Global Information Assurance Certification Paper

Intrusion Analysis

GCIA

Practical Assignment

Version 4.1

Glenn R. Wemple
Local Mentor Program
10/19/2004 – 1/25/2005

Table of Contents

List of Figures

Abstract

Using the knowledge obtained by attending the SANS Local Mentor Led Intrusion Detection In-Depth class, I have completed an in-depth analysis of 3 day's worth of Snort logs. These sample logs were obtained from sensors located at the University of Maryland, Baltimore County. This is the only reference this paper will make to the university's identity. The IP addresses for the university have been obfuscated and are identified as MY.NET.xx.xx.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`filename`	Filenames, paths, and directory names are represented in this style.
`computer output`	The results of a command and other computer output are in this style
URL	Web URL's are shown in this style.
Quotation	A citation or quotation from a book or web site is in this style.

Introduction / Executive Summary

The University in an effort to improve its security posture has sought consultation in analyzing their network logs. These logs were all produced by a program called Snort. Snort is an open source freely available network intrusion prevention system.

Providing security in a university setting has many challenges. A university has to provide internet access to its students and faculty members as well as supporting and securing its critical business infrastructure.

By providing internet access to students and faculty the university in effect becomes an internet service provider. As an internet service provider, the university may be called upon by outside organizations such as the RIAA concerning copyright infringement. The logs show signs of peer-to-peer file sharing, which is usually an indication that copyrighted information is being traded. While blocking these services entirely may not be acceptable, a policy can be devised to limit their usage. File sharing usually results in a high amount of traffic being transmitted over the network. By limiting the amount of traffic any dorm computer can receive in a week file sharing can be curtailed.

Another implication of being an internet service provider is the lack of control over the patching and antivirus protection of many computers on the network. The logs show evidence of several compromised hosts on the network, which are actively scanning for more vulnerable hosts. The university may be able to decrease the number of unprotected hosts by purchasing a site license for an antivirus suite and making it freely available to the entire university community.

While there was no evidence that any critical business systems were compromised, several were scanned. This along with evidence of compromised student machines illustrates the importance of a defense-in-depth posture. Perimeter defenses are not only necessary to protect the business critical systems and the outside world but from the students' machines as well.

Immediate action is necessary in the case of the compromised hosts. These hosts are being controlled by outside sources and are within the perimeter defenses of the network. This can lead to more compromised hosts within the university's network. These hosts should immediately be taken off the network. Once off the network, they should be scanned for viruses and trojans. After this analysis, a decision should be made as to whether the systems need a full rebuild or not.

Firewall rules should be revisited in order to curtail the infections of hosts. It appears several hosts on the university network have been compromised using

Microsoft RPC vulnerabilities. These RPC ports along with other LAN protocols should be blocked at the perimeter.

Part II – Detailed Analysis

Log Files

The University has provided log files for the time period of 11/01/2003 through 11/03/2003. The log files used for analysis are as follows:

```
alert.031101
alert.031102
alert.031103
scans.031101
scans.031102
scans.031103
oos_report_031101
oos_report_031102
oos_report_031103
```

These log files were broken down by date and included Snort alert files, scan files and OOS (out-of-spec) files. There were some anomalies in these files, which made some records fall out of automated script processing. These records were identified and manually reviewed. Therefore, these records will not be represented when reporting aggregate counts.

There is a total 1,039,440 lines in the 3 alert log files. 811,890 of these lines were produced by the portscan preprocessor for Snort. This leaves us with 227,550 lines.

There were 1418 alert lines identified with anomalies. 106 of these contained portscan records, while 1312 did not. 1406 of these anomalies were alerts imbedded within other alerts. The remaining 6 records were incomplete. Thus, there are 226,238 alert records included in aggregate counts (226,238 + 1312 = 227,550).

There are a total of 12,378,932 records in the 3 scan log files. There were only 10 errors in these files and the remaining 12,378,922 records are available for aggregate reporting. These log files were not obfuscated in the same manner as the alert and oos log files. However, it was easy to identify the pattern used to obfuscate the IP addresses and I was able to do matching between all 3 log files.

The OOS log files for the period 11/1/2003 to 11/3/2003 were all identical and did not include the dates specified by their filenames. The dates included in the file were 10/27 to 10/29. There were 9195 records included in the OOS log file.

Analysis

© SANS Institute 2000 - 2005 Author retains full rights.

MY.NET Network Topology

The following assumptions concerning the network topology of the university's network were inferred by port usage and rule names. Hosts with an * have also been identified by Brett Hutley GCIA. Student networks were defined based on signs of online game play and file sharing activities.

Hosts:
MY.NET.1.3* – DNS Server
MY.NET.1.5* – DNS Server
MY.NET.1.200 – DNS Server
MY.NET.12.4 – Mail server
MY.NET.70.50* – Help Desk FTP Server
MY.NET.70.225 – Possible Web server
MY.NET.70.129 – Possible Web server
MY.NET.24.47* – FTP server
MY.NET.24.44 – Web server
MY.NET.30.3* – Novell server/Web Server
MY.NET.30.4* – Novell server/Web Server
MY.NET.12.6 – SMTP server
MY.NET.25.12 – SMTP server

Student Networks:
MY.NET.53.xx
MY.NET.97.xx
MY.NET.98.xx
MY.NET.163.xx

Detects

During the course of the 3 day period from November 1, 2003 through November 3, 2003 the university's snort sensors recorded 226,238 alerts. Of these 226,238 alerts there were 46 unique Snort alerts triggered. Following is a list of these alerts and their frequency.

Alert	Count
TCP SRC and DST outside network	178364
MY.NET.30.4 activity	14407
connect to 515 from inside	10466
MY.NET.30.3 activity	4642
SMB Name Wildcard	4170
EXPLOIT x86 NOOP	3253
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	2266
High port 65535 tcp - possible Red Worm – traffic	2058
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	2020

High port 65535 udp - possible Red Worm – traffic	1806
Incomplete Packet Fragments Discarded	643
NMAP TCP ping!	432
External RPC call	312
Possible trojan server activity	295
[UMBC NIDS] External MiMail alert	218
Null scan!	199
ICMP SRC and DST outside network	113
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	108
FTP passwd attempt	77
[UMBC NIDS IRC Alert] Possible drone command detected.	55
SUNRPC highport access!	54
Tiny Fragments - Possible Hostile Activity	41
SMB C access	39
FTP DoS ftpd globbing	37
TCP SMTP Source Port traffic	31
IRC evil - running XDCC	27
EXPLOIT x86 setuid 0	20
DDOS shaft client to handler	15
EXPLOIT x86 setgid 0	14
RFB - Possible WinVNC - 010708-1	12
[UMBC NIDS] Internal MiMail alert	8
EXPLOIT NTPDX buffer overflow	7
TFTP - Internal UDP connection to external tftp server	7
TFTP - External TCP connection to internal tftp server	6
Attempted Sun RPC high port access	3
connect to 515 from outside	3
External FTP to HelpDesk MY.NET.70.50	2
Probable NMAP fingerprint attempt	2
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	1
SYN-FIN scan!	1
[UMBC NIDS IRC Alert] K:lined user detected, possible trojan.	1
Samba client access	1
Bugbear@MM virus in SMTP	1
EXPLOIT x86 stealth noop	1
TOTAL	**226238**

TABLE 1

TCP SRC and DST outside network

Description:

The Snort alert “TCP SRC and DST outside network” is used to identify network traffic which is neither addressed to the home network nor addressed from the

home network. This alert can be triggered by an incorrect configuration of the HOME_NET variable within the snort.conf file, incorrectly configured network devices or by internal hosts spoofing there IP addresses.

Reason this detect was selected:
This detect was selected due the high frequency of alerts and it being a symptom of compromised hosts. 79% of the Snort alerts over the three day period were triggered by this rule. Considering that these alerts do not appear to be due to configuration errors compromised hosts will likely be found on the university's network.

Detect was generated by:

This detect was generated by a Snort NIDS running a custom rule to detect network traffic with a source IP address and destination IP address outside there network range. Here is a rule I tested on Snort 2.1.1 that would detect similar traffic:

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

Probability the source address was spoofed:
This alert is indicative of spoofed source addresses. There is no reason why network activity should be found on a network, which is neither addressed to or from that particular network. The only reason this type of activity would be found is due to incorrectly configured network devices or spoofing of source addresses.

Spoofed addresses are used in SYN flood denial of service attacks. A SYN flood is accomplished by creating excessive half-open connections, which can prevent the target from servicing additional requests. The half-open connections are created when a spoofed IP packet gets sent to the target with its SYN flag set. This causes the target machine to allocate memory in order to establish a connection. However, when the target machine tries to send its SYN-ACK packet, the second step in the 3-way handshake necessary for the TCP connection, it cannot reach the spoofed IP address. It will retry this transmission several times before timing out. In the mean time the attacker will send more and more spoofed SYN packets causing the target to allocate more resources.

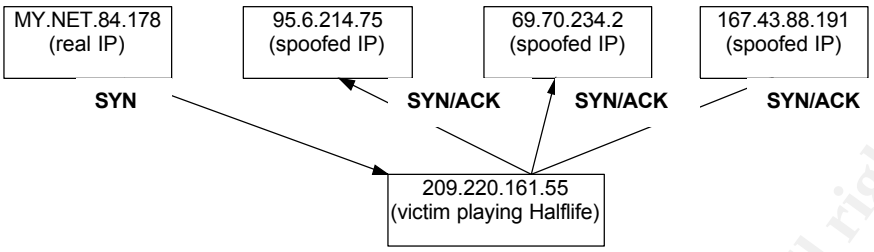

Figure 1: SYN Flood DOS using spoofed IP addresses

Attack Mechanism:
Internal hosts are launching denial of service attacks against hosts on the internet. There is a list of possibly compromised hosts in the appendix, which may have taken part in this attack. During this three day period this alert was triggered by 71628 unique source IP addresses destined for 30 unique destinations.

There are several interesting correlations between IRC drone commands and the start of DOS attacks against external hosts. Here are a few Snort alerts, which occur just prior to the DOS attacks in table 2.

```
11/01-18:58:43.200907  [**] [UMBC NIDS IRC Alert] Possible drone
command detected. [**] 144.37.2.10:6667 -> MY.NET.84.178:1071

11/01-19:48:19.948429  [**] [UMBC NIDS IRC Alert] Possible drone
command detected. [**] 144.37.2.10:6667 -> MY.NET.84.178:1071

11/01-21:57:46.392326  [**] [UMBC NIDS IRC Alert] Possible drone
command detected. [**] 144.37.2.10:6667 -> MY.NET.84.178:1071
```

Count	DST IP	DST Port	Duration	Service
11625	199.43.172.14	80	2003-11-01 19:01:04 - 2003-11-01 19:01:14	Web
21964	209.220.161.55	27015	2003-11-01 19:48:22 - 2003-11-01 19:50:53	Half-life
780	128.146.240.206	80	2003-11-01 21:58:12 - 2003-11-01 21:58:33	Web

TABLE 2

Correlations:
More information concerning DOS attacks can be found in CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks.
http://www.cert.org/advisories/CA-1996-21.html

Evidence of active targeting:
Compromised machines on the university's network were targeted and ordered to take part in the denial of service attacks. The hosts on the receiving end of the denial of service attack are definitely being targeted.

Severity:
Severity=(criticality + lethality) – (system countermeasures + network countermeasures)

Criticality	3	Network traffic can overwhelm IDS and other network devices on the university's network.
Lethality	4	Compromised hosts
System Countermeasures	1	Hosts not up to date with patches or AV protection
Network Countermeasures	3	Egress filtering at border routers will stop spoofed traffic from reaching internet. However, it is likely that this traffic has already passed internal routers.
Severity = 3 = (3+4) – (1+3)		

Corrective Measures:
Make sure routers use egress filtering to stop spoofed IP addresses from leaving the local LAN. There is a good paper on egress filtering in the SANS Reading Room http://www.sans.org/rr/whitepapers/networkdevs/1536.php.

Detect and disconnect all compromised hosts taking part in DOS attacks. NetFlow statistics from the router and switches can be used to help identify hosts that took part in these DOS attacks.

More information on NetFlow available at:
http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/products_implementation_design_guide09186a00800d6a11.html

[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC

Description:
Symantec Security Response gives the description below of the Sdbot Trojan:

> *Backdoor.Sdbot is a Backdoor Trojan horse that allows the Trojan's creator to control a computer by using Internet Relay Chat (IRC). Backdoor.Sdbot can update itself by checking for newer versions over the Internet.*

Backdoor.Sdbot contains its own IRC client, allowing it to connect to an IRC channel that was coded into the Trojan. Using the IRC channel, the Trojan listens for the commands from the Trojan's creator. The creator of the Trojan accesses the Trojan by using a password-protected authorization.

Reason this detect was selected:
This detect indicates the possibility of compromised hosts on the university's network. Compromised hosts located within the organization can be used to launch attacks against other internal hosts bypassing any perimeter protection.

Another reason this detect was selected was the possibility of its role in the high number of "TCP SRC and DST outside network" alerts found in the alert logs. A host with the Sdbot Trojan can be remotely controlled via IRC to launch DOS attacks. A SYN flood DOS attack is more efficient when spoofed addresses are used. A very likely cause of the alert "TCP SRC and DST outside network" is source address spoofing.

Detect was generated by:
A custom Snort rule developed either for or by the university. Here is a Snort rule, which also detects Sdbot floodnet retrieved from http://coders.meta.net.nz/~perry/irc.rules.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 6660:7000 \
(content: "USER ";\
content: " 0 0 "; nocase;\
msg: "Possible sdbot floodnet detected attempting to IRC";\
classtype:misc-activity;)
```

Probability the source address was spoofed:
There appears to be two-way communications between the suspected host MY.NET.84.178 and 144.37.2.10. Although a Mitnick attack is possible it is unnecessary considering that the attacker already has a middle-man in IRC. Thus, source address spoofing is not likely.

```
11/01-17:35:05.358523  [**] [UMBC NIDS IRC Alert] Possible sdbot
floodnet detected attempting to IRC [**] MY.NET.84.178:1071 ->
144.37.2.10:6667
11/01-17:50:35.111895  [**] [UMBC NIDS IRC Alert] Possible drone
command detected. [**] 144.37.2.10:6667 -> MY.NET.84.178:1071
```

Attack Mechanism:
Sdbot is a trojan which by itself has no replication mechanism for infecting hosts. Thus, it is usually transmitted via peer-to-peer networks, IRC, newsgroups etc. However, this trojan is also bundled with many worms. The Sdbot variants associated with worms replicate via Windows file shares and various Windows exploits. (http://vil.nai.com/vil/content/v_99410.htm)

Hosts MY.NET.112.179 and MY.NET.84.178 are actively scanning ports 135 and 445. Port 135 is associated with the Microsoft DCOM Service Control Manager. Port 445 would be indicative of a worm trying to propagate via Windows file shares. Port 135 has been victim of buffer overflow exploits, which have been exploited by variants of the Sdbot worm. This activity depicted in the table below shows probable infection of these hosts.

Count	SRC IP	DST Port
112277	MY.NET.112.179	445
112441	MY.NET.112.179	135
549713	MY.NET.84.178	445
552086	MY.NET.84.178	135

Correlations:
http://securityresponse.symantec.com/avcenter/venc/data/backdoor.sdbot.html
http://vil.nai.com/vil/content/v_100454.htm

CERT® Advisory CA-2003-19 Exploitation of Vulnerabilities in Microsoft RPC Interface at http://www.cert.org/advisories/CA-2003-19.html deals with Microsoft RPC vulnerabilities on ports 135 and 445. This advisory was posted July 31, 2003, which is only 3 months prior to this event.

Evidence of active targeting:
As far as the worm replication, there does not appear to be active targeting. The noisy scans of these hosts suggest that they are not targeting specific hosts. However, these hosts appear to be dialing home to an IRC channel. After the initial connection between MY.NET.84.178 and IRC the host triggers several "[UMBC NIDS IRC Alert] Possible drone command detected" indicating that it may be controlled remotely via IRC.

Severity:
Severity=(criticality + lethality) – (system countermeasures + network countermeasures)

Criticality	3	Student desktops
Lethality	5	Machines are compromised and being commanded via IRC
System Countermeasures	1	Hosts not up to date with patches or AV protection
Network Countermeasures	3	The university provides perimeter packet filtering and runs an IDS
Severity = 4 = (3+4) – (1+3)		

Corrective Measures:
The hosts MY.NET.84.178 and MY.NET.112.179 appear to be infected and are actively scanning for more hosts to infect. These compromised hosts should immediately be disconnected from the network to minimize the spread of the trojan. Due to the nature of this trojan a full system rebuild is recommended for compromised machines.

EXPLOIT x86 NOOP

Description:
The "EXPLOIT x86 NOOP" rule is configured to detect buffer overflow attacks against the x86 architecture. Buffer overflow attacks take advantage of unchecked variables which allow overwriting the stack's return pointer. This can allow an attacker to run commands on a victim's machine with the privileges of the exploited program.

An attacker will most likely not know the exact location of the return pointer on the stack. The data located at the return pointer must be executable or the program will crash. For this reason attackers will use a collection of no operation commands called a noop sled in order to provide some room for error when overwriting the return pointer. On an x86 machine, the noop command is typically represented by hex 0x90. Thus, Snort rules written to detect this type of behavior will look for multiple instances of 90 in the packet's data. (http://www.phrack.org/show.php?p=49&a=14)

This rule is subject to false positives when users download image files or executables, which may use the hex value 0x90 for padding.

Reason this detect was selected:
This detect was selected as it appears several hosts have been successfully compromised utilizing a buffer overflow attack.

Detect was generated by:
This detect was generate by a Snort NIDS using a rule similar to the following, which was obtained from the default rules for Snort 2.1.1:

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90
90 90 90 90 90|"; depth: 128; reference:arachnids,181;
classtype:shellcode-detect; sid:648; rev:6;)
```

The actual Snort rule used by the university probably didn't use the $SHELLCODE_PORTS variable or the $SHELLCODE_PORTS variable was not set to !80 to exclude web traffic. This is evident by the number of alerts with external hosts communicating over port 80 to ephemeral ports on local hosts.

Probability the source address was spoofed:
The probability that spoofed source addresses were used is unlikely. There are

signs of 2-way communications between the attackers and compromised hosts.

Attack Mechanism:
The majority of attacks were against ports 80, 135 and 445. These ports are all associated with known Microsoft Windows buffer overflow vulnerabilities at the time of the attack.

Correlations:

Microsoft Security Bulletin MS03-007, released March 17, 2003, reports a buffer overflow vulnerability in WebDAV, which listens on port 80.
http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx

Microsoft Security Bulletin MS03-039, posted on September 10, 2003, refers to a buffer overflow vulnerability in DCOM. These vulnerabilities can be exploited on ports 135 and 445.
http://www.microsoft.com/technet/security/bulletin/MS03-039.mspx
There are also a couple candidate CVE entries for these buffer overflows.
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0715
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0528

Evidence of active targeting:
The hosts MY.NET.190.101, MY.NET.190.102, MY.NET.29.24 and MY.NET.24.19 have by far the greatest number of overflow attempts against them. Most likely some reconnaissance was done prior to the attack in order to single out these hosts. The following table shows the amount of "EXPLOIT x86 NOOP" alerts for each of these hosts by port.

Count	Host	Port
294	MY.NET.29.24	80
310	MY.NET.29.19	80
502	MY.NET.190.101	135
525	MY.NET.190.102	135
919	MY.NET.190.102	445

After the "EXPLOIT x86 NOOP" alerts for host MY.NET.190.101 we see some alerts indicating that this attack was successful. Most alarming it appears that the host 81.250.38.61 successfully installed the SubSeven trojan on this host. This is indicated by the following sequence of alerts:

```
11/02-06:00:05.957703  [**] Samba client access [**]
81.250.38.61:3298 -> MY.NET.190.101:139
11/02-06:01:16.121095  [**] External RPC call [**] 81.250.38.61:3348 -
> MY.NET.190.101:111
11/02-06:01:18.125279  [**] Possible trojan server activity [**]
81.250.38.61:3434 -> MY.NET.190.101:27374
11/02-06:01:18.128037  [**] Possible trojan server activity [**]
```

```
MY.NET.190.101:27374 -> 81.250.38.61:3434
```

MY.NET.190.102 also shows signs of compromise as indicated by the "SMB C access", "SMB Name Wildcard", and "Possible Trojan server activity". Although, it does not appear that the host 68.112.250.127

The attacks on hosts MY.NET.29.24 and MY.NET.29.19 do not appear to have been successful. The only alerts generated by these hosts were the "EXPLOIT x86 NOOP" alerts. These hosts also do not appear as source IP addresses in the scan logs.

Severity:
Severity=(criticality + lethality) – (system countermeasures + network countermeasures)

Criticality	3	Student desktops
Lethality	5	Machines are compromised and show signs of trojans
System Countermeasures	1	Hosts not up to date with patches
Network Countermeasures	2	The university's perimeter packet filtering is too liberal but runs an IDS which made the detect.
Severity = 5 = (3+4) – (1+3)		

Corrective Measures:
Block the ports 135 and 445 at the perimeter. It would also be recommended to block ports 137,138 and 139 at the perimeter, as these are NetBIOS ports, which can be used for reconnaissance. This should aid in preventing more hosts from being exploited by the same vulnerabilities.

The ports above along with a list of other common Microsoft server ports can be found at:
http://www.microsoft.com/technet/security/smallbusiness/topics/serversecurity/ref_net_ports_ms_prod.mspx

Disconnect compromised hosts at once and determine whether they can be cleaned or if a rebuild is necessary. As part of either the clean-up or rebuild process, ensure that the systems are fully patched.

Network Statistics

Top Talkers

Alert:

SRC IP	Count
MY.NET.162.41	10468
67.21.63.15	4021
MY.NET.11.6	2670
67.117.44.87	2229
MY.NET.15.198	1892

MY.NET.162.41 is responsible for 10466 “connect to 515 from inside” alerts and 2 “SMB Name Wildcard”. All 10466 “connect to 515 from inside” alerts are triggered from port 721 to 128.183.110.242 (tek924.gsfc.nasa.gov) port 515 and persisted over all 3 days analyzed. Port 515 is associated with printer spooler and port 721 is associated with lpr, which is also printer related. Could a professor have a computer configured to print at NASA? Maybe an adjunct professor who works for NASA? Anyway the traffic does not appear to be malicious in nature. A similar detect can be found in Martin Vanborenbeek’s GCIA practical located at http://www.daemon.be/~maarten/Maarten_Vanhorenbeeck_GCIA.pdf.

The host 67.21.63.15 is the source IP for 164 “MY.NET.30.3 activity” alerts and 3857 “MY.NET.30.4 alerts”. Both of these alerts are related to the university’s Novell servers. According to the ARIN WHOIS database, 67.21.63.15 is owned by Adelphia Cable Communications. This information along with the fact that all the destination ports for this alert, 80, 524, and 51443 are associated with normal activity I see no malicious intent. This traffic is probably due to a member of the university community accessing their files from their home cable modem.

All 2670 alerts associated with MY.NET.11.6 are a result of the “SMB Name Wildcard” Snort rule. All of these alerts are destined for the 169.254.0.0 IP address, which is a reserved link local address defined in RFC 3330. According to RFC, this IP address should only be used for link local communications. This leads me to believe that a network sensor feeding the Snort NIDS is connected to the same link as this host.

67.117.44.87 appears in the alert logs 2229 times associated with the “MY.NET.30.4 activity” custom Snort rule. All traffic reported is aimed at ports 80 and 51433, which are associated with normal use of the MY.NET.30.4 system. According to the ARIN WHOIS database, 67.117.44.87 belongs to Pac

Bell Internet Services. Similar to the 67.21.63.15 detect earlier this probably just due to someone connecting to their files from their home ISP.

MY.NET.15.198 appears in the alert logs 3779 times. Of those alerts, "[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC" appears 1892 times. All occurrences of this alert have the destination IP 64.157.246.22 and destination port 6667. As per dshield.org this IP is now a Level 3 Communications DNS server. The other 1887 alerts are "[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan." For which MY.NET.15.198 is the destination IP.

Here are some Snort rules from http://coders.meta.net.nz/~perry/irc.rules which may be responsible for the aforementioned alerts.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 6660:7000 \
(content: "USER ";\
content: "dcc"; nocase;\
msg: "XDCC client detected attempting to IRC";\
classtype:misc-activity;)

alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any \
(content: "ERROR \:Closing Link\: "; nocase;\
msg: "IRC user /kill detected, possible trojan.";\
classtype:misc-activity;)
```

DST IP	Count
69.65.5.24	77655
66.90.76.113	36815
65.95.70.249	28041
209.220.161.55	21964
MY.NET.30.4	14407

The hosts 69.65.5.24, 66.90.76.113, 65.95.70.249, and 209.220.161.55 were all victims in a denial of service attack using spoofed source IP addresses. All alerts related to these hosts are "TCP SRC and DST outside network" for which they are the destination host.

MY.NET.30.4 is one of the university's Novell servers and a custom rule was written, titled "MY.NET.30.4 activity", in order to detect any traffic to this host from a host outside the home network. I was able to guess the OS on this machine by looking at the ports and their frequency of occurrence. I saw a lot of port 80 (web), 524 (ncp - Netware Core Protocol), and 51443 which I couldn't find in any of the usual port databases. A Google search of port 51443 brought me first to a University of Georgia Novell listserv, which mentioned it being a default port for Novell Netstorage. http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0212&L=ugnovell&F=&S=&P=785. Further searching led me to GCIA practical 738, which echoed my assumption.

MY.NET.30.4 was only involved in alerts being triggered by the custom Snort rule
"MY.NET.30.4 activity" and was always the destination IP address.

Scan:

SRC IP	Count
MY.NET.1.200	1824459
MY.NET.70.129	1797806
MY.NET.70.225	1751289
MY.NET.163.107	1398074
MY.NET.84.194	1389824

MY.NET.1.200 scanned port 53 (DNS) 1,821,737 times and port 123 (NTP) 2722 times. This appears to be legit traffic for a DNS server. Maybe the university is testing a new DNS server and hasn't yet updated their snort.conf file to ignore this server for the portscan preprocessor. Another sign that this is legitimate traffic is that most of the traffic on port 123 (network time protocol) is headed toward registered NTP servers. The IP 192.5.5.250 resolves to clock.isc.org which is listed in the following Microsoft article as a NTP server:
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q262680
The IP 128.4.1.2 resolves to mizbeaver.ude1.edu and is identified as a time server, in a PDF written by the University of Delaware.
http://www.eecis.udel.edu/~mills/database/brief/dcnet/dcnet.pdf

The hosts MY.NET.70.129 and MY.NET.70.225 are scanning ports 135 and 80 at an alarming rate. Most likely these hosts are infected with a worm, which tries to exploit the WebDAV and DCOM buffer overflow vulnerabilities on Windows. These vulnerabilities are described in Microsoft Security Bulletin MS03-039 and Microsoft Security Bulletin MS03-007. For this reason these hosts will be added to the list of compromised hosts in the appendix.

The hosts MY.NET.163.107 and MY.NET.84.194 are actively scanning for hosts listening on port 135. This is probably another worm, which tries to exploit DCOM buffer overflow vulnerabilities on Windows mentioned in Microsoft Security Bulletin MS03-039. These hosts will also be added to the list of compromised hosts in the appendix.

DST IP	Count
192.26.92.30	62677
131.118.254.33	32710
192.55.83.30	30810
130.94.6.10	25031
203.20.52.5	24997

The hosts 192.26.92.30, 131.118.254.33, 192.55.83.30, 130.94.6.10, and 203.20.52.5 appear to be DNS servers. The two machines scanning these IP addresses are university DNS servers MY.NET.1.5 and MY.NET.1.200. All scans are destined to port 53.

Top Targeted Ports

Alert:

DST Port	Service	Count
80	Web	119593
6667	IRC	30072
27015	Half-life	21964
53	DNS	10537
515	Spooler	10469

Scan:

DST Port	Service	Count
135	RPC	7132778
53	DNS	2146094
80	Web	739141
445	SMB	666464
6257	WinMx	324472

Suspicious Hosts - External

The hosts 67.86.211.219, 206.14.191.84 and 81.250.38.61 displayed suspicious activities during the monitoring period.

The host 67.86.211.219 is scanning the MY.NET.190.xx network for ports associated with SubSeven and Kuang2. It gets two hits with MY.NET.190.202 and MY.NET.203. Looking up port 17300 at http://isc.sans.org reveals an article by Rick Ballard mentioning a trojan named “Milkit”. More information regarding “Milkit” can be found at http://www.lurhq.com/sig-milkit.html.

The IP address 206.14.191.84 is associated with several Snort alerts. These alerts include “Tiny Fragments - Possible Hostile Activity”, “Null scan!”, and “SYN-FIN scan!”. These alerts are actively targeting 3 hosts on the MY.NET network. It appears that this host is attempting to collect information concerning the hosts MY.NET.97.40, MY.NET.97.20, and MY.NET.53.54. Typically null scans and syn-fin scans are used in OS fingerprinting attempts. This is a rather stealthy scan as it seems to target one IP per day and only scans for a very brief amount of time. This host should be put on a watch list to see if it attempts any future attacks. Looking this host up on http://www.dshield.org shows that the IP is registered to Verio, Inc. and that it has 189 records against it.

The most interesting host is 81.250.38.61. This host targeted one host MY.NET.190.101 and appears to have remote control of it. The first alert for this host came at 6:00:05 on November 2, when it used SAMBA to connect to a file share. It then scans the host on select ports, setting off a few other Snort alerts on its way. At 6:01:18 it hits port 27374, which is listening. It then makes several exchanges with MY.NET.190.101 on this port. During the scanning phase different ports were used for each SYN or UDP attempt. When it communicates, back and forth with MY.NET.190.101 on port 27374 it uses the same source port 3434 which is evidence of an ongoing connection.

Analysis Process

Tools:

Hardware: Toshiba Satellite Pro 4600 PIII 512MB RAM 30GB HD
OS: Suse Linux 9.1
Database: MySQL 4.1.9-standard
Web Server: Apache 2.0.49-27.18.3 with PHP 4.3.4-43.25
Scripts: Perl 5.8.3-24.2, PHP, SnortSort .03
Other: GNU Grep 2.5.1-416, wc 5.2.1

Data Mining Environment:

Nobody wants to reinvent the wheel. Thus, I set out to find the best free tools available to perform my analysis and researched how other analysts succeeded. During my search I ended up trying Snort-Sort, SnortSnarf and Jason Lam's Perl script. Snort-Sort was efficient at making the alert files more readable but was not suitable for an in-depth analysis. SnortSnarf never completed processing the alert log files but it did turn my laptop into a space heater. It was at this point when I decided a relational database would be the best way to perform an in-depth analysis. Using Jason Lam's Perl script as an example, I was able to build a few Perl scripts to import both the alert and scan log files into MySQL. These scripts along with the database creation statements are included in the appendix.

> Pitfall: The version of MySQL included with SuSe 9.1 was version 4.0.18-32, which did not yet support the function str_to_date(). I wanted to use this function to aid in the importing of the scan log files. Thus, I upgraded MySQL to the latest stable release 4.1.9.

> Pitfall: Corruption in the log files caused some records to error out. In order, to be able to account for all records and know what was imported I used a couple grep filters to separate the exceptions from the clean data. These grep filters are included in the appendix.

Now I love repetitive work just as much as the next guy so in order to thwart typing the same SQL statements over and over I developed a few web pages using PHP to do the work for me. The web pages gave me a way to drill down into the data in a way consistent with my analysis technique.

Process:

In analyzing the data, I took a two pronged approach. First I focused on the alert data. In doing so, I began with the custom Snort rules created by the University. This immediately illuminated several hosts, which the university was trying to protect along with several attacks that the university has experience previously. After processing the custom alerts, I went through the other alerts compiling as much information as possible in order to determine, which alerts required further

analysis.

The second approach involved looking at toptalker stats. While looking at the toptalker stats I was able to identify hosts, which I believed should be offering services versus those hosts which were most likely compromised scanning for other vulnerable hosts. Other than the custom Snort rules, this was probably the most efficient way of identifying the network topology.

Sites:
http://isc.sans.org
http://dshield.org
http://www.geektools.com/whois.php
http://www.treachery.net/tools/ports/lookup.cgi
http://www.securityfocus.com
http://www.whitehats.com/
http://www.cert.org/
http://vil.nai.com/vil/default.asp?wt.mc_n=us_entqlsearchvil&wt.mc_t=ext_li_con&cid=10368
http://www.google.com

Appendix

Log file corruption examples

```
11/01-12:14:28.372244  [**] MY.NET.30.4 activity [**]
67.21.63.15:1043 -> MY.NET.30.4
```
11/01-12:33:45.220591 [**] spp_portscan: portscan status from MY.NET.84.194: 20 connections across 20 hosts: TCP(20), UDP(0) [**]
```
:524

11/02-18:37:13.531529  [**] TCP SRC and DST outside network
[**] 204.26.108.250
```
11/02-19:03:22.282647 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.84.194 (THRESHOLD 12 connections exceeded in 0 seconds) [**]
```
:1805 -> 65.95.70.249:6667
```

Note the bold entries above, which were embedded in other entries.

Separate Out Corruption

```
grep -n "^11/0.*11/0" alert.031101 > exceptions.alert.031101
grep -n "^11/0.*11/0" alert.031102 > exceptions.alert.031102
grep -n "^11/0.*11/0" alert.031103 > exceptions.alert.031103
```

Using the grep commands above I created an exceptions file for each alert file. This exceptions file was comprised of lines, which had multiple dates which indicated multiple alerts on one line. By using the –n option I am including line numbers in my exceptions file.

```
grep -n -v "^11/0" alert.031101 >> exceptions.alert.031101
grep -n -v "^11/0" alert.031102 >> exceptions.alert.031102
grep -n -v "^11/0" alert.031103 >> exceptions.alert.031103
```

Using the grep commands above I appended the incomplete alert lines to the exceptions file for each alert file. Once again, I use –n option to include line numbers.

```
sort -n -t: exceptions.alert.031101 >
sort.exceptions.alert.031101
sort -n -t: exceptions.alert.031102 >
sort.exceptions.alert.031102
sort -n -t: exceptions.alert.031103 >
sort.exceptions.alert.031103
```

In order to put the exceptions files back in order I use the sort command. Considering that each line is now preceded by the line number and a colon I use the –n option for numerical sort and the –t: option to use the semicolon as a delimiter. At this point, we have a complete exceptions file for each alert log file.

```
grep "^11/0" alert.031101 | grep -v "^11/0.*11/0" >
cleanalert.031101
grep "^11/0" alert.031102 | grep -v "^11/0.*11/0" >
cleanalert.031102
grep "^11/0" alert.031103 | grep -v "^11/0.*11/0" >
cleanalert.031103
```

The above grep commands are used to create alert log files clean of any detected corruption.

MySQL Commands

These commands create the MySQL database structure.

```
create table scans (timestamp datetime, srcip varchar(15),
srcport int(5),dstip varchar(15), dstport int(5),scan
varchar(3),other varchar(25));

create table alert (timeof varchar(30),action varchar(100),
srcip varchar(15), srcport integer(5),dstip varchar(15), dstport
integer(5));

create table portscan (timeof datetime,srcip
varchar(15),totaltime varchar(5),nohost varchar(6),tcpno
varchar(6), udpno varchar(6));

alter table alert add index idxAlertAction (action);
alter table scans add index idxScansSrcip (srcip);
alter table scans add index idxScansDstip (dstip);
alter table scans add index idxScansSrcport (srcport);
alter table scans add index idxScansDstport (dstport);
alter table alert add index idxAlertSrcport (srcport);
alter table alert add index idxAlertDstport (dstport);
```

AlertParser.pl

The following perl script was based off one created by Jason Lam.

```
#!/usr/bin/perl -w
use DBI;
my $dsn = "DBI:mysql:host=localhost;database=gcia";
my $dbh = DBI-> connect($dsn, "<username>", "<password>")
     or die "Cannot connect to mysql: $!\n";
$fh="cleanalert.031103";
open(ALERT,$fh) or die "Can't open file $fh: $!\n";
while (defined ($line = <ALERT>)){
if ($line=~/(\d+)\/(\d+)-
(\d+:\d+:\d+).\d+\s+\[\*\*\]\sspp_portscan: End of portscan
from\s(.*):\sTOTAL
time\((.*\w)\)\shosts\((\d+)\)\sTCP\((\d+)\)\sUDP\((\d+)\)\s\[\*
```

```
\*\]/){
#\((d+)\)\sTCP\((\d+)\)\sUDP\((\d+)\) \[\*\*\]/) {
#print "$line\n";
#print "2003-$1-$2 $3 srcip $4 time $5 hosts $6 tcp $7 udp
$8\n";
$SQL="insert into portscan values('2003-$1-$2
$3','$4','$5','$6','$7','$8');";
$dbh->do($SQL);
}
elsif ($line=~/(\d+)\/(\d+)-
(\d+:\d+:\d+).\d+\s+\[\*\*\]\s(.*)\s\[\*\*\]\s(.*):(\w+)\s-
>\s(.*):(\w+)/) {
#print "2003-$1-$2 $3 alert: $4 srcip: $5 srcport: $6 dstip: $7
dstport: $8\n";
$SQL="insert into alert values('2003-$1-$2
$3','$4','$5',$6,'$7',$8);";
$dbh->do($SQL) or print $SQL;
}
elsif ($line=~/(\d+)\/(\d+)-
(\d+:\d+:\d+).\d+\s+\[\*\*\]\s(.*)\s\[\*\*\]\s(.*)\s->\s(.*)/) {
#print "2003-$1-$2 $3 alert: $4 srcip: $5 dstip: $6\n";
$SQL="insert into alert(timeof,action,srcip,dstip) values('2003-
$1-$2 $3','$4','$5','$6');";
$dbh->do($SQL) or print $SQL;
}
elsif ($line!~/portscan/) {
print $line;
}
}
close(ALERT);
$dbh->disconnect();
print "done.\n";
exit(0);
```

ScanParser.pl

This script was used to import scan logs into MySQL.

```
#!/usr/bin/perl -w
use DBI;
my $dsn = "DBI:mysql:host=localhost;database=gcia";
my $dbh = DBI-> connect($dsn, "<username>", "<password>")
        or die "Cannot connect to mysql: $!\n";
$fh="scans.031103";
open(SCANS, $fh) or die "Can't open file $fh: $!\n";
$count=0;
while (defined ($line = <SCANS>)){
$count++;
if ($count % 100000 == 0){
print "$count records processed\n"}
if ($line=~/(\w+\s+\d+\s\d+:\d+:\d+)\s(.*):(\d+) ->
(.*):(\d+)\s(\w+)\s(.*)/) {
     $SQL= "insert into scans values(str_to_date('2003-$1','%Y-
```

```
%b %e %T'),'$2','$3','$4','$5','$6','$7');";
      $dbh->do($SQL);}
elsif ($line=~/(\w+\s+\d+\s\d+:\d+:\d+)\s(.*):(\d+) ->
(.*):(\d+) (\w+)/) {
      $SQL= "insert into scans values(str_to_date('2003-$1','%Y-
%b %e %T'),'$2','$3','$4','$5','$6',null);";
      $dbh->do($SQL);}
else {
print $line;}
}
close(SCANS);
$dbh->disconnect();
print "hey" . "\n";
exit(0);
```

Compromised Hosts

MY.NET.70.129
MY.NET.70.225
MY.NET.84.178
MY.NET.84.194
MY.NET.112.179
MY.NET.163.107
MY.NET.190.101
MY.NET.190.102
MY.NET.190.202
MY.NET.190.203

References

Ballard, Rick. "User Comment – Port 17300" SANS Internet Storm Center. 2 May 2003. 12 March 2005. <http://isc.sans.org/show_comments.php?id=241>

"CAN-2003-0528" Common Vulnerabilities and Exposures 8 July 2003. 19 March 2005. <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0528>

"CAN-2003-0715" Common Vulnerabilities and Exposures 2 Sept. 2003. 19 March 2005. <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0715>

"CERT® Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks" CERT Coordination Center 29 Nov. 2000. 3 March 2005 <http://www.cert.org/advisories/CA-1996-21.html>

"CERT® Advisory CA-2003-19 Exploitation of Vulnerabilities in Microsoft RPC Interface" CERT Coordination Center 31 July 2003. 17 March 2005 <http://www.cert.org/advisories/CA-2003-19.html>

"Egress Filtering For a Better Internet" SANS Reading Room 7 Sept. 2004.3 March 2005 <http://www.sans.org/rr/whitepapers/networkdevs/1536.php>

Hutley, Brett. "GIAC Certified Intrusion Analyst Practical Assignment" Global Information Assurance Certification 22 September 2004. 3 March 2005 <http://www.giac.org/certified_professionals/practicals/gcia/0775.php>.

"IRC-Sdbot" McAfee Virus Information Library 7 March 2003.12 March 2005.<http://vil.nai.com/vil/content/v_99410.htm>

"A List of the Simple Network Time Protocol Time Servers That Are Available on the Internet" Microsoft Help and Support 13 Oct 2003. 15 March 2005. <http://support.microsoft.com/default.aspx?scid=kb;EN-US;q262680>

"Microsoft Security Bulletin MS03-007" Microsoft TechNet 17 March 2003. 19 March 2005. <http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx>

"Microsoft Security Bulletin MS03-039" Microsoft TechNet 10 September 2003. 19 March 2005. <http://www.microsoft.com/technet/security/bulletin/MS03-039.mspx>

Mills, David L. "DCnet Research Network", 2 Aug. 2004.15 March 2005. <http://www.eecis.udel.edu/~mills/database/brief/dcnet/dcnet.pdf>

"NetFlow Services Solution Guide" Cisco Systems 23 Oct 2004. 3 March 2005 <http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/products_implementation_design_guide09186a00800d6a11.html>

"Network Ports Used by Key Microsoft Server Products" Microsoft TechNet 19 March 2005. <http://www.microsoft.com/technet/security/smallbusiness/topics/serversecurity/ref_net_ports_ms_prod.mspx>

One, Aleph. "Smashing The Stack For Fun And Profit" PHRACK 18 March 2005. <http://www.phrack.org/show.php?p=49&a=14>

Lam, Jason. "GIAC Certified Intrusion Analyst Practical Assignment" Global Information Assurance Certification 14 Oct. 2001. 16 Feb. 2005. <http://www.giac.org/certified_professionals/practicals/gcia/0441.php>

LURHQ Threat Intelligence Group. "Milkit: An Innovator of Old Technology" LURHQ. 12 March 2005. <http://www.lurhq.com/sig-milkit.html>

Perry. 11 April 2003. 17 March 2005. <http://coders.meta.net.nz/~perry/irc.rules>

Sevcenco, Serghei. "Backdoor.Sdbot" Security Response 13 September 2004. 12 March 2005. <http://securityresponse.symantec.com/avcenter/venc/data/backdoor.sdbot.html>

Van Horenbeek, Maarten. "GIAC Certified Intrusion Analyst Practical Assignment" Global Information Assurance Certification 8 Jan 2003.14 March 2005. <http://www.daemon.be/~maarten/Maarten_Vanhorenbeeck_GCIA.pdf>