services, configure them to give away less information. I was able to find out a great deal about the University webserver because it volunteers a tremendous amount of information with every response it sends. This information can be extremely useful to an attacker, and should be restricted as much as possible.
- Install and configure a host-based firewall on servers and workstations. Linux machines can use iptables, while Windows servers and workstations have a number of commercial and free software available such as Black Ice and Zone Alarm. Use the recommended firewall settings below to configure these host-based defenses.
- Verify that workstations and servers are running the latest software and OS versions and have up-to-date virus protection. The last and best defense against any network attack is to be running a software or OS version that isn't susceptible to that attack.
- In a university environment the previous requirement can be impossible for students' computers. In this case, carefully segregate these network

segments from the rest of the university's network to minimize their access to critical systems, and pay close attention to IDS alerts coming from those segments.

Firewall

- At a minimum, the firewall should be configured to drop invalid packets, both from the inside and the outside. These include:
  - Packets with invalid source addresses: private ranges (192.168.0.0/16, 172.16.0.0/24, 10.0.0.0/8), network addresses (e.g. 204.60.0.0), and broadcast addresses (e.g., 255.255.255.255)
  - Externally sourced packets with internal source addresses. Your firewall should not accept packets from the outside that claim to originate from inside your network.
  - Packets that are somehow 'broken' and not RFC-compliant: bad checksum, bad flag combinations (e.g., SYN and FIN on a single packet), invalid fragmentation, etc.
  - Otherwise invalid packets: source or destination port of 0, unknown protocol (not TCP, UDP, ICMP, or any other protocols that you are known to be using like GRE or IGRP)
- While it is common to use a firewall as a default-allow device, permitting everything except for known bad traffic, please consider making your firewall default-deny, at least for inbound traffic. By allowing only specified traffic to specified hosts, you will never be surprised by traffic coming in to unexpected hosts on obscure ports. If for technical or political reasons this is not possible, then your job will be much more difficult as you will have to carefully monitor all traffic and implement blocking for the network connections that prove to be illegitimate or dangerous.
- Likewise, I advise at least a limited default-deny policy on outbound traffic. If academic freedom and research concerns preclude a policy as draconian as this, then I suggest blocking at least the following traffic.
  - Known bad ports, like 31337 (Back Orifice) and 12345 (Netbus).
  - If network congestion or intellectual property laws are a concern, strongly consider blocking P2P application port ranges, like 6346 (Gnutella) and 1214 (Kazaa). Unfortunately Kazaa and Gnutella are able to port-hop until it finds an open port, however, so you'd probably need an application-level filtering device to fully detect and block this traffic. Closing the default ports for these applications is a start, however.

IDS

- Tune Snort ruleset. Make a list of all available services on each host, and its OS, and configure Snort accordingly. This will cut down on false positives, and reduce the processor load on your IDS. For instance, why perform checks for Windows worms such as Code Red on traffic for a webserver running Linux?

- Consider setting up several IDS sensors on different network wires. This will allow you to customize the rulesets on each sensor to best match the monitoring needs of each network segment. You can then use an IDS aggregator like ACID to collect and analyze the alerts coming from the separate sensors.

III. Analysis Process

I used an Apple Powerbook running Mac OS X 10.3 for the analysis of these logs and to compose the present report. Initial analysis of the packets was performed by a cursory glancethrough using tcpdump piped through less; much more detail and high-level statistics were obtained using a Perl script I custom wrote for the purpose.

This script uses the Perl NetPacket and Net::PCAP packages to read and sort through the vast sea of data, then print a large number of useful statistics in CSV format. (See Appendix A for a source listing.) These statistics enabled me to focus my efforts upon the most promising source and destination IP addresses and ports, and provided a first look at the overall network traffic profile.

As a side note, there appears to be a limitation on the time resolution detail on the Snort IDS that generated these logs. Past the second decimal point, all timestamps end in 4488. At first I noticed this in one detect and thought it was a part of the attack, but then I had a look at the rest of the packets and noticed that every one of them had these timestamps.

I attempted to use Snort to duplicate the alerts that must have been triggered on each of these packets. Unfortunately, however, none of the current Snort rules triggered for most of the packets in the four days I chose to analyze, so I was left to my own devices to determine why each packet was tagged as unusual and logged.

Eventually I was able to get Snort to properly evaluate the majority of these packets; by using the Netdude packet crafting tool to fix the checksums I created a capture file that Snort was glad to produce alerts for. Even then, however, not all of the packets triggered alerts, so the motivation of the IDS remains obscure even now for some of the detects. I am not sure why Snort did not alert for these packets; perhaps the older Snort ruleset was more prone to false positives than is the current one.

Once Snort had created its log file, I correlated my own analyses with the Snort alerts and found a high degree of agreement, a few surprises, and as mentioned above a few unexpected omissions. (Some packets were almost certainly triggered by the ‘zone transfer’ Snort rule, for instance, but were not picked up as such by my own Snort scan.) For most of the alerts generated, I looked at the rule descriptions and references on the snort site in an attempt to understand why the alerts were generated and whether or not they were likely to be false positives. From the remaining list of likely-accurate detects, I chose the three that seemed the most important for further analysis in the ‘Three Detects’ section.

Finally, because these logs are over two years old they have already been used and picked clean by several generations of GCIA students, and several versions of the GCIA practical. As such, just about everything of interest in these logs has already been analyzed within an inch of its life, and these analyses often show up as the first or

second result in Google. I have done my best to perform my own analyses, but it is inevitable that this paper be colored by these previous excellent works.

Appendix A. Perl source listing

```
#!/usr/bin/perl

use Net::PCAP;
use NetPacket::Ethernet qw(:strip);
use NetPacket::IP;
use NetPacket::TCP;
use NetPacket::UDP;

# functions

sub tcp_decode {

    my ($pkt) = @_;
    $tcp_packet = NetPacket::TCP->decode($pkt->{data});

    # populate TCP hashes

    my $tcp_length = $pkt->{len} - $pkt->{hlen} - $tcp_packet->{hlen}; # ouch!

    $tcp_sources{$pkt->{src_ip}}{count}++;
    $tcp_sources{$pkt->{src_ip}}{data} += $tcp_length;
    $tcp_destinations{$pkt->{dest_ip}}{count}++;
    $tcp_destinations{$pkt->{dest_ip}}{data} += $tcp_length;

    # populate ALL hashes
    $all_sources{$pkt->{src_ip}}{count}++;
    $all_sources{$pkt->{src_ip}}{data} += $tcp_length;
    $all_destinations{$pkt->{dest_ip}}{count}++;
    $all_destinations{$pkt->{dest_ip}}{data} += $tcp_length;

    # port hashes
    $source_ports{$tcp_packet->{src_port} . 'T'}{count}++;
    $source_ports{$tcp_packet->{src_port} . 'T'}{data} += $tcp_length;
    $dest_ports{$tcp_packet->{dest_port} . 'T'}{count}++;
    $dest_ports{$tcp_packet->{dest_port} . 'T'}{data} += $tcp_length;

}

sub udp_decode {

    my ($pkt) = @_;
    $udp_packet = NetPacket::UDP->decode($pkt->{data});

    # populate UDP hashes
    $udp_sources{$pkt->{src_ip}}{count}++;
    $udp_sources{$pkt->{src_ip}}{data} += $udp_packet->{len};
    $udp_destinations{$pkt->{dest_ip}}{count}++;
    $udp_destinations{$pkt->{dest_ip}}{data} += $udp_packet->{len};

    # populate ALL hashes
    $all_sources{$pkt->{src_ip}}{count}++;
    $all_sources{$pkt->{src_ip}}{data} += $udp_packet->{len};
    $all_destinations{$pkt->{dest_ip}}{count}++;
    $all_destinations{$pkt->{dest_ip}}{data} += $udp_packet->{len};

    # port hashes
    $source_ports{$udp_packet->{src_port} . 'U'}{count}++;
    $source_ports{$udp_packet->{src_port} . 'U'}{data} += $udp_packet->{len};
    $dest_ports{$udp_packet->{dest_port} . 'U'}{count}++;
    $dest_ports{$udp_packet->{dest_port} . 'U'}{data} += $udp_packet->{len};

}

# sorting functions

sub byip {
    ($a1, $a2, $a3, $a4) = split /\./, $a;
```

```
    ($b1, $b2, $b3, $b4) = split /\./, $b;

    $a1 <=> $b1 or $a2 <=> $b2 or $a3 <=> $b3 or $a4 <=> $b4;
}

# use these files

@file_list = qw(2002.4.19 2002.4.20 2002.4.21 2002.4.22);

foreach $file (@file_list) {

    # open file
    $pcap_object = Net::Pcap::open_offline("$file", \$err);
    unless (defined $pcap_object) {
        die("Unable to open file! - ", $err);
    }

    while ($packet = Net::Pcap::next($pcap_object, \%header)) {
        $ip_object = NetPacket::IP->decode(eth_strip($packet));

        # choose the decoder

        if ($ip_object->{proto} eq 6) { tcp_decode($ip_object); }
        elsif ($ip_object->{proto} eq 17) { udp_decode($ip_object); }
        else { die "Unexpected protocol found: ", $ip_object->{proto}; }
    }

    Net::Pcap::close($pcap_object);
}

# print statistics

print "TCP\n";

print "SOURCES\n";
foreach $ip_address (sort byip keys %tcp_sources) {
    print $ip_address, ",", $tcp_sources{$ip_address}{count}, ",",
$tcp_sources{$ip_address}{data}, "\n";
}

print "\nDESTINATIONS\n";
foreach $ip_address (sort byip keys %tcp_destinations) {
    print $ip_address, ",", $tcp_destinations{$ip_address}{count}, ",",
$tcp_destinations{$ip_address}{data}, "\n";
}

print "\n\n";

print "UDP\n";

print "SOURCES\n";
foreach $ip_address (sort byip keys %udp_sources) {
    print $ip_address, ",", $udp_sources{$ip_address}{count}, ",",
$udp_sources{$ip_address}{data}, "\n";
}

print "\nDESTINATIONS\n";
foreach $ip_address (sort byip keys %udp_destinations) {
    print $ip_address, ",", $udp_destinations{$ip_address}{count}, ",",
$udp_destinations{$ip_address}{data}, "\n";
}

print "\n\n";

print "ALL\n";
print "SOURCES\n";
foreach $ip_address (sort byip keys %all_sources) {
    print $ip_address, ",", $all_sources{$ip_address}{count}, ",",
$all_sources{$ip_address}{data}, "\n";
}
```

```
print "\nDESTINATIONS\n";
foreach $ip_address (sort byip keys %all_destinations) {
    print $ip_address, ",", $all_destinations{$ip_address}{count}, ",",
$all_destinations{$ip_address}{data}, "\n";
}

print "\n\nPORTS";
print "\nSOURCE\n";
foreach $port (sort { $source_ports{$b} <=> $source_ports{$a} } keys %source_ports) {
    print $port, ",", $source_ports{$port}{count}, ",", $source_ports{$port}{data}, "\n";
}

print "\nDESTINATION\n";
foreach $port (sort { $dest_ports{$b} <=> $dest_ports{$a} } keys %dest_ports) {
    print $port, ",", $dest_ports{$port}{count}, ",", $dest_ports{$port}{data}, "\n";
}
```

References

Baltes, Craig. “GCIA Certified Intrusion Analyst (GCIA) Practical Assignment, Version 3.2”, http://www.giac.org/practical/GCIA/Craig_Baltes_GCIA.doc

Checkpoint Knowledge Base, “TCP/UDP port 0 verifications performed by VPN-1/FireWall-1 NG with "any any any accept" rule”, https://secureknowledge.checkpoint.com/sk/public/idsearch.jsp?id=sk27109

Gordon, Les. “What is the Q Trojan?”, http://www.sans.org/resources/idfaq/qtrojan.php

Harbour, Thomas. “GCIA Certified Intrusion Analyst (GCIA) Practical Assignment, Version 3.4”, http://www.giac.org/practical/GCIA/Thomas_Harbour_GCIA.pdf

IEEE OUI codes, http://standards.ieee.org/regauth/oui/oui.txt

Malikai, “FW1 Port 0 UDP DoS”, Bugtraq 9 August 2000, http://www.securityfocus.com/archive/1/23615

Maslowski-Yerges, Al. “GCIA Certified Intrusion Analyst (GCIA) Practical Assignment, Version 3.3”, http://www.giac.org/practical/GCIA/Al_Maslowski-Yerges_GCIA.pdf

McBee, Rob. “GIAC GCIA Practical (version 3.3)”, http://www.giac.org/practical/GCIA/Rob_McBee_GCIA.pdf

Mogul, J.C. “Broadcasting Internet Datagrams” (RFC 919), ftp://ftp.rfc-editor.org/in-notes/rfc919.txt

Morgan, Barbara. “GCIA Certified Intrusion Analyst (GCIA) – Practical Assignment Version 3.1”, http://www.giac.org/practical/GCIA/Barbara_Morgan_GCIA.doc

Northcutt, Stephen and Novak, Judy. Network Intrusion Detection, 3rd Edition. Boston: New Riders, 2003.

SecurityFocus, “Top Attacks for the 1st Quarter 2002”, http://www.securityfocus.com/corporate/research/top10attacks_q1_2002.shtml

Spitzner, Lance. “Understanding the FW-1 State Table”, http://www.spitzner.net/fwtable.html