University Security Audit

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 4.1
Justin Snyder
March 7, 2005

Abstract

The university has requested that a security audit of their intrusion detection logs be conducted. This assignment includes an assessment of intrusion detection logs over a five day period. The paper has been divided into three different sections: (1) an executive summary that highlights the findings and recommendations; (2) the detailed analysis; and (3) the analysis process followed during the assignment.

Table of Contents

Part I - Executive Summary

A review was conducted of the university's intrusion detection system (IDS) logs from the five-day period dating February 16, 2004 to February 20, 2004. Inasmuch as this review only included the IDS logs, further investigation of potentially compromised machines is warranted.

During the five-day period, over six million IDS logs were generated. Approximately 99% of the logs were the result of port scanning. Port scanning is the act of connecting to ports on computers to see which ports are listening. These log entries are commonly associated with attackers identifying points of entry into a computer, but do not indicate a compromise. Consequently, the review primarily focused on the alert logs (over 61,000), which have a higher probability of identifying compromised machines on the university's network. The remaining logs, including the port scan logs, were analyzed when researching the alerts.

Many of the alerts could be considered informational alerts, meaning that the alerts did not indicate that a system was compromised. In addition, many alerts could also be considered false positives; alerts that indicate a possible intrusion when the activity that caused the alert was legitimate.

Although only a few of the alert types were investigated in detail, several computers on the university's network appear to be compromised. A list of machines that have been potentially compromised is included in Appendix B. Based on the machine names, it appears that several important servers have been compromised, including some university Web servers and Windows domain controllers. I recommend that these machines be investigated to determine whether or not they have been compromised. In addition, the remaining alerts that were not investigated should be analyzed by the university's security team to ensure that additional machines on the university's network have not been compromised.

I have three recommendations that should provide some immediate improvement to the security of the university's network. First, I recommend that the university develop an incident response team (if one does not already exist) that will respond to security-related incidents. The members of the team should be adequately trained to handle incidents and should be provided training opportunities annually to stay abreast of current security risks and trends. In addition, I recommend that the university implement a firewall or reconfigure the existing firewall to improve the perimeter security of the university's network. The firewall should be configured to only allow network traffic that is required, enforce the university's written security policies, and segregate critical university computers from the rest of the network. Finally, I recommend that the university re-evaluate the IDS architecture to ensure IDS sensors have been placed in the appropriate locations. The IDS sensors should also be reconfigured to reduce false positives and the IDS rulebases should be updated. Updating the rulebase will also help reduce false positives and will ensure that alerts will be generated for the latest vulnerabilities.

The details of the analysis performed and additional recommendations are included in the following section of the paper.

Part II – Detailed Analysis

1 Analysis dates

I used the following alert, OOS, and scan logs from http://isc.sans.org/logs for the security audit.

Alert	OOS	Scan
alert.040216.gz	Oss_report_040216.gz	scans.040216.gz
alert.040217.gz	Oss_report_040217.gz	scans.040217.gz
alert.040218.gz	Oss_report_040218.gz	scans.040218.gz
alert.040219.gz	Oss_report_040219.gz	scans.040219.gz
alert.040220.gz	Oss_report_040220.gz	scans.040220.gz

It should be noted that the OOS log files I selected (2/16/04 – 2/20/04) contained data from 2/20/04 to 2/24/04, as opposed to data from the 2/16/04 to 2/20/04 timeframe.

2 Network topology

Using the assumption that the MY.NET.X.X network is the university's network, I searched all logs to identify hosts on the university's network. It was noted that no MY.NET.X.X hosts were identified in the scan logs. Further analysis was performed which identified that the MY.NET.X.X network was the 130.85.X.X network represented in the scan logs.

Analyzing all source and destination addresses in all logs for hosts on the MY.NET.X.X network identified 15,737 unique hosts. Although 15,737 unique hosts were identified in the logs, this may not represent the actual number of hosts on the university's network. There were 394 hosts identified in the logs at least 100 times, while 15,343 hosts were identified in the logs less than 100 times. For example, some hosts may exist but did not trigger an IDS alert. In addition, some hosts identified in the logs as destination addresses may be a result of a port scan of the university's network. If a host does not exist at a specific network address and that network address is scanned, the network address will appear in the IDS logs. Furthermore, hosts identified as the source address in the logs may actually be spoofed addresses.

Of the 15,737 hosts identified, 468 were the source address in log entries, 15,736 were the destination address in log entries, and 467 hosts were identified as both a source and destination address in the logs. MY.NET.214.190 (resnet-214-190.resnet.UMBC.EDU) is the only host that was a source address in log entries but never a destination address. This host was identified in 17 OOS log entries scanning TCP port 80 (HTTP typically runs over TCP port 80) on MY.NET.24.34 and in 24 OOS log entries scanning TCP port 443 on MY.NET.12.7 (HTTPS typically runs over TCP port 443). Here are a few of the OOS log entries identified:

```
02/20-00:55:24.841257 MY.NET.214.190:32797 -> MY.NET.24.34:80
TCP TTL:61 TOS:0x0 ID:9567 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xE09C1556  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 33565 0 NOP WS: 0

02/20-00:55:55.202404 MY.NET.214.190:32824 -> MY.NET.12.7:443
TCP TTL:61 TOS:0x0 ID:61920 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x60BE2D1F  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 36600 0 NOP WS: 0
```

Next, I determined the number of log entries that exist for each destination and source port on each host on the university's network. Focusing on the hosts that appeared as destination hosts in the logs and some popular ports, I created a table that includes the destination host address, the destination port, the number of times a log entry was created for the host and port, and the host's name (if available). The host names were determined using 'nslookup', a utility used to query domain name (DNS) servers. The 'nslookup' utility can perform reverse DNS lookups to identify the hostname of a given IP address. An example query would be 'nslookup 130.85.30.4'. Here is the result of the analysis:

Destination Host	DstPort	Service	Total Instances	Hostname
MY.NET.24.47	21	FTP	87	ftp1.umbc.edu
MY.NET.70.50	21	FTP	14	ecs020pc-15.ucs.umbc.edu
MY.NET.24.27	21	FTP	11	ragnarok.umbc.edu
MY.NET.12.6	25	SMTP	3616	mxin.umbc.edu
MY.NET.34.14	25	SMTP	54	imap.cs.UMBC.EDU
MY.NET.60.17	25	SMTP	25	alumni.umbc.edu
MY.NET.110.150	25	SMTP	13	eds1.engr.UMBC.EDU
MY.NET.1.3	53	DNS	346	UMBC3.UMBC.EDU
MY.NET.1.4	53	DNS	92	UMBC4.UMBC.EDU
MY.NET.1.5	53	DNS	46	UMBC5.UMBC.EDU
MY.NET.29.3	80	HTTP	3524	bb-app4.umbc.edu
MY.NET.30.4	80	HTTP	792	lan2.umbc.edu
MY.NET.24.44	80	HTTP	641	userpages.umbc.edu
MY.NET.24.34	80	HTTP	275	www.umbc.edu
MY.NET.84.235	80	HTTP	186	
MY.NET.34.11	80	HTTP	173	web1.cs.umbc.edu
MY.NET.30.3	80	HTTP	144	lan1.umbc.edu
MY.NET.6.7	80	HTTP	104	umbc7.umbc.edu
MY.NET.5.92	80	HTTP	103	
MY.NET.6.7	110	POP3	2959	umbc7.umbc.edu
MY.NET.12.4	110	POP3	266	mail.umbc.edu
MY.NET.24.8	119	NNTP	57	news.umbc.edu
MY.NET.30.3	389	LDAP	16	lan1.umbc.edu
MY.NET.30.4	389	LDAP	14	lan2.umbc.edu
MY.NET.75.13	389	LDAP	14	chpdm.umbc.edu

MY.NET.29.13	389	LDAP	13	tcl1.cl.umbc.edu
MY.NET.11.11	389	LDAP	12	dc1test.adtest.UMBC.EDU
MY.NET.24.49	389	LDAP	12	directory.umbc.edu
MY.NET.24.65	389	LDAP	11	fett.umbc.edu
MY.NET.30.5	389	LDAP	10	
MY.NET.30.7	389	LDAP	10	
MY.NET.24.7	389	LDAP	10	sluisvan.ucs.umbc.edu
MY.NET.12.7	443	HTTPS	49	webauth.umbc.edu
MY.NET.24.74	443	HTTPS	24	webmail.umbc.edu

I performed the same steps for MY.NET.X.X source addresses and ports to create the following table:

Source Host	SrcPort	Service	Total Instances	Hostname
MY.NET.29.3	80	HTTP	3659	bb-app4.umbc.edu
MY.NET.24.44	80	HTTP	67	
MY.NET.24.34	80	HTTP	41	
MY.NET.5.20	80	HTTP	15	centrelearn.umbc.edu
MY.NET.11.7	137	NETBIOS Name Service	1033	dc2.ad.umbc.edu
MY.NET.151.85	137	NETBIOS Name Service	836	
MY.NET.75.13	137	NETBIOS Name Service	495	
MY.NET.11.6	137	NETBIOS Name Service	56	dc1.ad.UMBC.EDU

Using the data in these two charts, we can develop ideas about the purpose of some of the hosts. For example, 87 log entries existed in which port 21 on MY.NET.24.47 was the destination of a packet. Since the name of MY.NET.24.47 is ftp1.ubmc.edu, we can make a reasonable assumption that this is one of the university's FTP servers. In addition, 3,616 log entries existed where port 25 on MY.NET.12.6 was the destination of a packet. Using the name of this host, mxin.umbc.edu, we can make the assumption that this is a university mail (SMTP) server. Port 53 on MY.NET.1.3, MY.NET.1.4, and MY.NET.1.5 was the destination port in 484 log entries. Performing a whois query (via http://www.networksolutions.com/en_US/whois/index.jhtml) verified that these were the DNS servers for the university. Multiple hosts appear to be running a Web server (TCP ports 80 and 443). The university's primary Web server (MY.NET.24.34; www.umbc.edu) was identified in the logs, along with what appears to be two sites, MY.NET.12.7 (webauth.umbc.edu) and MY.NET.24.74 (webmail.umbc.edu) using HTTPS.

In addition to looking at the source and destination addresses and ports, I examined the alert log descriptions to see if I could identify any additional information about hosts on the network and to help solidify some conclusions I had previously made. The first alert log description of interest is "MY.NET.30.4 activity". Analysis of all alert logs with a description of "MY.NET.30.4 activity" identified 11,115 log entries. Multiple destination ports appeared in the logs, meaning the rule that generated the log entries probably logged all packets destined for MY.NET.30.4. Another alert log description of interest is "MY.NET.30.3 activity". Analysis of all alert logs with this description identified 5,621 log entries. Again, multiple destination ports appeared in the logs for these entries. Because it appears alerts are generated for every packet that is targeted at MY.NET.30.3 and MY.NET.30.4, I'm guessing these machines are used as honeypots. A honeypot is a host (or network) with known vulnerabilities that are used to study attackers' behavior and to draw attention away from other potential targets. Here are some example log entries for the "activity" logs:

```
02/16-00:00:51.738166  [**] MY.NET.30.4 activity [**] 12.21.173.176:3142 -> MY.N
ET.30.4:524
02/16-19:27:25.663905  [**] MY.NET.30.4 activity [**] 151.196.24.31:3665 -> MY.N
ET.30.4:51443
02/16-23:06:30.104583  [**] MY.NET.30.3 activity [**] 68.55.62.79:2093 -> MY.NET
.30.3:524
02/17-03:39:14.718084  [**] MY.NET.30.3 activity [**] 147.162.167.49:4647 -> MY.
NET.30.3:20168
```

Additional analysis of the alert log descriptions disclosed some entries that included the string of "HelpDesk". Here are the ten log entries identified:

```
02/17-11:55:22.965264  [**] External FTP to HelpDesk MY.NET.70.50 [**] 172.178.0
.159:59344 -> MY.NET.70.50:21
02/17-11:55:23.609177  [**] External FTP to HelpDesk MY.NET.70.50 [**] 172.178.0
.159:62672 -> MY.NET.70.50:21
02/17-11:55:24.957246  [**] External FTP to HelpDesk MY.NET.70.50 [**] 172.178.0
.159:56528 -> MY.NET.70.50:21
02/17-11:55:25.605003  [**] External FTP to HelpDesk MY.NET.70.50 [**] 172.178.0
.159:55248 -> MY.NET.70.50:21
02/17-11:55:27.546286  [**] External FTP to HelpDesk MY.NET.70.50 [**] 172.178.0
.159:35536 -> MY.NET.70.50:21
02/17-11:55:28.886732  [**] External FTP to HelpDesk MY.NET.70.50 [**] 172.178.0
.159:37328 -> MY.NET.70.50:21
02/17-11:55:30.846816  [**] External FTP to HelpDesk MY.NET.70.50 [**] 172.178.0
.159:51408 -> MY.NET.70.50:21
02/17-11:55:32.155852  [**] External FTP to HelpDesk MY.NET.70.50 [**] 172.178.0
.159:54480 -> MY.NET.70.50:21
02/20-05:42:09.027685  [**] External FTP to HelpDesk MY.NET.53.29 [**] 80.54.18.
249:3144 -> MY.NET.53.29:21
02/20-06:17:43.529156  [**] External FTP to HelpDesk MY.NET.70.49 [**] 80.54.18.
249:3587 -> MY.NET.70.49:21
```

It appears there are at least three hosts used for the "HelpDesk" function, all running the FTP service.

Further analysis disclosed some potential TFTP servers. Here are the TFTP alerts:

```
02/16-09:51:06.660885  [**] TFTP - External UDP connection to internal tftp server [**]
MY.NET.6.48:69 -> 68.55.192.221:64336
```

```
02/16-13:00:51.995605  [**] TFTP - Internal TCP connection to external tftp server [**]
MY.NET.60.39:50441 -> 66.160.63.199:69
02/16-13:26:52.805357  [**] TFTP - Internal UDP connection to external tftp server [**]
24.15.191.203:69 -> MY.NET.53.42:6257
02/17-17:17:59.189706  [**] TFTP - External UDP connection to internal tftp server [**]
66.93.118.125:16242 -> MY.NET.82.118:69
02/17-19:57:09.170101  [**] TFTP - External UDP connection to internal tftp server [**]
61.135.144.203:56833 -> MY.NET.53.159:69
02/18-01:43:47.487072  [**] TFTP - External TCP connection to internal tftp server [**]
160.218.214.105:36552 -> MY.NET.70.90:69
02/18-01:43:47.489314  [**] TFTP - External TCP connection to internal tftp server [**]
MY.NET.70.90:69 -> 160.218.214.105:36552
02/18-21:52:08.286877  [**] TFTP - Internal UDP connection to external tftp server [**]
216.249.81.112:69 -> MY.NET.98.53:3617
02/19-14:39:45.464350  [**] TFTP - Internal UDP connection to external tftp server [**]
63.68.196.38:69 -> MY.NET.1.5:53
02/19-22:56:37.134837  [**] TFTP - External TCP connection to internal tftp server [**]
204.152.186.189:35924 -> MY.NET.25.73:69
02/19-22:56:37.135026  [**] TFTP - External TCP connection to internal tftp server [**]
MY.NET.25.73:69 -> 204.152.186.189:35924
02/20-10:41:27.978516  [**] TFTP - External UDP connection to internal tftp server [**]
63.208.107.43:1092 -> MY.NET.153.157:69
```

It appears that MY.NET.6.48 (hfs1.afs.umbc.edu), MY.NET.82.118 (oit-82-118.pooled.umbc.edu), MY.NET.53.159 (ecs333pc29.ucslab.umbc.edu), MY.NET.70.90 (oit004lj1.ucs.umbc.edu), MY.NET.25.73 (mx8in.umbc.edu), and MY.NET.153.157 (libstkpc15.libpub.umbc.edu) are running the TFTP service, a service used to transfer files without requiring authentication. The communication between MY.NET.70.90 and 160.218.214.105 is most likely the result of a file being transferred between the hosts. We do see another alert between these two hosts after the TFTP communication:

```
02/18-01:44:32.531260  [**] SUNRPC highport access! [**] 160.218.214.105:36552 ->
MY.NET.70.90:32771
```

These are the only 3 alert logs between these two hosts (the 2 TFTP logs and the SUNRPC log). There are no additional logs (OOS or scan logs) that involve 160.218.214.105. Based on the logs, my guess is that the attacker attempted to TFTP a file most likely containing some type of exploit code. The connection to the sunrpc port (32771) could have been an exploit attempt on the fam (file alteration monitor) RPC service that has been susceptible to buffer overflow attacks in the past (CVE-1999-0059; Bugtraq ID 353) or the attacker could have been using the fam service to see if the file transferred over TFTP was accepted by MY.NET.70.90. The 160.218.214.105 address is from an ISP, Eurotel Praha, in the Czech Republic. This was identified by performing a whois query on ripe.net (http://www.ripe.net/whois).

The next two alert logs show the communication between MY.NET.25.73 (mx8in.umbc.edu), an SMTP server, and 204.152.186.189, a host from Internet Systems Consortium, Inc. (according to a whois query on networksolutions.com):

```
02/19-22:56:37.134837  [**] TFTP - External TCP connection to internal tftp server [**]
204.152.186.189:35924 -> MY.NET.25.73:69
02/19-22:56:37.135026  [**] TFTP - External TCP connection to internal tftp server [**]
MY.NET.25.73:69 -> 204.152.186.189:35924
```

What follows these two log entries is also interesting:

```
02/19-23:09:29.185156  [**] SUNRPC highport access! [**] 204.152.186.189:58571 ->
MY.NET.25.73:32771
02/19-23:09:31.408978  [**] SUNRPC highport access! [**] 204.152.186.189:58608 ->
MY.NET.25.73:32771
02/19-23:09:31.488416  [**] SUNRPC highport access! [**] 204.152.186.189:58608 ->
MY.NET.25.73:32771
02/19-23:09:31.490694  [**] SUNRPC highport access! [**] 204.152.186.189:58608 ->
MY.NET.25.73:32771
02/19-23:09:33.454904  [**] SUNRPC highport access! [**] 204.152.186.189:58610 ->
MY.NET.25.73:32771
02/19-23:09:38.494553  [**] SUNRPC highport access! [**] 204.152.186.189:58617 ->
MY.NET.25.73:32771
02/19-23:09:39.663427  [**] SUNRPC highport access! [**] 204.152.186.189:58618 ->
MY.NET.25.73:32771
02/19-23:09:39.663438  [**] SUNRPC highport access! [**] 204.152.186.189:58618 ->
MY.NET.25.73:32771
02/19-23:09:39.663603  [**] SUNRPC highport access! [**] 204.152.186.189:58618 ->
MY.NET.25.73:32771
```

It was also noted that 204.152.186.189 performed what looks like a SYN scan of MY.NET.25.73 (23,570 log entries). Here are the first and last log entries of that scan from the scan logs:

```
Feb 19 22:32:15 204.152.186.189:35924 -> 130.85.25.73:23191 SYN ******S*
Feb 19 22:59:59 204.152.186.189:35927 -> 130.85.25.73:59493 SYN ******S*
```

It would be a good idea to investigate if a sunrpc service is running on port 32771 on MY.NET.25.73 and if this host has been compromised.

3 Detects

During the five day period, exactly 613,000 alert log entries were recorded of which 551,702 of the logs were “portscan” logs. Because the port scans were also recorded in the scan logs, I generated a unique list of alert log entries, excluding ”portscan” entries, and the number of times that type of alert occurred over the five day period. This narrowed the list to 50 different alert types. Here is the result of the analysis:

Alert Description	Total Instances
TCP SRC and DST outside network	19406
MY.NET.30.4 activity	11114
Possible trojan server activity	7467
Incomplete Packet Fragments Discarded	5636
MY.NET.30.3 activity	5621
SMB Name Wildcard	3973
High port 65535 tcp - possible Red Worm – traffic	2590
EXPLOIT x86 NOOP	1019
Tiny Fragments - Possible Hostile Activity	926
Null scan!	741
NMAP TCP ping!	701
External RPC call	543
High port 65535 udp - possible Red Worm – traffic	488
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.	168
connect to 515 from inside	162
SMB C access	108
IRC evil - running XDCC	92
TCP SMTP Source Port traffic	86
SUNRPC highport access!	84
FTP passwd attempt	63
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	41
[UMBC NIDS] External MiMail alert	32
EXPLOIT x86 setuid 0	30
connect to 515 from outside	28
EXPLOIT x86 setgid 0	21
ICMP SRC and DST outside network	20
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	20
EXPLOIT x86 stealth noop	15
EXPLOIT NTPDX buffer overflow	11
DDOS shaft client to handler	11
FTP DoS ftpd globbing	10
SYN-FIN scan!	10
RFB - Possible WinVNC - 010708-1	9
[UMBC NIDS] Internal MiMail alert	9
External FTP to HelpDesk MY.NET.70.50	8
Attempted Sun RPC high port access	6
TFTP - External UDP connection to internal tftp server	4
TFTP - External TCP connection to internal tftp server	4
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	3
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	3
TFTP - Internal UDP connection to external tftp server	3
EXPLOIT FTP passwd retrieval retr path	2
Traffic from port 53 to port 123	1
DDOS mstream handler to client	1
External FTP to HelpDesk MY.NET.53.29	1
DDOS mstream client to handler	1
External FTP to HelpDesk MY.NET.70.49	1

TFTP - Internal TCP connection to external tftp server	1
Probable NMAP fingerprint attempt	1
[UMBC NIDS IRC Alert] K\:line'd user detected possible trojan.	1

I then reviewed these alert types to identify which ones to research. Some alerts appear to be informational (e.g., "MY.NET.30.4 activity") or different scan types (e.g., "Null scan!") that I considered low priority. I then started analyzing the higher priority alerts based upon the number of instances the alert occurred and the alert description.

3.1 Detect 1 – Possible trojan server activity

3.1.1 Description of the Detect

During the five day period, 7,467 "Possible trojan server activity" alerts were generated. This alert appears to be triggered by traffic using either TCP source or destination port 27374. TCP port 27374 is most commonly associated with SubSeven 2.1, a known trojan for the Windows platform (http://www.sans.org/resources/idfaq/subseven.php), but can also be used as an ephemeral port. Further analysis of these alerts disclosed only 152 log entries (2% of the "Possible trojan server activity" alerts) in which the other port used in the communication was greater than 1024. In the other 7,315 log entries, TCP port 27374 was most likely selected as an ephemeral port. 7,141 of those log entries were between TCP port 27374 on 169.200.215.36 (an IP address registered to First Union National Bank Corporation) and TCP port 80 on MY.NET.29.3 (bb-app4.umbc.edu).

Because of the number of log entries associated with the traffic between TCP port 27374 on 169.200.215.36 and TCP port 80 on MY.NET.29.3, I reviewed all alert, scan, and OOS logs for port 80 on MY.NET.29.3 to see if other hosts were communicating with the Web server service. A review of the scan and OOS logs disclosed 27 scan logs and one OOS log. There were two other hosts (excluding a host that only performed an NMAP TCP ping) that generated alert logs while communicating with TCP port 80 on MY.NET.29.3. Because multiple hosts communicated with TCP port 80 on MY.NET.29.3, it appears that MY.NET.29.3 is actually running a Web server (although this could not be confirmed during the writing of this paper, the Web server may have been taken down). As a result, it also appears the alerts generated by the traffic between TCP port 27374 on 169.200.215.36 and TCP port 80 on MY.NET.29.3 are false positives.

I manually started reviewing the remaining 152 log entries and researched common services running on the other ports used in the communications. Of the 152 log entries, 135 (89%) were generated by 24.86.3.160, an address belonging to Shaw Communications, Inc (according to a whois query on networksolutions.com), an ISP. It appears 24.86.3.160 was using some type of scanning utility looking for hosts listening on TCP 27374. Here are some example alert logs:

```
02/20-19:31:52.394188  [**] Possible trojan server activity [**] 24.86.3.160:3225 ->
MY.NET.190.73:27374
```

```
02/20-19:31:52.404101  [**] Possible trojan server activity [**] 24.86.3.160:3226 ->
MY.NET.190.74:27374
02/20-19:31:52.404401  [**] Possible trojan server activity [**] 24.86.3.160:3229 ->
MY.NET.190.77:27374
02/20-19:31:52.404411  [**] Possible trojan server activity [**] 24.86.3.160:3228 ->
MY.NET.190.76:27374
02/20-19:31:52.404420  [**] Possible trojan server activity [**] 24.86.3.160:3237 ->
MY.NET.190.85:27374
02/20-19:31:52.404428  [**] Possible trojan server activity [**] 24.86.3.160:3233 ->
MY.NET.190.81:27374
02/20-19:31:52.404435  [**] Possible trojan server activity [**] 24.86.3.160:3235 ->
MY.NET.190.83:27374
02/20-19:31:52.404452  [**] Possible trojan server activity [**] 24.86.3.160:3234 ->
MY.NET.190.82:27374
```

I then queried the scan logs looking for any traffic with a source or destination port of 27374. This identified 132 scan log entries, of which 129 log entries had a source IP of 24.86.3.160 (they were all SYN scan entries).

Of the remaining 20 alert log entries, MY.NET.84.235 appeared to be using some P2P file sharing programs (TCP port 4661 and 4662 are used by eMule) in five logs and MY.NET.70.210 appeared to be using KAZAA (another P2P file sharing program). For further analysis of the ports used by eMule, this is a good reference:

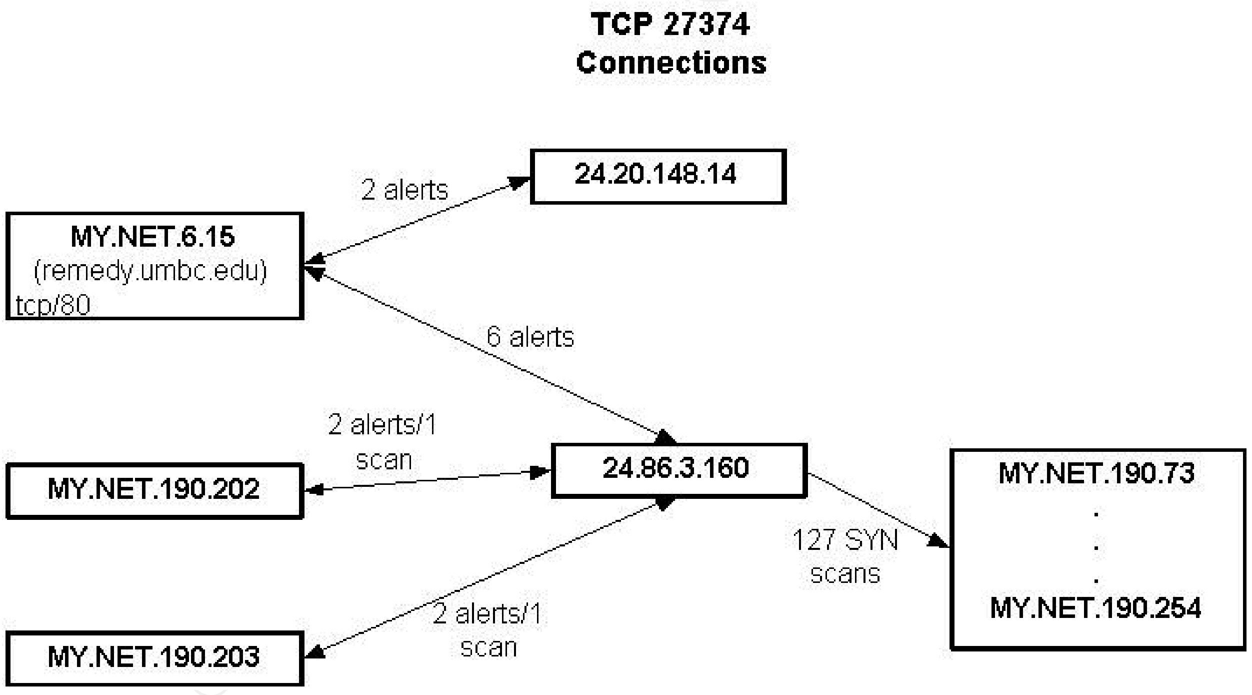
http://www.emule-project.net/home/perl/help.cgi?l=1&topic_id=122&rm=show_topic

In the remaining alert log entries, communication occurred between TCP port 2089 on 24.86.3.160 and TCP port 27374 on MY.NET.6.15 (remedy.umbc.edu). Here are the logs:

```
02/20-18:12:49.692625  [**] Possible trojan server activity [**] 24.86.3.160:2089 ->
MY.NET.6.15:27374
02/20-18:12:49.692846  [**] Possible trojan server activity [**] MY.NET.6.15:27374 ->
24.86.3.160:2089
02/20-18:12:50.194549  [**] Possible trojan server activity [**] 24.86.3.160:2089 ->
MY.NET.6.15:27374
02/20-18:12:50.194631  [**] Possible trojan server activity [**] MY.NET.6.15:27374 ->
24.86.3.160:2089
02/20-18:12:50.702391  [**] Possible trojan server activity [**] 24.86.3.160:2089 ->
MY.NET.6.15:27374
02/20-18:12:50.702488  [**] Possible trojan server activity [**] MY.NET.6.15:27374 ->
24.86.3.160:2089
```

I also found that MY.NET.6.15 is hosting a legitimate Web site, simply by browsing to the DNS name, remedy.umbc.edu, in my Web browser. This system may be compromised and may be running the SubSeven trojan. Furthermore, two more logs exist in which 24.20.148.14 (address belongs to Comcast Cable Communications, another ISP) communicated to TCP port 27374 on MY.NET.6.15. Finally, it appears that MY.NET.190.202 (wt-vpn1.umbc.edu) and MY.NET.190.203 (wt-vpn2.umbc.edu) appear to be running the SubSeven trojan and communicating with 24.86.3.160. However, these alert logs corresponded to the scans of TCP port 27374 by 24.86.3.160 (identified in the scan logs). Because these two hosts did respond to 24.86.3.160 that they were listening on TCP port 27374, it would be a good idea to investigate these hosts for the trojan.

The following link graph shows the connections from the potential attackers, 24.86.3.160 and 24.20.148.14, to TCP port 27374 on the university hosts. The bidirectional arrows indicate communication occurred between both hosts. The unidirectional arrow between 24.86.3.160 and the MY.NET.190.73 to MY.NET.190.254 address range depicts the port scan of the address range by 24.86.3.160. The port scan included MY.NET.190.202 and MY.NET.190.203, but those scan attempts are depicted on the left side of the graph because those two hosts responded to the port scan. In addition, I included the fact that MY.NET.6.15 was also running a Web server (TCP/80).



3.1.2 Reason the attack was selected

Although the "Possible trojan server activity" alert has a high probability of false positives because of the use of TCP port 27374 as an ephemeral port, this attack was selected because of the large number of alerts generated (it had the third highest total) and because of the danger the SubSeven trojan presents. The SubSeven trojan gives an attacker complete control of a compromised host.

3.1.3 Detect was generated by

The detect was generated by Snort rules that generate an alert for any traffic over TCP port 27374. The rules would look similar to these:

```
alert tcp any 27374 -> any any (msg:"Possible trojan server activity";)
alert tcp any any -> any 27374 (msg:"Possible trojan server activity";)
```

3.1.4 Probability the source address was spoofed

Because communication with the SubSeven trojan requires a full TCP connection, it is highly unlikely the source addresses were spoofed.

3.1.5 Attack mechanism

Here is a good reference about the details of the SubSeven trojan:
http://www.sans.org/resources/idfaq/subseven.php

SubSeven is a trojan that typically spreads via email attachments, but can also be packaged with downloaded software that is downloaded through P2P networks, etc. Once the trojan is executed (either via clicking the email attachment, launching the mp3 you think you just downloaded, etc.), it installs the trojan "server" component (called server.exe by default). The server then listens on TCP port 27374 by default (this is configurable). Depending on how the attacker pre-configured the trojan, the trojan will communicate back to the attacker via e-mail, ICQ, etc., the IP address of the infected machine. The attacker can then use the SubSeven client to connect to the "server" and take complete control of the infected machine.

In this detect, all traffic over TCP port 27374 generated the alerts. Because TCP port 27374 can also be used as an ephemeral port, there is a high probability that the alerts are false positives.

3.1.6 Correlations

SubSeven has evolved over the years since it was originally released on February 28, 1999. Aaron Greenlee's GSEC paper (http://www.sans.org/rr/whitepapers/malicious/958.php), although it focuses on SubSeven 2.2, provides excellent details on the different components of the trojan and its history.

I also reviewed the remaining alert logs, as well as the OOS and scan logs, to see if the potentially infected hosts (MY.NET.6.15, MY.NET.190.202, and MY.NET.190.203) were identified in any additional logs. Other than the scan logs mentioned in the "Description of the detect" section above, no other scan logs disclosed anything about the hosts in question. In addition, the OOS logs did not disclose anything. There were 10 additional alerts that included these hosts as the destination hosts, all of which were "External RPC call" alerts. These were attempts to identify any RPC services running on the university hosts by two different attackers (217.172.186.136 and 213.85.29.136), neither of which were the attackers in this detect. Consequently, these alerts do not appear to be related to this detect.

3.1.7 Evidence of active targeting

In reviewing the scan logs, 24.86.3.160 scanned 129 hosts, all of which were on the MY.NET.190.X network, for TCP port 27374 using SYN scans. These scans, as mentioned earlier, included MY.NET.190.202 and MY.NET.190.203. The scans performed by 24.86.3.160 were only for TCP port 27374. Although the attacker scanned 129 hosts, the hosts were all on a small segment of the university's network (MY.NET.190.X) and the scans were specifically for TCP port 27374. In my opinion, this appears to be active targeting.

In addition, the attacks against MY.NET.6.15 appear to be focused attacks inasmuch as the attackers did not scan the university machine for TCP port 27374. Again, I believe this is active targeting.

3.1.8 Severity

In this attack, two of the targets appeared to be workstations and one was a legitimate Web server (remedy.umbc.edu). Although this was not the Web server hosting the main university site (www.umbc.edu), it was publicly accessible. As a result, I gave a criticality score of 4. For lethality, if the attack was successful, it could result in total system compromise. Consequently, I gave a lethality score of 5. For system countermeasures, the Web server may not be running anti-virus software which is one of the primary methods of identifying the SubSeven trojan. Unless files are being uploaded to a Web server or the server is doubling as a mail server, most Web servers probably don't run anti-virus software. As a result, I gave a system countermeasure score of 2. Finally, traffic to TCP port 27374 on the Web server is most likely not required and, as a result, should be blocked by a firewall or router. However, this traffic does not appear to be blocked. Consequently, I gave a network countermeasure score of 2. Here is the final score:

Severity = (Criticality + Lethality) – (System countermeasures + Network countermeasures)

Severity = (5+5) – (2+2) = 6

3.2 Detect 2 – TCP SRC and DST outside network

3.2.1 Description of the Detect

During the five day period, over 19,000 alerts were generated for traffic between hosts that were both outside of the university's network. This could indicate address spoofing or that routing problems exist. Further analysis identified 19,290 alerts in which traffic was destined for port 80 (typically the Web server port) on 64.136.21.233 (my-eap.nyc.untd.com). According to a whois search on networksolutions.com, this address belongs to Juno Online Services, Inc. A screenshot of the Web site hosted at this address can be found in Appendix C.

All of the source addresses were from the 169.254.0.0/16 network, a class B private address block reserved for automatic private address allocation (DHCP clients configure themselves with these addresses if they cannot reach the DHCP server). These source addresses appeared in random order as well (17,080 different source addresses were identified in the attack). In addition, the source ports ranged from 1000-1999. The alerts were generated between 15:39:44 and 16:07:46 (28 minutes and 2 seconds) on February 20th, equating to almost 12 alerts per second. Here is a sample of the alerts:

```
02/20-15:45:00.505689  [**] TCP SRC and DST outside network [**]
169.254.162.208:1361 -> 64.136.21.233:80
02/20-15:45:00.537349  [**] TCP SRC and DST outside network [**]
169.254.228.209:1197 -> 64.136.21.233:80
02/20-15:45:00.553217  [**] TCP SRC and DST outside network [**]
169.254.38.81:1032 -> 64.136.21.233:80
02/20-15:45:00.857917  [**] TCP SRC and DST outside network [**]
169.254.240.80:1648 -> 64.136.21.233:80
02/20-15:45:01.097509  [**] TCP SRC and DST outside network [**]
169.254.79.115:1483 -> 64.136.21.233:80
02/20-15:45:01.097681  [**] TCP SRC and DST outside network [**]
169.254.1.244:1196 -> 64.136.21.233:80
```

I then analyzed the remaining alert log entries to see if any 169.254.X.X addresses were identified in other alerts, as well as the scan and OOS log entries. Neither additional alert log entries existed, nor any OOS log entries. Four scan log entries existed:

```
Feb 18 09:02:08 130.85.1.4:32788 -> 169.254.161.194:53 UDP
Feb 18 09:02:33 130.85.1.4:32788 -> 169.254.161.194:53 UDP
Feb 18 21:05:14 130.85.112.222:1648 -> 169.254.32.45:25 SYN ******S*
Feb 20 13:52:18 130.85.97.182:1070 -> 169.254.40.237:41170 UDP
```

None of these scan log entries appear to be related to the attack. The four scan log entries were all from hosts on the university's network to the 169.254.0.0/16 network.

Since the 169.254.0.0/16 addresses are private addresses, they could be used for Network Address Translation (NAT); however, it does not appear that the university was using these addresses for NAT. If the university was using these addresses for NAT, they would most likely be found in many other log entries.

3.2.2 Reason the attack was selected

The attack appears to be a denial of service (DoS) attack against the Web site hosted by 64.136.21.233 because of the number of packets (11-12 per second) over a short timeframe (28 minutes and 2 seconds) destined to port 80 on the target from what appears to be a large number of spoofed source addresses. Although the attack looks like a DDoS (Distributed DoS) in which multiple hosts attack a single target, this attack appears to come from a private network that does not exist on the university's network.

Because the network does not exist, the source addresses were probably spoofed from a machine on the university's network. The attack was selected because of the possible liabilities involved with a university computer attacking a remote Web site. Although it is difficult to identify the true source of the spoofed addresses, it may be possible to backtrace the spoofed packets and identify that the university is responsible for the DoS attack.

3.2.3 Detect was generated by

The detect was generated by a Snort rule that generates an alert for any traffic that does not include a university network address. The rule would look similar to this:

```
alert tcp $EXTERNAL_NET any -> $EXTERNAL_NET any (msg:"TCP SRC and DST
outside network";)
```

3.2.4 Probability the source address was spoofed

The probability that the source address was spoofed is high. In a DoS attack, the source addresses are typically spoofed to prevent the source of the attack from being denied service. This is one of the few instances in which an attacker does not want a response from the target. By spoofing the source addresses, the responses from the target will not be sent to the attacker. In addition, it appears that some sort of tool was used to perform the attack based on the fact that the source addresses were all from the same network and the source ports were from a small range (both were probably configurable options).

3.2.5 Attack mechanism

This attack was a denial of service attack against port 80, the Web server service, on 64.136.21.233. The attack was most likely performed using some type of automated tool that used random ports between 1000 and 1999 while spoofing IP addresses in random order from the 169.254.0.0/16 address block. The attack lasted just over 28 minutes from 15:39:44 to 16:07:46 on February 20th. The purpose of the attack was most likely to prevent legitimate use of the Web site hosted by 64.136.21.233.

3.2.6 Correlations

This attack looks very similar to the Blaster worm and its variants. The original Blaster worm's denial of service traffic used TCP source ports between 1000 and 1999 and was destined for port 80 on windowsupdate.com. A variant could have been trivially created to change the destination of the attack to 64.136.21.233. Here are some references explaining the details of the vulnerability that the Blaster worm attacked:

http://www.cert.org/advisories/CA-2003-20.html
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx

The vulnerability was identified by the Last Stage of the Delirium Research Group (http://lsd-pl.net/). John Van Hoogstraten's analysis of the Blaster worm can be found here (Note – at the time of this writing the URL was not working; however, John's paper had been cached by Google):

http://www.giac.org/certified_professionals/practicals/gcih/0489.php

Because the Blaster worm spreads by sending data to hosts on TCP port 135, I analyzed the alert logs for all alerts that included port 135. This analysis identified four unique hosts (in 223 alert log entries) on the university's network that had traffic sent to TCP port 135; MY.NET.190.93, MY.NET.190.95, MY.NET.190.97, and MY.NET.190.102. Here are some example logs:

```
02/20-00:02:33.467813  [**] EXPLOIT x86 NOOP [**] 172.186.175.40:2978 ->
MY.NET.190.95:135
02/20-02:37:54.983436  [**] EXPLOIT x86 NOOP [**] 220.197.192.39:25250 ->
MY.NET.190.93:135
02/20-03:18:05.425385  [**] EXPLOIT x86 NOOP [**] 67.38.244.224:2616 ->
MY.NET.190.97:135
02/20-07:15:46.403649  [**] EXPLOIT x86 NOOP [**] 203.59.206.76:1448 ->
MY.NET.190.102:135
```

Next, I analyzed all alerts that included these hosts, which identified 449 alert logs (this includes the 223 alert log entries for TCP port 135). The only records in which the university hosts were the source address were "SMB Name Wildcard" alerts (122 alerts). Excluding these alerts, 174 unique hosts generated between one and eleven alerts while communicating with these university hosts. Of these 327 alerts, 227 were "EXPLOIT x86 NOOP" alerts, 95 were "SMB C access" alerts, four were "External RPC call" alerts, and one alert was an "EXPLOIT x86 setuid 0" alert. In conclusion, it may be worth investigating MY.NET.190.93, MY.NET.190.95, MY.NET.190.97, and MY.NET.190.102 to see if they have been compromised and if they were used to attack 64.136.21.233. Because "EXPLOIT x86 NOOP" alerts (the exploit code used by Blaster would trigger this alert because of the NOOPs used within the exploit code) were generated within five minutes of the DoS attack for MY.NET.190.93 and MY.NET.190.97, I recommend investigating those two hosts first.

3.2.7 Evidence of active targeting

The DoS attack against TCP port 80 on 64.136.21.233 was a very specific attack and, as a result, I conclude that the target was actively targeted.

I also decided to analyze the hosts on the university's network that were possibly comprised and used as attackers in the DoS attack. Because the original DoS attack was using spoofed source addresses, we can only assume that the MY.NET.190.X hosts mentioned in the previous section were the potential source of the DoS attack. Using this assumption, 327 alert logs were generated that included these hosts (the breakdown of these logs was mentioned in the previous section). A review of the OOS logs disclosed two log entries, neither of significance. Review of the scan logs identified 1,161 log entries. Of those 1,161 log entries, 166 logs were for TCP port 135 and 153 logs were for TCP port 4444. Blaster attacks TCP port 135 and, when successful, it creates a remote shell listening on TCP port 4444. Performing a quick comparison between the alert log entries near the time of the DoS attack and the scan log entries, we see the following logs:

```
Alert logs:   02/20-15:30:17.186987  [**] EXPLOIT x86 NOOP [**] 193.248.0.182:4716 -
> MY.NET.190.97:135

Scan logs:    Feb 20 15:30:18 193.248.0.182:4736 -> 130.85.190.97:4444 SYN
******S*
```

It's possible that the attacker (193.248.0.182; a network address belonging to a French telecom (according to ripe.net/whois)) used an automated tool to exploit MY.NET.190.97 and then immediately attempted to connect to the remote shell on TCP port 4444 (the SYN scan log entry could be the actual SYN used to start the connection to the remote shell). In conclusion, it appears that the host was actively targeted. However, there were a total of 13,166 scan log entries in which hosts on the university's network were scanned for TCP port 135. Of those log entries, 27 included the source address of the attacker. In each of these 27 scan logs, the destination address was unique. Also, of the 13,166 scan log entries, 13,157 log entries had a destination address in the MY.NET.190.X subnet. It appears that although the MY.NET.190.X subnet was actively targeted, the specific target was not.

3.2.8 Severity

For the DoS attack, I assume that a publicly facing Web server for NetZero would be a critical machine; hence, I gave this a criticality score of 5. For lethality, if the attack was successful, legitimate users would not have been able to use the Web site. This could lead to legal ramifications for the university, but would not compromise the university's network. Consequently, I gave a lethality score of 3. Since the target is on a remote network, I can only make guesses about the countermeasure scores. I tried using telnet to connect to port 80 on 64.136.21.233 to attempt to identify the Web server version; however, I only identified that Apache (the version was not included) was being used (the screenshot can be found in Appendix E). In addition, the host was not identified in a "Webserver search" on netcraft.com (64.136.21.230 and 64.136.29.230 were the only hosts for my.netzero.net). Because I could not determine what the system countermeasures were, I decided to give the target a system countermeasure score of 3. HTTP traffic must be allowed to the target, but we don't' know if they are blocking traffic from private addresses at the border routers, firewalls, etc. As a result, I gave them a network countermeasure score of 2. Here is the final score:

Severity = (5+3) – (3+2) = 3

I also decided to score the potentially comprised hosts on the university's network. The criticality of these machines is low inasmuch as they appear to be Windows workstations. Other than the normal Windows services (TCP port 135, 139, 445), no other services were identified in the logs for these hosts. In addition, it appears that at most four hosts on the university's network have been compromised (all workstations). I gave these machines a criticality score of 1. If the systems were compromised, the attacker would have complete control of the systems. As a result, I gave a lethality score of 5. If the systems were compromised, they are missing patches from 2003. Consequently, I gave a system countermeasure score of 1. Assuming these machines are compromised, the university's network has allowed NETBIOS traffic, amongst other traffic, into the network from hosts outside the network. Because NETBIOS traffic should not be allowed into the network and because the university is allowing traffic from private addresses to leave the university network, I gave a network countermeasure of 1. Here is the final score:

Severity = (1+5) – (1+1) = 4

3.3 Detect 3 – EXPLOIT x86 NOOP

3.3.1 Description of the Detect

During the five-day timeframe, 1,019 "EXPLOIT x86 NOOP" alerts were generated indicating that a possible buffer overflow attempt occurred. The Intel x86 NOOP character, 0x90, is commonly used in shellcode that is used in buffer overflow attempts. NOOPs, or "no operations", are typically used to improve the attacker's chances of successfully exploiting the vulnerability. Here are some good references that describe NOOPs and buffer overflows:

http://www.phrack.org/show.php?p=49&a=14
http://www.sans.org/resources/idfaq/polymorphic_shell.php

Although NOOPs are common in buffer overflow attempts, they can also be found in binary transmissions across the network. For example, when a user browses a Web page, the image files that are sent to the user's browser may contain NOOPs, possibly triggering an IDS alert that is a false positive.

Because I discussed the buffer overflow attempts on TCP port 135 in the previous detect, I removed those log entries for this analysis (223 alerts removed resulting in 796 alerts to analyze). After eliminating these alerts, I identified 57 unique source addresses and 90 unique destination address. All of the destination address were on the university's network. Here is a breakdown of the addresses generating a majority of the alerts:

Top destination addresses (46.98% of the alerts)

Destination Address	Total Instances	% of Total
MY.NET.84.235	180	22.61%
MY.NET.5.92	99	12.44%
MY.NET.24.8	55	6.91%
MY.NET.150.207	40	5.03%

Top source addresses (89.7% of the alerts)

SrcIP	Total Instances	% of Total
195.154.199.21 0	448	56.28%
212.87.86.80	175	21.98%
131.118.254.13 0	51	6.41%
80.144.50.223	40	5.03%

Additional analysis disclosed that 46 alerts had a source port of 80, meaning that a university machine most likely visited the source address's Web site and an image (or some other type of binary file) was sent to the university machine. Of the remaining 750 alerts, here's the breakdown of the destination ports:

DstPort	Total Instances	% of Total	Common Service
80	451	60.13%	HTTP
389	175	23.33%	LDAP
119	55	7.33%	NNTP
12353	40	5.33%	Unknown
6881	11	1.47%	Unknown
1214	6	0.80%	KAZAA
445	4	0.53%	Microsoft-DS
6129	3	0.40%	Unknown
6882	3	0.40%	Unknown
2034	1	0.13%	Scoremgr
1071	1	0.13%	BSQUARE-VOIP

The data in the "Common Service" column was created using IANA's port list (http://www.iana.org/assignments/port-numbers). A destination port of 80 causes concern because users typically only send text to a Web server. Of the 451 alerts that had a destination port of 80, 448 had a source address of 195.154.199.210 (address belonging to a French telecommunications company) and three had a source address of 166.82.147.114 (address belonging to an ISP, CTC Internet Services, Inc.). Here is the breakdown of the corresponding destination addresses:

Destination Address	Total Instances	DNS Name
MY.NET.84.235	180	Unknown
MY.NET.5.92	99	Unknown
MY.NET.80.232	18	Unknown
MY.NET.83.98	17	Unknown
MY.NET.83.70	12	Unknown
MY.NET.112.216	12	Unknown
MY.NET.15.50	11	Unknown
MY.NET.189.62	11	Unknown
MY.NET.29.19	10	lyekka.umbc.edu
MY.NET.150.101	10	scholarseek.lib.umbc.edu
MY.NET.66.24	9	Unknown
MY.NET.5.95	8	Unknown
MY.NET.5.44	8	ndms.umbc.edu

MY.NET.112.226	8	Unknown
MY.NET.29.8	8	cms.umbc.edu
MY.NET.150.44	7	illiad.lib.umbc.edu
MY.NET.5.67	7	ccrf.umbc.edu
MY.NET.111.72	5	cuereims.umbc.edu
MY.NET.5.46	4	pp1.umbc.edu
MY.NET.5.45	3	cyclone.umbc.edu
MY.NET.5.20	2	centrelearn.umbc.edu
MY.NET.5.25	2	ehs.umbc.edu

166.82.147.114 only targeted one host, MY.NET.29.8 (cms.umbc.edu):

```
02/20-01:12:27.521077  [**] EXPLOIT x86 NOOP [**] 166.82.147.114:3066 ->
MY.NET.29.8:80
02/20-01:12:27.804511  [**] EXPLOIT x86 NOOP [**] 166.82.147.114:3066 ->
MY.NET.29.8:80
02/20-01:12:27.820896  [**] EXPLOIT x86 NOOP [**] 166.82.147.114:3066 ->
MY.NET.29.8:80
```

195.154.199.210 attacked from 16:13:03 to 18:46:23 (2 hours and 33 minutes) on February 19, averaging close to 3 alerts per minute. Here are some example logs:

```
02/19-16:35:39.266075  [**] EXPLOIT x86 NOOP [**] 195.154.199.210:3518 ->
MY.NET.112.226:80
02/19-16:36:00.929232  [**] EXPLOIT x86 NOOP [**] 195.154.199.210:1654 ->
MY.NET.66.24:80
```

The attacker used source ports ranging from 1062 to 4862. The attacks occurred in bursts; hence, it appears that the attacker was not using an automated attack tool, unless the tool had some type of "attack delay" functionality. The first two sets of attacks occurred 22 minutes apart, followed by another attack 53 minutes later. Here is the breakdown of the times of the attacks (several alerts were generated during each attack; these times are the starting times of the groups of alerts):

Time	**Difference**
16:13:03	
16:35:39	0:22:36
17:28:38	0:52:59
17:35:05	0:06:27
17:41:01	0:05:56
17:44:14	0:03:13

18:22:39	0:38:25
18:24:42	0:02:03
18:26:42	0:02:00
18:28:41	0:01:59
18:30:41	0:02:00
18:32:40	0:01:59
18:34:33	0:01:53
18:36:23	0:01:50
18:38:19	0:01:56
18:40:11	0:01:52
18:41:59	0:01:48
18:43:56	0:01:57
18:46:06	0:02:10

Starting at 18:22, the attacks started occurring on a more regular basis (every 2 minutes). The destination hosts in the alerts appeared in groups (e.g., MY.NET.83.98 was the destination in nine consecutive log entries), but not in any particular order. Furthermore, the attacker was targeting the same host at several different intervals (e.g., MY.NET.112.226 was the target at 16:13 and 16:36).

3.3.2 Reason the attack was selected

The attack was selected because of both the total number of alerts (it had the eighth highest alert total) and the potential damage that a successful buffer overflow attempt could cause. In addition, buffer overflow attacks have become extremely popular in recent years and the end result is typically a total system compromise.

3.3.3 Detect was generated by

The detect was generated by a Snort rule that generates an alert for traffic from remote networks destined for hosts on the university's network that has a payload that includes NOOPs. The rule would look similar to this:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"EXPLOIT x86 NOOP";
content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90|";)
```

This rule is similar to the "SHELLCODE x86 NOOP" rule on snort.org (http://www.snort.org/snort-db/sid.html?sid=648).

3.3.4 Probability the source address was spoofed

It is not likely that the source addresses were spoofed, due to the fact that the goal of many buffer overflow exploits is to obtain a remote shell on the compromised machine. If the address is spoofed, the attacker will not be able to obtain a remote shell. In addition, the attack is focused on port 80. Because HTTP typically runs over TCP port 80, the attack is most likely occurring over TCP. This further concludes that it is unlikely the source addresses were spoofed.

3.3.5 Attack mechanism

For remote buffer overflow exploits, an attacker typically targets a specific service and attempts to get the service to execute the code of the attacker's choice. A buffer overflow occurs when a process attempts to store more data in a buffer than it was intended to hold. In a successful buffer overflow, the attacker overflows the buffer with data that includes the attacker's shellcode, and overwrites the return address pointer to point to the attacker's shellcode on the stack. The NOOPs are included so that the attacker does not need to know the exact memory location of the shellcode. As a result, the chances of successfully executing the shellcode improve. Here is a reference for a buffer overflow tutorial:

http://mixter.void.ru/exploit.html

For known buffer overflow vulnerabilities, exploit code is often publicly available on the Internet.

3.3.6 Correlations

As mentioned previously, there are several good references that discuss NOOP exploits. My analysis focused on buffer overflow exploits against TCP port 80. There have been many buffer overflow exploits for Web servers as indicated in the SANS Top 20 Vulnerabilities list (http://www.sans.org/top20/). The number one Windows vulnerability references the buffer overflow utilized by the Code Red worm to attack IIS Web servers. In addition, the number one UNIX vulnerability references Web servers running on UNIX platforms. Apache, a popular Web server for UNIX platforms, has been vulnerable to buffer overflow exploits in the past, including the "Apache Web Server Chunk Handling Vulnerability".

I searched the remaining alert log entries, in addition to the scan and OOS log entries, for additional logs containing either 195.154.199.210 or 166.82.147.114, but no other log entries were identified. A review of the targets disclosed some additional details. In the OOS logs, MY.NET.84.235 was the destination address in 446 of the 449 logs. In those 446 logs, the destination port was TCP 4662, a port commonly associated with the P2P file sharing program, eDonkey2000 (http://www.edonkey2000.com/). In the remaining three logs, three different university hosts were the destination addresses and all traffic was destined for TCP port 80. It should be noted that only 80 of the log entries occurred during the time frame of the security audit, all of which occurred after the attack that was analyzed. Because the dates in the log entries did not correspond to the file names, the remaining OOS log entries occurred after the time period evaluated.

Analysis of the remaining alert logs that included the targets in this attack disclosed an additional 224 alert logs. In 136 alerts, MY.NET.150.44 (illiad.lib.umbc.edu) was the source address and the destination port was always TCP 137 (NETBIOS Name Service). The alert generated was called "SMB Name Wildcard", which is triggered by the Windows command "nbtstat –A IP_Address" (http://www.sans.org/resources/idfaq/port_137.php). MY.NET.150.44 appeared to be looking for Windows hosts (the alerts were generated over each of the five days). In addition, only five different source ports were used:

SrcPort	Total Instances
1065	34
1052	30
1064	25
137	24
1061	23

In the scan logs, MY.NET.84.235 was the most active host as both the source and destination of scans. MY.NET.84.235 was the source address 5,355 times and was the only target found in the scan logs as a source address. This host was looking for TCP port 4662 (eDonkey2000) 2,455 times, UDP port 4672 (eMule) 612 times, and TCP port 6346 (Gnutella) 634 times. As the destination address, MY.NET.84.235 was identified in 843 logs (out 1,817 logs). Of those logs, TCP port 4662 (eDonkey2000) was the destination port 780 times.

TCP port 80 was the destination port in 89 scan logs (out of 1,817 can logs) amongst 22 of the targets identified in the attacks. MY.NET.5.20 (centrelearn.umbc.edu) was the most popular destination (18 logs) for these scans. The scans occurred on each of the five days.

3.3.7 Evidence of active targeting
It appears that the hosts were actively targeted. Of the "EXPLOIT x86 NOOP" alerts, TCP port 80 was the most popular target port. In addition, the exploit was only attempted on 22 different hosts, many of which were on different subnets. In addition, the source addresses of the attackers (166.82.147.114 and 195.154.199.210) were not found in any other log entries (includes all log types).

3.3.8 Severity
Of the 22 targets in the attack, 11 did not have DNS names. In addition, www.umbc.edu was not included as a target. However, eight of the 11 remaining hosts appeared to be valid university Web servers. As a result, I gave a criticality score of 4. For lethality, if the attack was successful, it could result in total system compromise. Consequently, I gave a lethality score of 5. For system countermeasures, patches are available for known buffer overflow exploits. However,

unless failover Web servers are available, patching may not occur as often as needed because the Web server typically has to be taken offline to apply the patch. As a result, I gave a system countermeasure score of 3. Finally, HTTP traffic must be allowed to the eight legitimate Web servers but could be blocked at the firewall for the remaining 14 targets. Because the legitimate Web servers are more important, I gave a network countermeasure score of 2. Here is the final score:

Severity = (5+5) – (3+2) = 5

4 Network statistics

4.1 Top Talkers

For my analysis on the "Top Talkers", I focused first on the alert logs because, in my opinion, the alerts are the most critical of the different types of logs. Before analyzing the alert logs, I excluded all of the “portscan” alerts inasmuch as the details of these alerts were captured in the scan logs. Here are the top five talkers from the alert logs and the number of instances they were identified in the alert logs:

Alert Logs

Host	Total Instances
64.136.21.233	19292
MY.NET.30.4	11114
MY.NET.29.3	7162
169.200.215.36	7141
MY.NET.30.3	5621

I then broke this down into the university’s top five talkers in the alert logs versus the top five external hosts in the alert logs:

Alert Logs

University Hosts	Total Instances	DNS Name	External Hosts	Total Instances	DNS Name
MY.NET.30.4	11114	lan2.umbc.edu	64.136.21.233	19292	
MY.NET.29.3	7162	bb-app4.umbc.edu	169.200.215.36	7141	
MY.NET.30.3	5621	lan1.umbc.edu	12.21.173.176	2706	
MY.NET.153.37	1988	refweb08.libpub.umbc.edu	68.6.96.171	1979	ip68-6-96-171.sb.sd.cox.net
MY.NET.11.7	1040	dc2.ad.UMBC.EDU	134.192.40.28	1884	

Two of the university’s top talkers were potentially honeypots (MY.NET.30.3 and MY.NET.30.4), while the top “talking” external host, 64.136.21.233, belonged to Juno Online Services, Inc. and was discussed in the second detect above.

I then performed the same analysis on the OOS and scan logs. Here are the top talkers from the OOS logs:

OOS Logs

Host	Total Instances
MY.NET.6.7	1222
68.54.84.49	1184
MY.NET.12.6	671
MY.NET.84.235	446
MY.NET.24.44	288

Here is the breakdown between the university's top five talkers in the OOS logs versus the top five external hosts in the OOS logs:

OOS Logs

University Hosts	Total Instances	DNS Name	External Hosts	Total Instances	DNS Name
MY.NET.6.7	1222	umbc7.umbc.edu	68.54.84.49	1184	
MY.NET.12.6	671	mxin.umbc.edu	69.10.138.119	237	
MY.NET.84.235	446		82.161.49.116	227	
MY.NET.24.44	288		80.54.216.115	195	
MY.NET.42.1	238		81.9.192.27	139	

Here are the top talkers from the scan logs:

Scan Logs

Host	Total Instances
MY.NET.1.3	2490625
MY.NET.81.39	859105
MY.NET.1.4	589086
MY.NET.111.197	191580
MY.NET.34.14	135436

Here is the breakdown between the university's top five talkers in the scan logs versus the top five external hosts in the scan logs:

Scan Logs

University Hosts	Total Instances	DNS Name	External Hosts	Total Instances	DNS Name
MY.NET.1.3	2490625	UMBC3.UMBC.EDU	192.26.92.30	75508	c.gtld-servers.net
MY.NET.81.39	859105		192.5.6.30	48083	a.gtld-servers.net
MY.NET.1.4	589086	UMBC4.UMBC.EDU	192.48.79.30	46277	j.gtld-servers.net
MY.NET.111.197	191580	trc157pc-03.engr.umbc.edu	69.6.68.10	41343	noname.wholesalebandwidth.com
MY.NET.34.14	135436	imap.cs.UMBC.EDU	203.20.52.5	37656	

Next, I identified the top talkers by reviewing all of the logs together:

All Logs

IP	Total Instances
MY.NET.1.3	2490984
MY.NET.81.39	859105
MY.NET.1.4	589162
MY.NET.111.197	191647
MY.NET.34.14	135510

Here is the breakdown between the university's top five talkers overall versus the top five external hosts overall:

All Logs

University Hosts	Total Instances	DNS Name	External Hosts	Total Instances	DNS Name
MY.NET.1.3	2490984	UMBC3.UMBC.EDU	192.26.92.30	75508	c.gtld-servers.net
MY.NET.81.39	859105		192.5.6.30	48083	a.gtld-servers.net
MY.NET.1.4	589162	UMBC4.UMBC.EDU	192.48.79.30	46277	j.gtld-servers.net
MY.NET.111.197	191647	trc157pc-03.engr.umbc.edu	69.6.68.10	41345	noname.wholesalebandwidth.com
MY.NET.34.14	135510	imap.cs.UMBC.EDU	203.20.52.5	37656	

It should be noted that the top talkers overall were the same as the top talkers in the scan logs. By comparing the overall top talkers versus the top talkers in the scan logs, it appears that over 99% of the logs were scan logs. The following chart shows hosts, the number of instances that each host was identified across all logs, and the percentage of those logs that came from the scan logs.

All Logs vs Scan Logs

University Hosts	Total Instances	% Scans	External Hosts	Total Instances	% Scans
MY.NET.1.3	2490984	99.99%	192.26.92.30	75508	100.00%
MY.NET.81.39	859105	100.00%	192.5.6.30	48083	100.00%
MY.NET.1.4	589162	99.99%	192.48.79.30	46277	100.00%
MY.NET.111.197	191647	99.97%	69.6.68.10	41345	100.00%
MY.NET.34.14	135510	99.95%	203.20.52.5	37656	100.00%

4.2 Top Targeted Services or Ports

I first identified the top ports used across all logs and the service commonly running on those ports using IANA's port list (http://www.iana.org/assignments/port-numbers):

Port	Total Instances	Service
53	3067913	DNS
32783	2481037	unassigned
135	1084698	DCE endpoint resolution
32788	586195	unassigned
25	211335	SMTP

Ports 32783 and 32788 were unassigned ports according to IANA. I also used the port report on dshield.org (e.g., http://www.dshield.org//port_report.php?port=32783), but this did not disclose what services commonly run on these ports. A quick Google search disclosed that these ports are typically used for RPC services (http://www.seifried.org/security/ports/32000/32783.html; http://www.seifried.org/security/ports/32000/32788.html). It should also be noted that these ports can also be used as ephemeral ports. Ports 135, 53, and 25 were the second, eighth, and tenth most targeted ports on the Internet respectively according to dshield.org (http://www.dshield.org/topports.php)

Next, I identified the top destination ports across all logs to help identify the targeted services:

Port	Total Instances	Service
53	3067718	DNS
135	1084698	DCE endpoint resolution
25	210996	SMTP
6129	123072	unassigned
20168	103416	unassigned

According to the port report on dshield.org, TCP port 6129 is commonly associated with Dameware Development's Remote Admin tool (http://www.dameware.com/products/dntu/), a tool used to remotely administer Windows machines. A port report on dshield.org for port 20168 did not disclose any additional details about the service running on that port. A Google search disclosed that TCP port 20168 is associated with some of the Lovgate worm variants. The Lovgate worm opens up a remote shell backdoor on TCP port 20168 on infected machines (from Symantec's site – http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.lovgate.d@mm.html).

I also wanted to see if these were the same top targeted services on university hosts as well. Here are the top targeted ports from the logs in which the destination address was a university address:

Port	Total Instances	Service
6129	123062	Unassigned
2016 8	103416	Unassigned
80	67245	HTTP
4000	61075	ICQ
81	23661	HOSTS2 Name Server

Although the top ports differed in this list, the only unexpected port in the top five was port 81. The HOSTS2 Name Server service typically runs over port 81 according to IANA. However, TCP port 81 is also associated with the RemoConChubo trojan (according to dshield.org) and variants of the Beagle worm according to symantec.com (http://securityresponse.symantec.com/avcenter/venc/data/w32.beagle.av@mm.html). Variants of the Beagle worm open a backdoor on TCP port 81 and UDP port 81, which are then used to relay e-mail.

4.3 Three suspicious external addresses

While searching for suspicious external addresses, I focused on addresses that created a lot of logs, but also had very focused attacks.

4.3.1 212.87.86.80

This IP address belongs to Novatech Directed Limited, a computer supply company based in the UK. The results of the whois query from ripe.net can be found in Appendix F.

On February 20 at 12:47:00 PM, this host SYN scanned the university's network for hosts listening on TCP port 389 (LDAP). The scan lasted nine minutes and 23 seconds and covered 5,721 university network addresses. Several addresses were scanned more than once and, as a result, 6,548 scan logs were generated for this scan. At 12:55:32 (before the scanning was complete), 212.87.86.80 generated 175 "EXPLOIT x86 NOOP" alerts in which TCP port 389 on 24 university hosts was targeted. Based on some of the DNS names of the targets, it appears the host was targeting some Windows servers (although the LDAP service can run on other platforms), including domain controllers (e.g., dc1.ad.umbc.edu (MY.NET.11.6) was a target).

4.3.2 213.85.29.136

This IP address belongs to RialCom, a Russian organization. The results of the whois query from ripe.net can be found in Appendix F.

On February 18 at 8:17:40 AM, 213.85.29.136 started SYN scanning 239 university hosts, all on the MY.NET.190.X subnet, for TCP port 111 (SUNRPC). The scan lasted only seven seconds but generated 382 scan logs (some hosts were scanned more than once). During the same time frame, 525 "External RPC call" alert logs were generated with 213.85.29.136 as the source address, port 111 as the destination port, and hosts on the MY.NET.90.X subnet as the destination address. Because we have more alert logs than scan logs, some of the university hosts may have responded to the SYN scan. Earlier in the day, the attacker generated the same types of alerts while communicating with TCP port 111 on five university hosts, none of which were in the MY.NET.190.X subnet. Of those five university hosts, one was a Web server and three appeared to be some type of VPN device based on their DNS names (e.g., umbcvpn-delta.umbc.edu).

4.3.3 204.152.186.189

This IP address belongs to Internet Systems Consortium, Inc., the non-profit corporation that develops and supports BIND (among other applications). BIND is a popular software application used to implement DNS protocols. The results of the whois query from networksolutions.com can be found in Appendix F.

204.152.186.189 was the "noisiest" of the port scanners that also created alert logs. Of the 30,501 scan logs that included 204.152.186.189, 6,931 scan logs existed in which the university's DNS servers, MY.NET.1.3 and MY.NET.1.4, communicated with UDP port 53 (DNS) on 204.152.186.189. Because this external address belongs to ISC, this may be valid DNS traffic. Of the remaining 23,570 scan logs, all of the logs were the result of a SYN scan against MY.NET.25.73 (mx8in.umbc.edu), a university mail server. The scans used source ports ranging from 35924 to 35933, destination ports ranging from 167 to 65391, and lasted 32 minutes and 12 seconds, starting at 10:32:15 on February 19. During the scans, two TFTP alerts were generated between the hosts in which the attacker was communicating with TCP port 69 on the mail server. Five minutes after the scans, the attacker tried communicating with TCP port 32771 on the mail server, generating nine alerts.

5 Correlations

I have mentioned several practicals and Web sites throughout the paper. These are a list of GCIA practicals that influenced my analysis: Dave McFarland, Doug Kite, Eric Evans, Gregory Lalla, Johnny Calhoun, Jorge Perez, and Travis Bow. Other practicals used were Jeffrey Tomaszewski (GCUX), Mark Donaldson (GSEC), and William Townsend (GSEC). All practicals and sites are included in the References section below.

6 Malicious activity

Malicious activity was identified in each of the three detects, as well as some possible malicious activity from the three suspicious external hosts discussed above. A breakdown of the university hosts that may have been compromised can be found in Appendix B.

7 Defensive Recommendations

From a policy and procedure standpoint, I recommend that the university review any existing policies to ensure that they are up to date. In addition, policies should be modified to explicitly define acceptable use of the university's network. For example, the policy should state that the use of P2P programs on the university's network is prohibited. The consequences of using P2P programs should also be clearly outlined. A policy on acceptable use was found (http://www.umbc.edu/oit/sans/security/policy/2-UMBC/IT-01-final.html) on the university's Web site that was dated 1996. The university should consider reviewing this policy and making revisions to ensure it is current.

In addition to the policies, some type of security awareness training should be conducted, perhaps during student orientation, through articles in the school newspaper, or by creating mandatory training classes for students. This will help educate the users of the university's network about the risks involved in using poor information security practices.

The university should also create an incident response team if one does not already exist, and ensure that the information security team and incident response members are appropriately trained to protect the university's network and handle incidents.

From a technical standpoint, the university should either implement a firewall, establishing a perimeter defense around the university's network, or re-configure their existing firewall. Written policies should be enforced in the firewall by blocking traffic into and out of the university's network. In addition, the firewall should be configured to deny any traffic that is not explicitly allowed. Furthermore, the firewall should be configured to only allow legitimate traffic from the Internet to specific machines (e.g., university Web servers). For example, users from the Internet should not be able to communicate with Web servers running on student laptops unless it is authorized by the university.

The university should also define their idea of a trusted machine versus an untrusted machine. If students are allowed to connect their personal computers to the university's network and the university has no control over those computers, those machines should be considered untrusted and handled differently than university-owned computers. For example, student-owned computers could be quarantined to a specific subnet in which different firewall rules or router filters apply.

Anti-virus software should be distributed to all university workstations and should be updated regularly. If the funding is available, all students should be given anti-virus software to install on their computers that will be connecting to the university's network. Ideally, there would be technical people available to help students install the anti-virus software, install patches, install personal firewalls, and answer security-related questions.

The university should also develop a patch management strategy to ensure trusted devices are updated regularly. It would also be a good idea for the university to invest in an endpoint security solution if funding is available. An endpoint security solution can enforce the university's patch management strategy, enforce anti-virus updates, and protect machines when connecting to untrusted networks (e.g., university-owned laptops connecting to hotel networks during business travel).

Finally, the university should spend time tuning their IDS sensors, re-evaluating the placement of their IDS sensors, and developing a process for testing and implementing new IDS rules. For example, the rule that triggers the "Possible trojan server activity" alerts can be modified to reduce the number of false positives. To facilitate the tuning of the IDS sensors, creating a process to test and implement new IDS rules as they are released by snort.org (or other organizations) will also help reduce false positives. In addition, the university may want to consider placing IDS sensors inside of and outside of the firewall to see not only what traffic is coming into the network, but also leaving the network.

Part III - Analysis Process

I used a few different machines and tools to perform my analysis. I first downloaded all of the logs for the time period selected from the SANS Web site onto a Windows XP SP2 machine using an Intel Pentium 4 3.4 GHz processor and 1 GB RAM.

Because of the size of the scan logs, I decided to first merge the log files into one file for each log type using the 'cat' command in cygwin (http://www.cygwin.com/) (e.g., cat scans.040220 >> allscans). I then created a Perl script to parse each log into semicolon delimited files. The Perl script can be found in Appendix A. Next, I manually removed log entries that were either incomplete or merged with other logs entries. The log entries that were removed were saved in separate files.

To ensure the number of records in the original logs was the same as the number of entries in the parsed logs, I used the 'grep', 'cut', and 'wc –l' commands in cygwin to compare the numbers. For example, I used the "grep '02/' oos_report_all | cut -d\ -f1 | wc –l" command to determine the number of log entries in the original OOS logs from 2/20/04. I then used the "grep '02/' oos_report_all-scrubbed.txt | cut -d\; -f1 | wc –l" command to verify that the same number of log entries existed in the semicolon delimited file. These same commands (substituting the files names) were used on all original and parsed files. The number of records in the original files equaled the number of records in the parsed files plus the removed logs in all instances.

The files were then imported into Microsoft Access 2003. Because Access queries were utilizing all of my computing resources (due to the number of records in the database), I used the DTS Import/Export Wizard provided with Microsoft SQL Server 2000 to copy the data from Access to a server running SQL Server 2000. The server running the SQL Server was a Windows Server 2003 Standard Edition server running on an Intel Pentium 4 1.6 GHz processor with 256 MB RAM.

After running some SQL queries using Query Analyzer, I realized that "MY.NET." was not found in the scan logs. After cross-referencing the entries in the scan logs with the "portscan" entries in the alert logs, I determined that "MY.NET." was the equivalent of "130.85.". I then wrote a simple Perl script to replace "130.85." with "MY.NET." in the scan logs and re-imported the scan logs into SQL Server. The Perl script can also be found in Appendix A (called my.net.pl).

I then started analyzing the data. Because the alert logs were the most critical in my opinion, I focused on analyzing the alerts first. Because port scans were identified in both the alert logs and the scan logs, my original queries against the alert logs excluded any ”portscan” events. The rest of the steps of my analysis are included throughout the paper.

I’ve also included some of the SQL queries I used to analyze the data in Appendix D. Additional tools used were Microsoft Word 2003 for writing the paper and Microsoft Excel 97 and 2003 for performing further analysis on the SQL query results.

References

- Network Solutions – Whois queries
http://www.networksolutions.com/en_US/whois/index.jhtml

- CVE-1999-0059 (fam service)
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0059

- SecurityFocus HOME Vulns Info: IRIX fam service Vulnerability
http://www.securityfocus.com/bid/353

- RIPE Network Coordination Centre – Whois queries
http://www.ripe.net/whois

- SANS Intrusion Detection FAQ – SubSeven Trojan v 1.1
http://www.sans.org/resources/idfaq/subseven.php

- eMule-Project.net – common eMule ports
http://www.emule-project.net/home/perl/help.cgi?l=1&topic_id=122&rm=show_topic

- Aaron Greenlee GSEC practical
http://www.sans.org/rr/whitepapers/malicious/958.php

- CERT® Advisory CA-2003-20 W32/Blaster worm
http://www.cert.org/advisories/CA-2003-20.html

- CVE – CAN-2003-0352 (Blaster)
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352

- Microsoft Security Bulletin MS03-026 (Blaster)
http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx

- Last Stage of the Delirium Research Group (Blaster)
http://lsd-pl.net/

- John Van Hoogstraten GCIH practical
http://www.giac.org/certified_professionals/practicals/gcih/0489.php

- Netcraft What's That Site Running Results
http://uptime.netcraft.com/up/graph/?host=my.netzero.net

- Phrack.org – Phrack 49 (buffer overflows)
http://www.phrack.org/show.php?p=49&a=14

- SANS Intrusion Detection FAQ – What is polymorphic shell code and what can it do? (buffer overflows)
http://www.sans.org/resources/idfaq/polymorphic_shell.php

- IANA – Port Numbers
http://www.iana.org/assignments/port-numbers

- Snort.org - "SHELLCODE x86 NOOP" rule
http://www.snort.org/snort-db/sid.html?sid=648

- Writing buffer overflow exploits – a tutorial for beginners
http://mixter.void.ru/exploit.html

- SANS Top 20 Vulnerabilities – The Experts Consensus
http://www.sans.org/top20/

- eDonkey2000 – Overnet
http://www.edonkey2000.com/

- SANS Intrusion Detection FAQ – Port 137 Scan
http://www.sans.org/resources/idfaq/port_137.php

- Dshield.org – Port Report (example port report for port 32783 below)
http://www.dshield.org//port_report.php?port=32783

- Kurt Seifried – Information Security
http://www.seifried.org/security/ports/32000/32783.html
http://www.seifried.org/security/ports/32000/32788.html

- Dshield.org – Top 10 Target Ports
http://www.dshield.org/topports.php

- DameWare – NT Utilities (Remote Admin)
http://www.dameware.com/products/dntu/

- Symantec Security Response – W32.HLLW.Lovgate.D@mm
http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.lovgate.d@mm.ht

ml

- Symantec Security Response – W32.Beagle.AV@mm
http://securityresponse.symantec.com/avcenter/venc/data/w32.beagle.av@mm.html

- Dave McFarland GCIA practical
http://www.giac.org/certified_professionals/practicals/gcia/0698.php

- Doug Kite GCIA practical
http://www.giac.org/certified_professionals/practicals/gcia/0609.php

- Eric Evans GCIA practical
http://www.giac.org/certified_professionals/practicals/gcia/0684.php

- Gregory Lalla GCIA practical
http://www.giac.org/certified_professionals/practicals/gcia/0700.php

- Johnny Calhoun GCIA practical
http://www.giac.org/certified_professionals/practicals/gcia/0600.php

- Jorge Perez GCIA practical
http://www.giac.org/certified_professionals/practicals/gcia/0742.php

- Travis Bow GCIA practical
http://www.giac.org/certified_professionals/practicals/gcia/0699.php

- Jeffrey Tomaszewski GCUX practical
http://www.giac.org/certified_professionals/practicals/gcux/0221.php

- Mark Donaldson GSEC practical
http://www.giac.org/certified_professionals/practicals/gsec/1814.php

- William Townsend GSEC practical
http://www.giac.org/certified_professionals/practicals/gsec/2383.php

- Cygwin
http://www.cygwin.com/

Appendix A

The following script was used to parse the original logs into semicolon delimited files. A log file is also created for lines that don't match any patterns.

```
##########################################################################
#Script to parse GCIA logs (oos, scan, and alert logs)
#1/19/05
```

```
$input = $ARGV[0];
$output = $ARGV[1];
$log = $ARGV[2];
$debug = $ARGV[3];
$ctr = 0;
$lineno = 0;

open(INFILE,$input) or die "Could not open $input : $!\n";
open(OUTFILE,">$output") or die "Could not open $output for output\n";
open(LOG,">$log") or die "Cound not open log file $log : $!\n";

while(<INFILE>)
{
#for alert logs
 if($_ =~ /^(.*)\[\*\*\](.*)\[\*\*\] (\S*)\:(\S*) \-\> (\S*)\:(\S*)/)
 {
   $tframe = $1;
   $desc = $2;
   $srcip = $3;
   $srcport = $4;
   $dstip = $5;
   $dstport = $6;
   $tframe =~ s/^\s+//;
   $tframe =~ s/\s+$//;
   $desc =~ s/^\s+//;
   $desc =~ s/\s+$//;
   if($debug == 1) {print "$tframe;$desc;$srcip;$srcport;$dstip;$dstport;alertlogs\n"}
   else {print OUTFILE "$tframe;$desc;$srcip;$srcport;$dstip;$dstport;\n"};
   $lineno+=1;
 }
#for alert logs
 elsif($_ =~ /^(.*)\[\*\*\](.*)\[\*\*\]/)
 {
   $tframe = $1;
   $desc = $2;
   $srcip = "";
   $srcport = "";
   $dstip = "";
   $dstport = "";
   $tframe =~ s/^\s+//;
   $tframe =~ s/\s+$//;
   $desc =~ s/^\s+//;
   $desc =~ s/\s+$//;
   if($debug == 1) {print "$tframe;$desc;$srcip;$srcport;$dstip;$dstport;alertlogs\n"}
   else {print OUTFILE "$tframe;$desc;$srcip;$srcport;$dstip;$dstport;\n"};
   $lineno+=1;
 }
#for scan logs
 elsif($_ =~ /(\S*) (\S*) (\S*) (\S*):(\S*) \-\> (\S*):(\S*) (\S*) (\S*)/)
 {
   $mon = $1;
   $day = $2;
   $time = $3;
   $srcip = $4;
   $srcport = $5;
   $dstip = $6;
   $dstport = $7;
   $ptype = $8;
   $tcpflags = $9;
   if($debug == 1) {print
"$mon;$day;$time;$srcip;$srcport;$dstip;$dstport;$ptype;$tcpflags;scanlogs\n"}
   else {print OUTFILE
"$mon;$day;$time;$srcip;$srcport;$dstip;$dstport;$ptype;$tcpflags;\n"};
   $lineno+=1;
 }
#for scan logs
 elsif($_ =~ /(\S*) (\S*) (\S*) (\S*):(\S*) \-\> (\S*):(\S*) (\S*)/)
 {
   $mon = $1;
   $day = $2;
   $time = $3;
   $srcip = $4;
   $srcport = $5;
   $dstip = $6;
   $dstport = $7;
   $ptype = $8;
   $tcpflags = "";
   if($debug == 1) {print
"$mon;$day;$time;$srcip;$srcport;$dstip;$dstport;$ptype;$tcpflags;scanlogs\n"}
   else {print OUTFILE
"$mon;$day;$time;$srcip;$srcport;$dstip;$dstport;$ptype;$tcpflags;\n"};
   $lineno+=1;
 }
 #for oos logs
 #for time and ip line
 #e.g., 02/20-00:05:10.349493 68.54.84.49:60002 -> MY.NET.6.7:110
 elsif($_ =~ /(\S*) (\S*)\:(\S*) \-\> (\S*)\:(\S*)/)
 {
   $tframe = $1;
   $srcip = $2;
   $srcport = $3;
   $dstip = $4;
```

```
  $dstport = $5;
  if($debug) {print "\n$tframe;$srcip;$srcport;$dstip;$dstport;"}
  else {print OUTFILE "\n$tframe;$srcip;$srcport;$dstip;$dstport;"};
  $lineno+=1;
}
#skip oos record delimiter and blank lines
#e.g., =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
elsif(($_ =~ /^\=\+.*\=\+$/) || ($_ =~ /^$/))
{
}
#Iplen line
#e.g, TCP TTL:51 TOS:0x0 ID:64221 IpLen:20 DgmLen:60 DF
elsif($_ =~ /(\S*) TTL\:(\S*) TOS\:(\S*) ID\:(\S*) IpLen\:(\S*) DgmLen\:(.*)/)
{
  $proto = $1;
  $ttl = $2;
  $tos = $3;
  $id = $4;
  $iplen = $5;
  $dgmlen = $6;
  chomp($dgmlen); #in case we grab the newline
  if($debug) {print "$proto;$ttl;$tos;$id;$iplen;$dgmlen;"}
  else {print OUTFILE "$proto;$ttl;$tos;$id;$iplen;$dgmlen;"};
  $lineno+=1;
}
#win size line with urgptr
#e.g., 12UAP*S* Seq: 0xD17DF816  Ack: 0x16E1347C  Win: 0x8556  TcpLen: 36
UrgPtr: 0x6798
elsif($_ =~ /^(\S*) Seq\: (\S*)  Ack\: (\S*)  Win\: (\S*)  TcpLen\: (\S*)  UrgPtr\: (\S*)/)
{
  $tcpflags = $1;
  $seq = $2;
  $ack = $3;
  $win = $4;
  $tcplen = $5;
  $urgptr = $6;
  chomp($urgptr);
  if($debug) {print "$tcpflags;$seq;$ack;$win;$tcplen;$urgptr;"}
  else {print OUTFILE "$tcpflags;$seq;$ack;$win;$tcplen;$urgptr;"};
  $lineno+=1;
}
#win size line, no urgptr
#e.g., 12****S* Seq: 0x5CE1AE3A  Ack: 0x0  Win: 0x16D0  TcpLen: 40
elsif($_ =~ /^(\S*) Seq\: (\S*)  Ack\: (\S*)  Win\: (\S*)  TcpLen\: (.*)/)
{
  $tcpflags = $1;
  $seq = $2;
  $ack = $3;
  $win = $4;
  $tcplen = $5;
  $urgptr = '';
  chomp($tcplen);
  if($debug) {print "$tcpflags;$seq;$ack;$win;$tcplen;$urgptr;"}
  else {print OUTFILE "$tcpflags;$seq;$ack;$win;$tcplen;$urgptr;"};
  $lineno+=1;
 }
 #for TCP options line
 #e.g, TCP Options (5) => MSS: 1460 SackOK TS: 465851623 0 NOP WS: 0
 elsif($_ =~ /^TCP Options/)
 {
  $tcpopt = $_;
  chomp($tcpopt);
  if($debug) {print "$tcpopt;ooslogs"}
  else {print OUTFILE "$tcpopt;"};
  $lineno+=1;
 }
 #in case we don't match any patterns..
 else {
  $line = $_;
  chomp($line);
  $ctr+=1;
  $lineno+=1;
  if($debug) {print "Line number = $lineno\t$line\n"}
  else {print LOG "Line number = $lineno\t$line\n"};
 }
}

#print count of bad lines
if($debug) {print "\nCtr\=$ctr\n"}
else {print LOG "\nCtr\=$ctr\n"};

print "\nDone\n";

close INFILE;
close OUTFILE;
close LOG;

#the end
####################################################################
```

This Perl script was used to replace “130.85.” with “MY.NET.” in the scan logs:

```
####################################################################
```

```
#my.net.pl

$input = $ARGV[0];
$output = $ARGV[1];

open(INFILE,$input);
open(OUTFILE,">$output");

while(<INFILE>)
{
 $str = $_;
 $str =~ s/\;130\.85\./\;MY\.NET\./g;
 print $str;
}

close INFILE;
close OUTFILE;
#####################################################################
```

Appendix B

Here is a breakdown of possibly compromised hosts on the university's network. The table also includes which section of the paper the hosts were identified in:

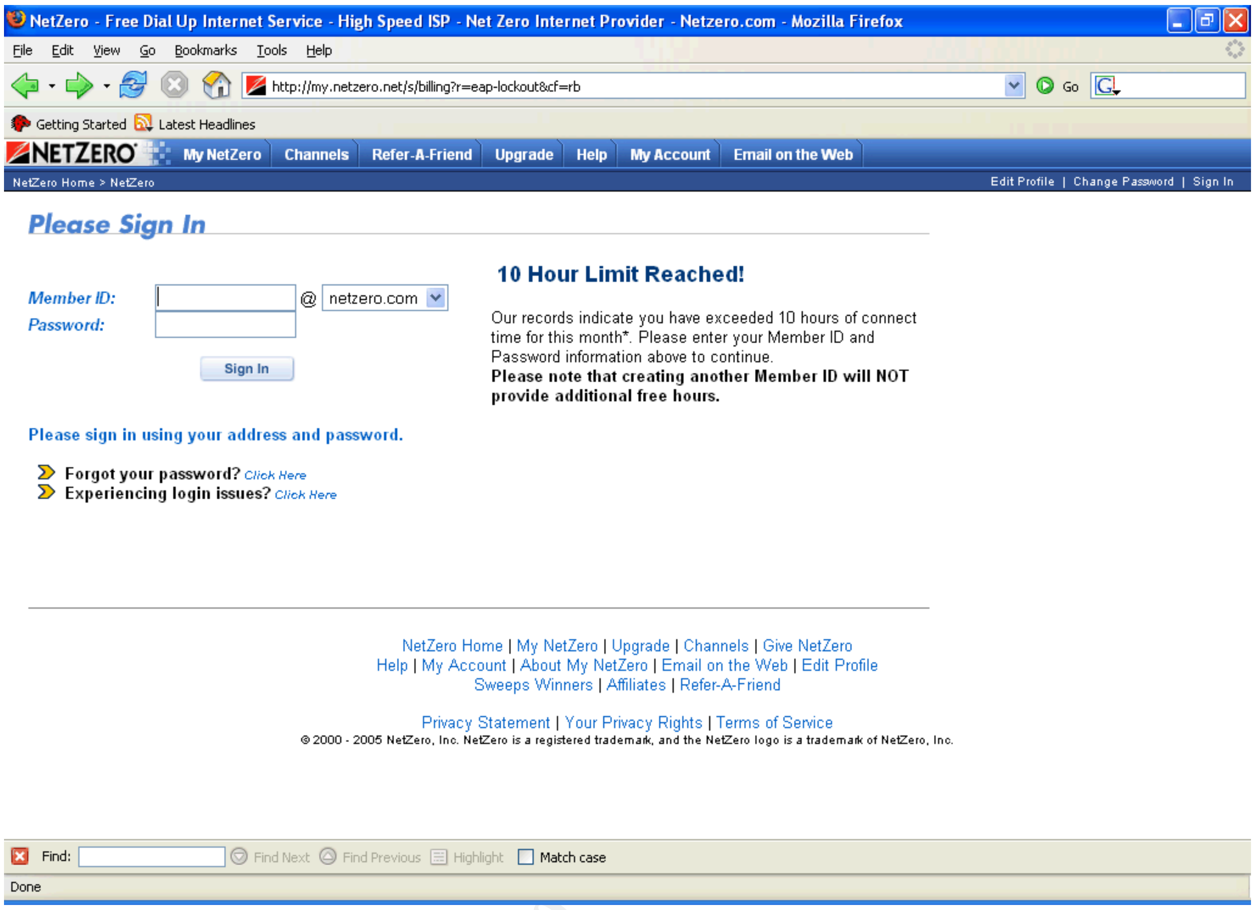
Malicous Activity

Host	DNS Name	Source
MY.NET.6.15	remedy.umbc.edu	Detect 1
MY.NET.190.202		Detect 1
MY.NET.190.203	wt-vpn2.umbc.edu	Detect 1
MY.NET.190.93		Detect 2
MY.NET.190.95		Detect 2
MY.NET.190.97		Detect 2
MY.NET.190.102		Detect 2
MY.NET.84.235		Detect 3
MY.NET.5.92		Detect 3
MY.NET.80.232		Detect 3
MY.NET.83.98		Detect 3
MY.NET.83.70		Detect 3
MY.NET.112.216		Detect 3
MY.NET.15.50		Detect 3
MY.NET.189.62		Detect 3
MY.NET.29.19	lyekka.umbc.edu	Detect 3
MY.NET.150.101	scholarseek.lib.umbc.edu	Detect 3
MY.NET.66.24		Detect 3
MY.NET.5.95		Detect 3
MY.NET.5.44	ndms.umbc.edu	Detect 3
MY.NET.112.226		Detect 3
MY.NET.29.8	cms.umbc.edu	Detect 3

MY.NET.150.44	illiad.lib.umbc.edu	Detect 3
MY.NET.5.67	ccrf.umbc.edu	Detect 3
MY.NET.111.72	Cuereims.umbc.edu	Detect 3
MY.NET.5.46	pp1.umbc.edu	Detect 3
MY.NET.5.45	cyclone.umbc.edu	Detect 3
MY.NET.5.20	centrelearn.umbc.edu	Detect 3
MY.NET.5.25	ehs.umbc.edu	Detect 3
MY.NET.11.3	dc2test.adtest.UMBC.EDU	Suspicious hosts-212.87.86.80
MY.NET.11.6	dc1.ad.UMBC.EDU	Suspicious hosts-212.87.86.80
MY.NET.11.7	dc2.ad.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.11.11	dc1test.adtest.UMBC.EDU	Suspicious hosts-212.87.86.80
MY.NET.12.5	ds.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.24.7	sluisvan.ucs.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.24.49	directory.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.24.65	fett.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.25.19	mr2.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.25.34	dengar.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.25.35	ig88.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.29.13	tcl1.cl.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.29.15	tcl2.cl.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.30.5		Suspicious hosts-212.87.86.80
MY.NET.30.6		Suspicious hosts-212.87.86.80
MY.NET.30.8		Suspicious hosts-212.87.86.80
MY.NET.30.9		Suspicious hosts-212.87.86.80
MY.NET.30.10		Suspicious hosts-212.87.86.80
MY.NET.30.11		Suspicious hosts-212.87.86.80
MY.NET.30.66	anubis.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.30.7		Suspicious hosts-212.87.86.80
MY.NET.70.5	captainamerica.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.75.13	chpdm.umbc.edu	Suspicious hosts-212.87.86.80
MY.NET.75.202		Suspicious hosts-212.87.86.80
MY.NET.5.5		Suspicious hosts-213.85.29.136
MY.NET.6.15	remedy.umbc.edu	Suspicious hosts-213.85.29.136
MY.NET.16.106	oncampus.vpn.umbc.edu	Suspicious hosts-213.85.29.136
MY.NET.16.90	Umbcvpn-delta.umbc.edu	Suspicious hosts-213.85.29.136
MY.NET.16.114	oncampus-private.vpn.umbc.edu	Suspicious hosts-213.85.29.136
MY.NET.190.X (subnet)		Suspicious hosts-213.85.29.136
MY.NET.25.73	mx8in.umbc.edu	Suspicious hosts-204.152.186.189

Appendix C

Visiting 64.136.21.233, the host identified in detect two as the target of a DoS attack, via Web browser resulted in the following NetZero page:


NetZero - Free Dial Up Internet Service - High Speed ISP - Net Zero Internet Provider - Netzero.com - Mozilla Firefox

http://my.netzero.net/s/billing?r=eap-lockout&cf=rb

Please Sign In

Member ID: @ netzero.com
Password:

Sign In

10 Hour Limit Reached!

Our records indicate you have exceeded 10 hours of connect time for this month*. Please enter your Member ID and Password information above to continue.
Please note that creating another Member ID will NOT provide additional free hours.

Please sign in using your address and password.

Forgot your password? Click Here
Experiencing login issues? Click Here

NetZero Home | My NetZero | Upgrade | Channels | Give NetZero
Help | My Account | About My NetZero | Email on the Web | Edit Profile
Sweeps Winners | Affiliates | Refer-A-Friend

Privacy Statement | Your Privacy Rights | Terms of Service
© 2000 - 2005 NetZero, Inc. NetZero is a registered trademark, and the NetZero logo is a trademark of NetZero, Inc.

Appendix D

Here are some of the SQL queries used to perform the analysis. Comments are included following the '--':

```
-- Unique source hosts from university network from all logs and number of instances
select A.SrcIP,sum(A."Total Instances") as "Total" from (
select      SrcIP, count(SrcIP) as "Total Instances"
from        ALERTS
where       (SrcIP like 'MY.NET.%')
group by    SrcIP
union all
select      SrcIP, count(SrcIP) as "Total Instances"
from        OOS
where       (SrcIP like 'MY.NET.%')
group by    SrcIP
union all
select      SrcIP, count(SrcIP) as "Total Instances"
from        SCANSMY
where       (SrcIP like 'MY.NET.%')
group by    SrcIP
```

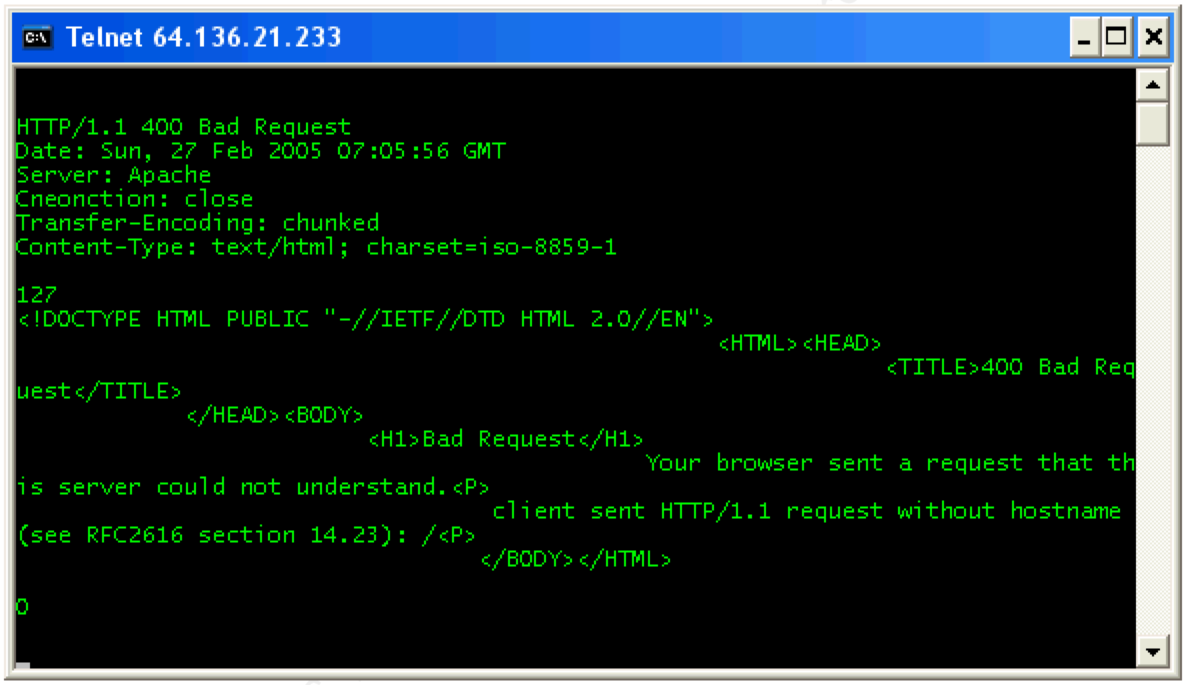
```
)A
group by      A.SrcIP
order by      A.Total desc
------

-- Get total instances of each university address from each table and sum them
select A.IP,sum(A."Total Instances") as "Total" from (
select        SrcIP as IP, count(SrcIP) as "Total instances"
from          ALERTS
where         (SrcIP like 'MY.NET.%')
group by      SrcIP
union all
select        SrcIP as IP, count(SrcIP) as "Total instances"
from          OOS
where         (SrcIP like 'MY.NET.%')
group by      SrcIP
union all
select        SrcIP as IP, count(SrcIP) as "Total instances"
from          SCANSMY
where         (SrcIP like 'MY.NET.%')
group by      SrcIP
union all
select        DstIP as IP, count(DstIP) as "Total instances"
from          ALERTS
where         (DstIP like 'MY.NET.%')
group by      DstIP
union all
select        DstIP as IP, count(DstIP) as "Total instances"
from          OOS
where         (DstIP like 'MY.NET.%')
group by      DstIP
union all
select        DstIP as IP, count(DstIP) as "Total instances"
from          SCANSMY
where         (DstIP like 'MY.NET.%')
group by      DstIP
)A
group by      A.IP
order by      A.Total desc
------

-- Dst Hosts and ports in all logs (port was identified at least 10 times)
select A.DstIP,A.DstPort,sum(A."Total Instances") as "Total" from (
select        DstIP,DstPort,count(DstPort) as "Total Instances"
from          ALERTS
where         (DstIP like 'MY.NET.%')
```

```
group by      DstIP,DstPort
union all
select        DstIP,DstPort,count(DstPort) as "Total Instances"
from          OOS
where         (DstIP like 'MY.NET.%')
group by      DstIP,DstPort
union all
select        DstIP,DstPort,count(DstPort) as "Total Instances"
from          SCANSMY
where         (DstIP like 'MY.NET.%')
group by      DstIP,DstPort
) A
group by      A.DstIP,A.DstPort
having        sum(A."Total Instances") > 9
order by      A.Total desc
------

--Find hosts involved in buffer overflow attempts in the scan logs
SELECT        * FROM SCANSMY WHERE (DstIP in (
select DstIP
from   ALERTS
where DstPort = 80
and    Description like 'EXPLOIT x86 NOOP'
group by      DstIP
)
or SrcIP in(
select DstIP
from   ALERTS
where DstPort = 80
and    Description like 'EXPLOIT x86 NOOP'
group by      DstIP
))
------

--dst ports - MY.NET. only
--Top Talkers
select A.Port,sum(A."Total Instances") as "Total" from (
select        DstPort as Port, count(DstPort) as "Total instances"
from          ALERTS
where         (DstIP like 'MY.NET.%')
group by      DstPort
union all
select        DstPort as Port, count(DstPort) as "Total instances"
from          OOS
where         (DstIP like 'MY.NET.%')
group by      DstPort
```

```
union all
select       DstPort as Port, count(DstPort) as "Total instances"
from         SCANSMY
where        (DstIP like 'MY.NET.%')
group by     DstPort
)A
group by     A.Port
order by     A.Total desc
------
```

Appendix E

This is a screenshot of a telnet to port 80 on 64.136.21.233 to attempt to identify the Web server version:



```
HTTP/1.1 400 Bad Request
Date: Sun, 27 Feb 2005 07:05:56 GMT
Server: Apache
Cneonction: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

127
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
                                                  <HTML><HEAD>
                                                              <TITLE>400 Bad Req
uest</TITLE>
            </HEAD><BODY>
                         <H1>Bad Request</H1>
                                             Your browser sent a request that th
is server could not understand.<P>
                                  client sent HTTP/1.1 request without hostname
(see RFC2616 section 14.23): /<P>
                                 </BODY></HTML>

0
```

Appendix F

These are the results of the whois queries for the suspicious external hosts:

	212.87.86.80		213.85.29.136		204.152.186.189
Inetnum	212.87.86.64 - 212.87.86.127	**inetnum**	213.85.29.0 - 213.85.29.255	**OrgName**	Internet Systems Consortium, Inc.
Netname	NOVATECH-UK	**netname**	RialCom-net	**OrgID**	ISC-94
Descr	Novatech Direct Limited	**descr**	Sverdlova 15	**Address**	950 Charter Street
Descr	Computer Supplies	**descr**	Podolsk, 142100,	**City**	Redwood City

Countr y	GB	**descr**	Russian Federation	**StateProv**	CA
admin-c	NP3220-RIPE	**countr y**	RU	**PostalCode**	94063
tech-c	CW494-RIPE	**admin-c**	YVK-RIPE	**Country**	US
rev-srv	usui.newnet.co.uk	**tech-c**	YVK-RIPE		
rev-srv	hayashi.newnet.co.uk	**status**	ASSIGNED PA "status:" definitions	**NetRange**	204.152.184.0 - 204.152.191.255
Status	ASSIGNED PA "status:" definitions	**notify**	noc@cnt.ru	**CIDR**	204.152.184.0/21
Remar ks	ADDRESS FOR ABUSE e-mail: luke.ashworth@novatech.co.uk	**mnt-by**	CNT-MNT	**NetName**	ISC-NET2
Notify	admin@newnet.co.uk	**chang ed**	noc@cnt.ru 20020819	**NetHandle**	NET-204-152-184-0-1
mnt-by	NEWNET-MNT	**chang ed**	vvss@cnt.ru 20030530	**Parent**	NET-204-0-0-0-0
Chang ed	nick@newnet.co.uk 20030611	**chang ed**	ip-dbm@ripn.net 20030530	**NetType**	Direct Allocation
Source	RIPE	**source**	RIPE	**NameServer**	NS-EXT.ISC.ORG
				NameServer	NS-EXT.LGA1.ISC.ORG
route	212.87.64.0/19	**route**	213.85.0.0/17	**NameServer**	NS-EXT.NRT1.ISC.ORG
descr	NewNet - Fast Access Internet	**descr**	CNT-network BLOCK	**NameServer**	NS-EXT.STH1.ISC.ORG
descr	Internet Service Provider	**origin**	AS8615	**Comment**	
origin	AS9191	**mnt-by**	CNT-MNT	**RegDate**	1997-02-26
notify	admin@newnet.co.uk	**chang ed**	noc@cnt.ru 20000531	**Updated**	2004-10-05
mnt-by	NEWNET-MNT	**source**	RIPE		
change d	peter@newnet.co.uk 20010127			**OrgAbuse Handle**	ISCAT-ARIN
source	RIPE	**person**	YURI V. KRIVITSKY	**OrgAbuse Name**	Internet Systems Consortium Abuse Team
		nic-hdl	YVK-RIPE	**OrgAbuse Phone**	+1-650-423-1300
person	Nick Petty	**addres s**	Yuri V. Krivitsky	**OrgAbuse Email**	abuse@isc.org
addres s	NewNet plc	**addres s**	Rial Com JSC.		
addres s	Cams Hall Estate	**addres s**	14 Bolshaya Serpuhovskaya str.	**OrgNOC Handle**	ISCN-ARIN

addres s	Fareham Hants	**addres s**	142100 Moscow reg. Podolsk	**OrgNOC Name**	Internet Systems Consortium NOC
addres s	PO16 8UJ UK	**phone**	+7 0967 680168	**OrgNOC Phone**	+1-650-423-1310
phone	+44 1329 226722	**fax-no**	+7 0967 571872	**OrgNOC Email**	noc@isc.org
fax-no	+44 1329 226721	**e-mail**	noc@rialcom.ru		
e-mail	ripe-abuse@newnet.co.uk	**chang ed**	noc@rialcom.ru 20020717	**OrgTechHan dle**	JDA87-ARIN
nic-hdl	NP3220-RIPE	**source**	RIPE	**OrgTech Name**	Damas, Joao
notify	admin@newnet.co.uk			**OrgTech Phone**	+1-650-423-1312
change d	peter@newnet.co.uk 20010221			**OrgTech Email**	Joao_Damas@isc.org
change d	nickpetty@newnet.co.uk 20011217				
change d	nick@newnet.co.uk 20020924			**OrgTech Handle**	JA39-ARIN
source	RIPE			**OrgTech Name**	Abley, Joe
				OrgTech Phone	+1-519-679-7573
person	Chris Wright			**OrgTech Email**	jabley@automagic.org
addres s	NewNet plc				
addres s	Cams Hall Estate				
addres s	Fareham Hants				
addres s	PO16 8UJ UK				
phone	+44 1329 226722				
fax-no	+44 1329 226721				
e-mail	ripe-abuse@newnet.co.uk				
nic-hdl	CW494-RIPE				
notify	admin@newnet.co.uk				
change d	nick@newnet.co.uk 20020927				
source	RIPE				