Global Information Assurance Certification Paper

SANS Network Security 2004
Las Vegas
September 29th – October 4th
GCIA – Track 3c

Practical Assignment – Version 4.1

Submitted by

Mohan K. Chirumamilla

January 29th 2005

Table of Contents

1.0 Executive Summary

Lack of security policies and controls coupled with high bandwidth, computing and storage power makes university networks hackers' a first choice for attacks. In order to evaluate the overall health of the University, a total of 15 log files - five alert logs, five scan logs and five Out-of-Spec (OOS) logs, from March 8th till March 12th were examined for this report. While not all 3975967 scan logs, 39388 alert logs and 2906 OOS logs were considered, three alerts that were deemed critical-enough were analyzed in full detail. In addition, 15 external hosts were closely examined to determine the level of risk associated with each of them. Refer to section 3 for details on how these hosts were selected. Furthermore, the top five most targeted ports are also discussed at the end of the report.

During the analysis, it has been observed that the network was being heavily scanned for both well-known services (which have known vulnerabilities associated) and for ports associated with some viruses/trojans. Unfortunately, there were incidents where some internal hosts responded to such scanning probes and were actively targeted. Details about these hosts are provided below; these hosts should be immediately taken offline and investigated for signs of compromise. Refer to Appendix B for details about the hosts discussed below.

- At least two internal hosts were participating in illegal file-sharing using one of two methods – Internet Relay Chat (IRC) or BitTorrent file-sharing technology. One of the hosts using IRC may also be participating in launching Distributed Denial of Service attacks.
- At least one host may have been compromised due to a vulnerability in a service that is used for synchronizing time between hosts.
- At least eight hosts should be investigated for a trojan called SubSeven, which gives remote attackers full control of the compromised machines.
- At least one host may have a remote administration tool called VNC installed.
- At least four hosts may have a remote administration tool called Dameware installed.
- The University Web servers were attacked.

A closer look at 15 external hosts that generated the maximum number of logs disclosed that not all of them were hostile in nature. Some hosts are listed under the top talkers list due to obsolete Intrusion Detection System (IDS) rules. Based on the analysis, each of the 15 external hosts is categorized into one of the high, medium, low categories. High-risk hosts should be blocked at the perimeter firewall and medium-risk hosts should be under close watch.

High risk	Medium risk	Low risk
24.31.31.188	62.111.194.65	68.54.84.49
63.251.52.75	213.157.171.109	66.225.198.20
	134.2.78.155	64.91.255.232
	202.155.56.180	67.114.19.186
	62.243.174.161	220.37.240.35
	218.197.122.51	63.211.17.228

	68.32.127.158	

Even though providing resources and supporting research activities is one of the primary objectives of educational institutes, lack of proper security controls could jeopardize the productivity of such conducive environments. Most of the hostile events that we have seen in this report could have been prevented by stricter policies and controls. At the very minimum, new firewall policies should be implemented to restrict in-bound traffic to unwanted services at the network perimeter. In addition, centralized patch management and anti-virus solutions should be employed to mitigate the risks associated with vulnerabilities and viruses. Furthermore, processes should be designed to maintain up-to-date inventories of all authorized services and servers hosting those services. These services and servers should be considered while designing the firewall policies and IDS rules. Lastly, as technology changes, the IDS rules should be regularly updated to reduce the number of false positives as they could be major decoys to real attacks otherwise. Many of the OOS logs were false positives and were the result of not updating the IDS rules.

1.1 Files Analyzed

Alert Files	OOS Files	Scan Files
alert.040308	oos_report_040308	Scans.040308
alert.040309	oos_report_040309	Scans.040309
alert.040310	oos_report_040310	Scans.040310
alert.040311	oos_report_040311	Scans.040311
alert.040312	oos_report_040312	Scans.040312

All files listed above were taken from http://isc.sans.org/logs.

Note: The dates in the OOS log files do not match the dates shown on the filenames. The files were supposed to have logs from March 8th-March 12th but were found to have logs from March 12th –March 16th. This was realized after I started my analysis; so, I decided to stick to these files for the rest of my analysis.

There were a total of 3,975,967 scan logs, 39,388 alert logs and 2,906 OOS logs altogether. One important note is that all port scan alerts (alerts starting with spp_portscan) were eliminated during the parsing of alert logs. Furthermore, the following scan and alert logs were not considered during my analysis due to formatting problems:

*130.85.190.92:4096 -> 218.245.94.61:135 SYN ******S** (from scan.040312)

03/09-07:01:13.356481 [**] [UMBC NIDS IRC Alert] K\:line'd user detected, possible trojan. [**] 81.174.249.138:6881 -> MY.NET.82.109:1361 (from alert040309)

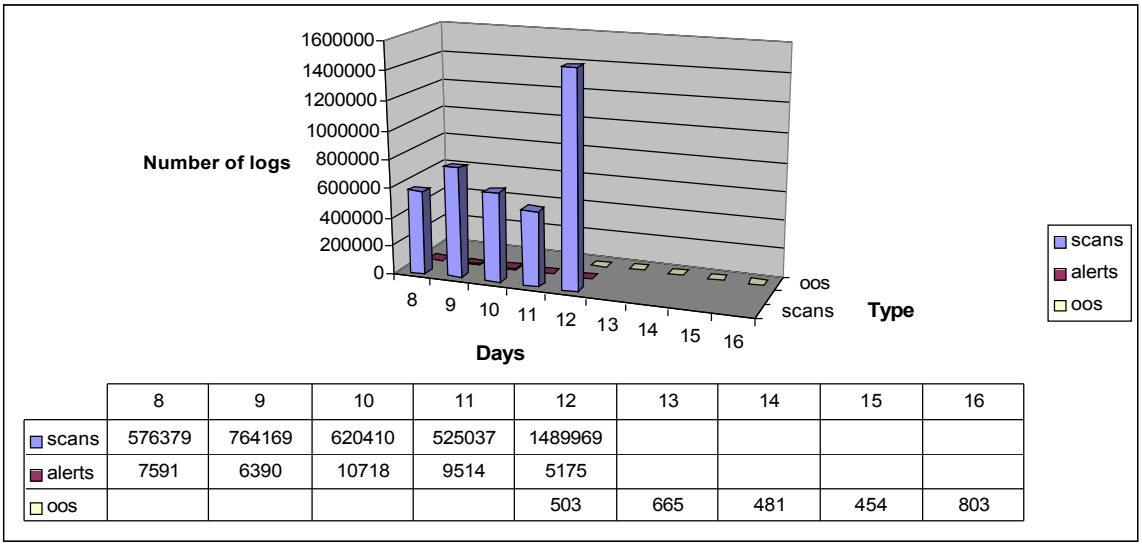
The following chart shows the distribution of all alert logs.

ALERT	FREQ
MY.NET.30.4 activity	11051
MY.NET.30.3 activity	10839
High port 65535 tcp - possible Red Worm - traffic	6494
SMB Name Wildcard	2742
connect to 515 from outside	2203
EXPLOIT x86 NOOP	1353
Null scan!	1048
NMAP TCP ping!	668
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	435

SUNRPC highport access!	344
Possible trojan server activity	271
IRC evil - running XDCC	244
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	240
High port 65535 udp - possible Red Worm - traffic	212
Incomplete Packet Fragments Discarded	166
External RPC call	163
connect to 515 from inside	120
RFB - Possible WinVNC - 010708-1	109
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	90
SMB C access	84
FTP DoS ftpd globbing	81
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	73
TCP SRC and DST outside network	63
TFTP - Internal UDP connection to external tftp server	49
FTP passwd attempt	42
EXPLOIT x86 setuid 0	39
[UMBC NIDS] External MiMail alert	29
EXPLOIT x86 setgid 0	18
Attempted Sun RPC high port access	17
EXPLOIT NTPDX buffer overflow	13
TCP SMTP Source Port traffic	12
DDOS shaft client to handler	9
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	9
HelpDesk MY.NET.70.49 to External FTP	8
External FTP to HelpDesk MY.NET.70.49	6
SYN-FIN scan!	6
External FTP to HelpDesk MY.NET.70.50	5
TFTP - External TCP connection to internal tftp server	5
TFTP - External UDP connection to internal tftp server	5
NETBIOS NT NULL session	4
NIMDA - Attempt to execute cmd from campus host	4
Probable NMAP fingerprint attempt	3
DDOS mstream client to handler	2
EXPLOIT x86 stealth noop	2
ICMP SRC and DST outside network	2
Tiny Fragments - Possible Hostile Activity	2
Back Orifice	1
TFTP - Internal TCP connection to external tftp server	1
Traffic from port 53 to port 123	1
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	1

Refer to Appendix A-8 for a detailed description of the format of all log types (OOS, scan and alert). The graph on the following page shows the distribution of logs spread across the days.

Analysis of all the logs indicates that the IDS sensor was placed on the same subnet as that of MY.NET.12.6. The reason for this is because there are about 53 OOS logs where the source IP and the destination IP were both



	8	9	10	11	12	13	14	15	16
scans	576379	764169	620410	525037	1489969				
alerts	7591	6390	10718	9514	5175				
oos					503	665	481	454	803

internal; the destination IPs had a prefix of either “MY.NET.12.”, “MY.NET.24.” or “MY.NET.29.” (the prefixes were not labeled as subnets as the subnet mask information is not known). Furthermore, the TTL in one of the OOS entries with MY.NET.12.6 as the source IP (oos_report_040308) was set to 255 (the maximum value, usually used by Solaris [46]); this shows that the packet has not yet traversed any router. Refer to Appendix-11 for the detailed OOS log. In addition, the facts that MY.NET.12.6 is listed as the University’s mail exchanger and that the University’s Web server (www.umbc.edu) has an IP address of MY.NET.12.11, indicate that the IDS sensor, MY.NET.12.6 and the Web server were placed on the same broadcast domain that was publicly accessible from the Internet (possibly in the DMZ). Furthermore, the Web server and the mail exchanger did not respond to any ping requests (even though they were alive) indicating the presence of a filtering device that is filtering out ICMP messages in front of the IDS sensor.

2.1 Detect 1

03/09-19:56:48.499526 [**] [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected. [**] 66.235.194.217:6667 -> MY.NET.42.3:3485

Note: *In the following section XDCC bot and XDCC client are used interchangeably. The main difference between a bot and a client is that, a bot is an automated program and a client needs human intervention.*

2.1.1 Reason this detect was selected

The above detect was taken from alert.040309. The file can be found at http://isc.sans.org/logs/alerts. According to the alert, there’s an incoming XDCC send request to MY.NET.42.3. This is indicative that MY.NET.42.3 may have been compromised and is being used to host warez files using an XDCC bot. Such bots can also be used for launching DDoS attacks. Therefore the main reason for selecting this alert is to determine if MY.NET.42.3 was hosting any such malicious software.

2.1.2 Detect was generated by

By examining the format of the logs (OOS, scan and alert), it is determined that the alert was generated by Snort. However, no information on

the version of Snort is available. I could not find any rule in Snort that could have triggered this alert. Nonetheless, http://coders.meta.net.nz/~perry/irc.rules (last accessed on 12/03/04) listed some rules regarding IRC traffic; the following rule was associated with the above alert message.

```
alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any \
(content: " |3a 01|XDCC "; \
msg: "Possible Incoming XDCC Send Request Detected.";\
classtype: misc-activity; )
```

The alert gets triggered by any in-bound TCP segment with a payload that contains "0x3a 0x01XDCC" and has a source port between 6660 and 7000. The IDS personnel at the University may have been using the above custom rule.

2.1.3 Probability the source address was spoofed

The source address was probably not spoofed. As per the above rule, the alert should trigger after the TCP three-way handshake was completed. The fact that a successful handshake occurred suggests that the source address was not spoofed.

Note: *Since there were no OOS logs coming from either the source or the destination IPs, it is reasonable enough to assume that there was no data sent in SYN, SYN+ACK and ACK packets and that the trigger was tripped after the TCP handshake was completed.*

2.1.4 Attack description

The alert was triggered because of the traffic between an XDCC bot and an IRC server. The bot can be configured to automatically connect to an IRC server and to join some pre-determined channels. After joining a channel, the bots can automatically advertise the list of files that they are serving and can also interactively communicate with other users (like receiving commands to launch a DDoS etc.). Users can select files to download using Client-to-Client-protocol (CTCP) and Direct Client to Client (DCC) commands. The bot and users then setup separate channels when sharing files. The connection setup between the bot and users can happen immediately or users' requests can be put in queues depending on the load and settings on the bot. Refer to [20 – 26] for details about the relation between IRC, DCC, CTCP and XDCC. There are numerous vectors for a host to get infected with this bot. A review of the scan logs identified that there were three sets of out-bound SYN connections going to ports between 6881 – 6889 from MY.NET.42.3. In each case the source ports were increasing by one. This traffic is representative of BitTorrent file-sharing [4 - 11]. Furthermore, 207.44.214.88 and 66.235.194.217 seem to be "SYN" scanning MY.NET.42.5 and MY.NET.42.4 for various ports. Refer to Appendix-10 for details about the scans. MY.NET.42.3 could have been infected by 1) user installation or 2) a virus/worm/trojan via BitTorrent file-sharing 3) social engineering or, 4) exploiting a weakness associated with a service.

2.1.5 Attack mechanism

The alert by itself does not speak of any attack per se. However, it does indicate the presence of an XDCC bot/client on MY.NET.42.3. As there were no raw packet traces available, other alert logs were examined to determine if this was a legit alert or a false positive. The following table shows other alerts that MY.NET.42.3 was involved in.

Alert	Frequency

EXPLOIT x86 setgid 0	1
IRC evil – running XDCC	60
SMB Name Wildcard	9
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	6
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	14
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	1

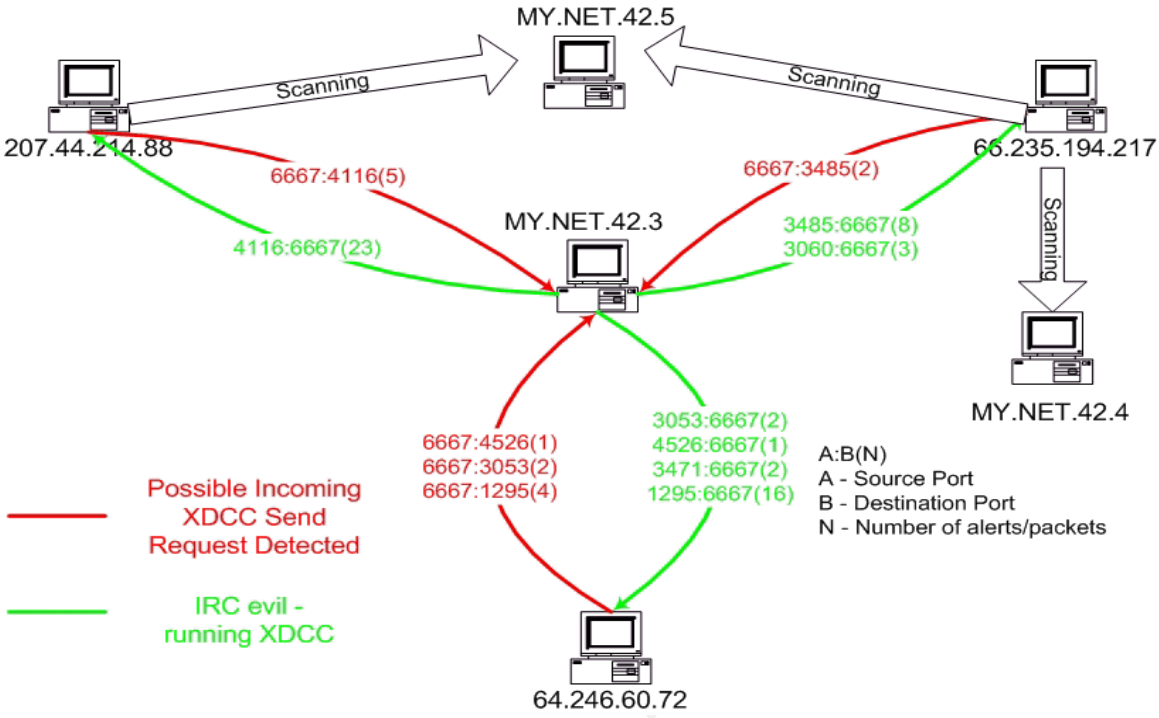
Since “IRC evil – running XDCC” [17] and “[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected” were the two most frequently triggered alerts and seem to be related, I plotted a link graph and found that MY.NET.42.3 was interactively communicating (the source ports and the destination ports were swapped with the change in direction) with external hosts.

There’s no specific pattern on the timestamps associated with the alerts except that the bot was constantly talking back to some IRC servers.

Here’s a snippet of alerts associated with MY.NET.42.3 and 66.235.194.217

03/09-17:02:10.146848 [**] IRC evil - running XDCC [**] MY.NET.42.3:3485 -> 66.235.194.217:6667
03/09-16:51:24.146967 [**] IRC evil - running XDCC [**] MY.NET.42.3:3485 -> 66.235.194.217:6667
03/09-17:08:20.131219 [**] IRC evil - running XDCC [**] MY.NET.42.3:3485 -> 66.235.194.217:6667
...
03/09-18:48:09.175031 [**] IRC evil - running XDCC [**] MY.NET.42.3:3485 -> 66.235.194.217:6667
03/09-19:45:57.193705 [**] IRC evil - running XDCC [**] MY.NET.42.3:3485 -> 66.235.194.217:6667
…
03/09-19:56:48.499526 [**] [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected. [**] 66.235.194.217:6667 -> MY.NET.42.3:3485
03/09-20:22:42.395031 [**] [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected. [**] 66.235.194.217:6667 -> MY.NET.42.3:348
...

Also, telnet’ing to port 6667 (on 12/09/04) proved that two of the three servers in the graph were IRC servers. Refer to Appendix A-1 for details. *Note: The third IP was not responding to any pings as of 12/09/04.*



Link Graph

XDCC bots are programs that mimic real users. They can be used to serve files to users in an IRC channel or act as intermediate agents in launching DDoS attacks. Clients can download files using DCC and CTCP [23]. If client A wants to request a file from client B: Client A opens a new TCP socket that is bound to a port and sends a CTCP command using DCC. The main elements of the commands are the type, the host address of A and the port on which A is expecting to receive the file; other type-specific arguments are also included when appropriate. The type could either be DCC CHAT or DCC SEND. Refer to [24] for detailed specifications about the DCC protocol. XDCC is an enhanced automated version of DCC where one side of the communication is a computer program taking the role of a normal user [25]. XDCC uses DCC as the underlying communication protocol. When used as intermediate agents for launching a DDoS attacks, the agents (XDCC bots) receive commands from an attacker via IRC channels. Hosts that are infected with such bots will be under the control of an attacker.

2.1.6 Correlations

XDCC bots are quite prevalent in university environments because of their open nature. Lack of security policies and controls coupled with high-bandwidth and excessive computing power makes universities attractive targets. No reports or references on the external hosts were found. Some students in the past did discuss some issues related to IRC and bots. Ryan Barrert [20] has briefly discussed on the "XDCC client detected attempting to IRC" alert – something similar but not same. Marcus Wu [19] has analyzed an IRC related alert and discussed about XDCC bots briefly. An article by TonikGin [27] discussed up to a certain detail on how machines at universities could be compromised.

2.1.7 Evidence of active targeting

All evidence presented in the above sections indicates that MY.NET.42.3 has an XDCC bot and is involved in two-way conversations with some IRC servers. So, the traffic is definitely active and targeted at MY.NET.42.3.

2.1.8 Severity

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

Criticality: 3

Apparently, MY.NET.42.3 is a normal workstation because of its involvement in BitTorrent file-sharing activities. Normally, a machine that's dedicated to a special service would not have a file-sharing software unless installed by a system administrator (One reasonable assumption that I am making is that BitTorrent file-sharing software was installed locally by someone because of its dependence on human intervention to operate). It is possible that a remote attacker could be controlling the application but, a line has to be drawn some where.

Lethality: 5

The XDCC bot may have been installed via a system compromise, a virus/worm/trojan or installed by a user. Therefore, we either have a compromised host or a host that has opened a new attack vector into the network.

System countermeasures: 2

There were no tangible preventive (restricted admin access etc) or detective controls (anti-virus software etc) on the machine. Since there is no additional any information, I assign a rating of 2.

Network countermeasures: 3

Since in-bound IRC connections were allowed into the network, it appears there were no restrictions at the perimeter firewall; however, detective controls to detect IRC traffic were present.

Severity = (3+5)-(2+3) = 3

2.1.9 Defensive recommendations

My main recommendation would be to have the firewall block all outbound SYN-ACK packets destined to port 6667 and other common IRC ports. I would also recommend the use of a good campus-wide anti-virus solution. Furthermore, wherever possible, controls should be in place to restrict end-users from logging into the machines as administrators. This measure could potentially prevent attack vectors that use social engineering.

2.2 Detect 2

03/08-11:41:12.554397 [**] EXPLOIT NTPDX buffer overflow [**]
66.250.188.23:180 -> MY.NET.66.29:123

2.2.1 Reason this detect was selected

The above detect was taken from alert.040308. The file can be found at http://isc.sans.org/logs/alerts. The alert warns that there is a buffer overflow attempt against one of the internal machines and thus would like to investigate if there is a compromised host on the network.

2.2.2 Detect was generated by

The detect was generated by an unknown version of Snort. As of 12/11/04, the closest rule from the latest Snort rule-set that could have generated the above alert is

```
alert udp $EXTERNAL_NET any -> $HOME_NET 123 (msg:"EXPLOIT ntpdx
overflow attempt"; dsize:>128; reference:arachnids,492;
reference:bugtraq,2540; reference:cve,2001-0414; classtype:attempted-
admin; sid:312; rev:6;)
```

According to the rule, any inbound UDP packet destined to port 123 and greater than 128 bytes will trip this alert. However, there could be some false positives triggered by this rule. It has been explained in [30] that the only way to detect this attack is to watch out for oversized stimulus from an attacker.

2.2.3 Probability the source address was spoofed

There's a 50% probability that the source address was spoofed. Since NTP is based on UDP, it is very easy to spoof the source IP. There were a total of nine alerts from the same external host but from different source ports.

03/08-11:41:12.554397 [**] EXPLOIT NTPDX buffer overflow [**] 66.250.188.23:180 -> MY.NET.66.29:123
03/08-12:46:17.600602 [**] EXPLOIT NTPDX buffer overflow [**] 66.250.188.23:472 -> MY.NET.66.29:123
03/09-12:01:44.983930 [**] EXPLOIT NTPDX buffer overflow [**] 66.250.188.23:5 -> MY.NET.66.29:123
03/09-11:58:07.890713 [**] EXPLOIT NTPDX buffer overflow [**] 66.250.188.23:123 -> MY.NET.66.29:123
03/09-11:58:13.094392 [**] EXPLOIT NTPDX buffer overflow [**] 66.250.188.23:123 -> MY.NET.66.29:123
03/10-15:18:52.930067 [**] EXPLOIT NTPDX buffer overflow [**] 66.250.188.23:383 -> MY.NET.66.29:123
03/12-13:12:39.288652 [**] EXPLOIT NTPDX buffer overflow [**] 66.250.188.23:467 -> MY.NET.66.29:123
03/12-13:12:42.705309 [**] EXPLOIT NTPDX buffer overflow [**] 66.250.188.23:467 -> MY.NET.69.211:123
03/12-14:36:07.347091 [**] High port 65535 udp - possible Red Worm - traffic [**] 66.250.188.23:1279 -> MY.NET.66.29:65535

The sheer number of instances where the source IP was same is a significant observation that should be considered. Apparently, the attacker was persistently trying to communicate with MY.NET.66.29 over port 123. Also, the source ports were privileged ports which indicate that the packets could either have been crafted or the attacker has root privileges on 66.250.188.23.

2.2.4 Attack description

Network Time Protocol (NTP) is a UDP-based protocol to synchronize time between an NTP server and an NTP client or between peers participating in a time synchronization process. As of 12/11/04, the latest version of the NTP protocol is version 3 [35]. Previous versions of the NTP protocol could be found in [33, 34, 36]. As mentioned in [34], depending on the mode of operation [35], it is possible that the source port, the destination port or both could be using port 123. In the scans shown above, there were instances where the source port was an ephemeral port.

The alert may have been triggered by a stimulus – attacker trying to exploit a weakness in the implementation of NTP. Detailed analysis of this vulnerability could be found in [30, 41]. The bufferoverflow weakness exists mostly in Cisco IOS and in some flavors of Unix. Since Cisco IOS is proprietary, we will briefly discuss the implementation of the NTPD daemon in the vulnerable flavors of Unix. The weakness exists within the ctl_getitem() function and when exploited, the attacker can remotely execute arbitrary code with the privileges of the account that NTPD is running as (root, by default). Refer to [41] for details about the functions. CVE-2001-0414 and VU#970472 are the

corresponding advisories.

2.2.5 Attack mechanism

In order to successfully exploit this vulnerability, the attacker has to send two UDP datagrams. The first packet contains the shell code; this code is then written to an intermediate static location. When the attacker sends another UDP datagram, a NULL query with no data, the contents written to the static location would be copied over to a variable on the memory stack overwriting the return address while preparing a response packet. When the function returns, the code at the location overwritten by the attacker will be executed. Refer to the paper by Miika Turkia [41] for a detailed explanation on this vulnerability.

Examining all nine alerts shown in section 2.2.3, it appears that the attacker did not succeed in obtaining the root privilege, at least for the first seven attempts (the attacker wouldn't be going for subsequent attempts if any of the prior attempts were successful). The alerts were triggered approximately around noon ±2 hours on almost all days except on 11th. On the 12th, there was a ninth alert triggered by a UDP datagram coming from 66.250.188.23 destined to MY.NET.66.29 over port 65535. While it is technically possible to have a packet destined to 65535, it is very unusual; this activity is tied to the adore worm. If the machine were infected with the adore worm, then according to the analysis provided in [39], one of the BIND, wu-ftpd, rpc.statd or lpd services should also have been targeted. However, there is no evidence in the logs (scan, alert or OOS) that one of these four services was targeted. Furthermore, another host (MY.NET.109.86) that was being attacked over port 123 was also sent a similar UDP datagram destined to port 65535. [42, 43] also reported similar incidents where the NTPDX buffer overflow alert was accompanied with the "High port 65535 udp - possible Red Worm – traffic" alert. This might be an indication that there's a variant of the ntpdx buffer overflow attack that is using a similar technique as that of the adore worm in creating a backdoor. In the case of the adore worm, the victim host opens a backdoor on port 65535 only after receiving a ping packet of a pre-determined size [39]. One interesting observation is that all hosts that were attacked over port 123 had SYN connections coming from almost the same external IPs destined to almost the same destination ports. Refer to the below table for the list of unique destination ports that were scanned for and to Appendix A-2 for full scans. Initially, I suspected a relation between these scans and the NTPDX buffer overflow alerts.

Destination IP	Ports Scanned	Destination IP	Ports scanned
MY.NET.66.29	1257 17300 20168 21 3000 4000 4489 4898 4899 6129 7755 80 8080	MY.NET.109.86	17300 20168 21 3000 4489 4899 6129 7100 7755 80 8080 8150
MY.NET.53.204	1257 17300 20168 21 2745 3000 4000 4489 4898 4899 5900 6129 7100 7755 80	MY.NET.69.211	1524 17300 20168 21 2745 4489 4898 4899 5900 6129 7100 7755 80 8150

However, further querying on the destination ports listed above, indicated no relation between them. There was a separate distributed scan targeting thousands of internal hosts targeting the ports listed above. Most of these ports are associated with known trojan horses. Original raw packet traces should be examined to determine if the attacker was successful in compromising the host.

2.2.6 Correlations

The buffer overflow in NTP was first published by Przemyslaw Frasunek in April 2001 [37]. The corresponding CERT and CVE advisories could be found using the reference numbers VU#970472 and CAN-2001-0414. Glenn Larratt [42] and Philip Ljungberg [43] have observed similar NTPDX buffer overflow alerts followed by red worm alerts. Miika Turkia [41] and Philipp Stadler [30] have also performed detailed analyses in their papers explaining the technical details behind this vulnerability.

2.2.7 Evidence of active targeting

From the number of alerts spread across four days, it appears that the attacker was desperately sending oversized UDP packets to MY.NET.66.29 destined to port 123. This is a sign of active targeting.

2.2.8 Severity

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

Criticality: 3

There was no sufficient data in any of the three log files (scans, OOS or alerts) indicating that MY.NET.66.29 was running important services and hence a rating of 3.

Lethality: 3

A remote attacker could potentially gain administrative privileges. However, technical expertise is required to successfully launch the attack.

Note: *The size of the shell code should be less than 70 bytes. Otherwise, the memory stack on the target machine would be corrupted rendering the NTP service inactive* [37].

System countermeasures: 1

No sufficient data (raw packet dumps along with payload) is available to determine the patch level on the machine. However, I would suggest having a complete investigation on this host and therefore, would assign a rating of 1.

Network countermeasures: 3

Apparently, traffic destined to port 123 was not blocked at the perimeter firewall. However, the presence of the alert indicates the presence of a Snort rule to detect attacks against this service.

Severity = (3+3) – (1+3) = 2.

2.2.9 Defensive recommendations

Rules should be added to completely disable all inbound and outbound NTP traffic if possible. If there's a justification for NTP traffic to flow in and out, rules should be designed around specific IP addresses (using TCP wrappers), and the hosts should be hardened and installed with up-to-date patches. If NTP service is running but not needed, the service should be immediately disabled.

2.3 Detect 3

03/09-17:28:42.942900 [**] SUNRPC highport access! [**] 142.165.212.10:4316 -> MY.NET.70.50:32771

2.3.1 Reason this detect was selected

Sun Remote Procedure Call (RPC) services were known to have multiple vulnerabilities [47] associated with them. When alerts related to RPC services are triggered, it would be at the best interest of an intrusion analyst to review them. It was mentioned in [47] that "The broadly successful attack on U.S. military systems during the Solar Sunrise incident also exploited an RPC flaw found on hundreds of Department of Defense systems". There were a total of 344 "SUN RPC highport access" alerts in the alert files that I selected. Out of these 344 alerts, approximately 282 alerts had one of the ports associated with a known service and hence assumed that they were mostly false positives. I agree that a true analyst would make no assumptions; however, since all alerts cannot be discussed, a random alert from alerts that are most likely not false positives is selected.

2.3.2 Detect was generated by

By examining the format of the alert, it can be said that the alert was generated by an unknown version of Snort. As of 12/20/04, I was unable to find any rule that could have triggered this alert. However, the following rule was found on [44] – probably a sample configuration file from snort 1.2.1

```
alert tcp any any -> $HOME_NET 32771 (msg: "SUNRPC highport
access!";)
```

The University may have been using the above rule. According to this rule, any TCP segment destined to port 32771 would trigger this alert. Even though port 32771 is associated with sun RPC services, potential exists for false positives in a heterogeneous environment.

2.3.3 Probability the source address was spoofed

The source address was not likely spoofed. Apparently, 142.165.212.10 was involved in a reconnaissance mission. Detailed explanation to show that this alert was part of a scan is provided in the attack mechanism section.

2.3.4 Attack description

Port 32771 is generally associated with SUN RPC services which are known to have several vulnerabilities. CA-99-08 [48], CA-99-05 [49] and CA-98-11 [50] are some of the CERT advisories pertaining to RPC services. CVE-1999-0003, CVE-1999-0008, CVE-1999-0212, CVE-1999-0228, CVE-1999-0320, CVE-1999-0353, CVE-1999-0687, CVE-1999-0696, CVE-1999-0900, CVE-1999-0969 and CVE-1999-0974 are some of the CVE advisories regarding RPC services. Apparently, the alert under consideration is part of a scanning activity rather than that of any attack. Probably, the first scan shown below may have triggered this alert.

*Mar 9 17:28:42 142.165.212.10:4316 -> MY.NET.70.50:32771 SYN ******S**
Mar 9 17:28:44 142.165.212.10:4316 -> MY.NET.70.50:32771 SYN ******S**

Examining the timestamps on the first scan log and the alert log in section 2.3, it is evident that both were logged on March 9th at 17.28.42 (Hour: Minute: Second). In addition, the source ports were same in both the scan logs. This

might be an indication that they were retries. The presence of sequence numbers would have helped verify this. Depending on the operating systems, the duration between retry attempts will vary. Raw packet traces should be examined to see if there were any SYN-ACK packets sent back to the attacking host.

Furthermore, running an nslookup query on 142.165.212.10 indicated that this IP was mapped to www.infotaxi.ca. A whois query shows that the IP is owned by SaskTel. Refer to section 4.1 for details. As Ryan Barrett [20] pointed out, the contents of the Web site (last accessed on 12/21/04) seem to be benign, which lead me to believe that 142.165.212.10 was a victim itself and was used to disguise the attackers identity. By requesting a non-existing page from the Web site and sniffing the traffic, it is determined that 142.165.212.10 is probably a Wind9x/NT box running Apache 1.3.19 on port 80. Refer to section 4.1 for details.

2.3.5 Attack mechanism

In this section we would be examining various logs that help us determine that this detect belongs to a scanning activity rather than an attack. There were a total of 72 alert logs all targeting port 32771. The alerts were triggered at four different times. The first alert was triggered at 16:59:09 and the second set was triggered between 17:28:42 and 17:28:47; 48 hosts in the MY.NET.70 subnet were scanned during this time. The third set of alerts was triggered between 17:34:46 and 17:34:51; 13 hosts all in the MY.NET.70 subnet were scanned. The final set of alerts was triggered almost an hour later between 18:32:49 and 18:33:42; 10 hosts all in the MY.NET.190 subnet were scanned during this time. This pattern suggests that the program the attacker used was programmed to scan subnet by subnet, as opposed to scanning completely random hosts in the given address space. Following is a snippet of those 72 alerts.

03/09-16:59:09.752582 [**] SUNRPC highport access! [**] 142.165.212.10:4551 -> MY.NET.5.13:32771
03/09-17:34:46.978450 [**] Attempted Sun RPC high port access [**] 142.165.212.10:3962 -> MY.NET.70.1:32771
03/09-17:34:47.048223 [**] Attempted Sun RPC high port access [**] 142.165.212.10:3963 -> MY.NET.70.5:32771
…
03/09-18:33:42.204160 [**] Attempted Sun RPC high port access [**] 142.165.212.10:1409 -> MY.NET.190.203:32771
03/09-18:33:42.371335 [**] Attempted Sun RPC high port access [**] 142.165.212.10:1410 -> MY.NET.190.0:32771

Note: Even though there are two different alert messages in the snippet, both were logged by the same rule with different alert messages. The following rule can be found in [45].

```
alert tcp any any -> 192.168.1.0/24 32771 (msg: "Attempted Sun RPC high port access";)
alert udp any any -> 192.168.1.0/24 32771 (msg: "Attempted Sun RPC high port access";)
```

There were a total of 7,822 scan logs from 142.165.212.10 between 16:59:03 and 18:33:53 on March 9th. Out of these, 161 scans (approximately 2%) were targeting UDP ports; the remaining 7,661 scans were SYN scans. One more interesting observation is that all the UDP scans started after the SYN scans. The timing of the scans indicates that the attacker used an automated script. Another indication that the attacker used a script is that scans were sent to MY.NET.190.0, MY.NET.70.0 and MY.NET.71.0. Unless the attacker knew the subnet mask values, it is an odd coincidence (Some hosts that have a TCP/IP stack based on the Unix BSD operating systems consider addresses like MY.NET.190.0 as broadcast addresses. However, we know that the protocol involved here is TCP, and TCP is a unicast protocol. Hence, the possibility that

the attacker is aiming at broadcast addresses is ruled out).
Following is a snippet of the scans

*Mar 9 16:59:03 142.165.212.10:4329 -> MY.NET.5.13:66 SYN ******S**
Mar 9 16:59:03 142.165.212.10:4330 -> MY.NET.5.13:67 SYN ******S**
…
Mar 9 16:59:03 142.165.212.10:4323 -> MY.NET.5.13:23 SYN ******S**
Mar 9 16:59:04 142.165.212.10:4332 -> MY.NET.5.13:70 SYN ******S**
...
Mar 9 18:33:51 142.165.212.10:1579 -> MY.NET.190.102:32788 UDP
Mar 9 18:33:52 142.165.212.10:1584 -> MY.NET.190.92:32789 UDP
…
Mar 9 18:33:53 142.165.212.10:1608 -> MY.NET.190.203:43981 UDP
Mar 9 18:33:53 142.165.212.10:1607 -> MY.NET.190.202:43981 UDP*

Of all 7,882 scans, 47 scans were aimed at port 32771. To summarize, a total of 84 unique hosts were scanned for 207 unique ports. Although not every host was scanned for all 207 ports, there's a lot of overlap between hosts. Refer to Appendix A-3 to see how many ports were scanned on each host.

2.3.6 Evidence of active targeting

No, there's no evidence of active targeting. The alert was triggered by a stimulus to see the state (closed or opened) of port 32771 on MY.NET.70.50.

2.3.7 Correlations

I queried to see if there were any alerts where the destination IP is 142.165.212.10 and found the following alerts. Interestingly enough, I couldn't find any corresponding scans (if they were any) that could have triggered these alerts:

*03/09-17:27:20.334881 [**] RFB - Possible WinVNC - 010708-1 [**] MY.NET.70.156:5900 -> 142.165.212.10:4096
03/09-18:32:49.229147 [**] Possible trojan server activity [**] MY.NET.190.202:27374 -> 142.165.212.10:4220
03/09-18:32:49.233474 [**] Possible trojan server activity [**] MY.NET.190.102:27374 -> 142.165.212.10:4219
03/09-18:32:49.236385 [**] Possible trojan server activity [**] MY.NET.190.203:27374 -> 142.165.212.10:4221
03/09-18:32:49.747401 [**] Possible trojan server activity [**] MY.NET.190.102:27374 -> 142.165.212.10:4219*

The reason there weren't any stimuli for these responses is because the Snort sensor may have dropped the SYN scan probes destined to these internal hosts, or depending on the topology of the network, the SYN packets (if there were any) may have taken a different path. The following alerts had corresponding scans which means that the attacker got some responses for his/her stimuli:

*03/09-18:32:48.737584 [**] Possible trojan server activity [**] MY.NET.190.1:27374 -> 142.165.212.10:4214
03/09-18:32:48.742644 [**] Possible trojan server activity [**] MY.NET.190.93:27374 -> 142.165.212.10:4216
03/09-18:32:48.744579 [**] Possible trojan server activity [**] MY.NET.190.97:27374 -> 142.165.212.10:4218
03/09-18:32:49.242534 [**] Possible trojan server activity [**] MY.NET.190.92:27374 -> 142.165.212.10:4215*

There is only one alert associated with each internal IP. Putting the bits and pieces together, I guess these are responses to stimuli (SYN scan probes to ports 5900 and 27374). Port 27374 is associated with a very popular trojan called SubSeven. This trojan gives attackers full access to remote machines. Port 5900 is associated with VNC, a tool for remotely administering Windows/Unix/Linux machines. These hosts should be investigated for any signs of compromise.

These alert logs indicate that 142.165.212.10 was indeed scanning the internal network. Ryan Barrett [20] has discussed about the attacking host, 142.165.212.10, and indicated that it was involved in some large-scale information gathering activities. In addition, James Haywood [52] has discussed about the "SUNRPC highport access" alert. Various CERT/CVE advisories

regarding RPC vulnerabilities have been released. CA-99-08 [48], CA-99-05 [49] and CA-98-11 [50] are some of the CERT advisories pertaining to RPC services. CVE-1999-0003, CVE-1999-0008, CVE-1999-0212, CVE-1999-0228, CVE-1999-0320, CVE-1999-0353, CVE-1999-0687, CVE-1999-0696, CVE-1999-0900, CVE-1999-0969 and CVE-1999-0974 are some of the CVE advisories regarding RPC services.

2.3.8 Severity

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

Criticality: 2

The attacker was scanning multiple internal hosts and not just one.

Lethality: 2

The alerts were triggered because of scanning activity.

System countermeasures: 1

Out of 84 hosts that were scanned, at least eight hosts responded. Out of the nine responses, seven indicated that port 27374 was open and one response indicated that port 5900 was open. Even though only eight hosts responded, we do not have enough information to determine that there were proper countermeasures and hence would assign a paranoid rating of 2.

Network countermeasures: 2

Scans to, at least, sensitive ports should have been blocked at the perimeter firewall. Apparently, they were not. However, IDS sensors were tuned to detect these scans.

Severity = (2+2) – (1+2) = 1

2.3.9 Defensive recommendations

The seven hosts that responded to SYN probes to port 27374 should be investigated for the SubSeven trojan. One host that responded for port 5900 (associated with VNC – a remote administration tool) should also be investigated for possible infections. Access to ports that are not required to be contacted from the Internet should be blocked at the perimeter firewall. To prevent attackers from gaining information about the network, ICMP messages should be appropriately controlled. Blocking all ICMP messages may break the functionality of some systems that the university is using. So, extreme care should be exercised when tweaking firewall rules relating to ICMP messages. I would also suggest updating RPC-related IDS rules number of false positives.

3. Network Statistics

In the following sections we will be exploring details about hosts that generated the most number of scan, OOS and alert logs. For our analysis, we will be considering the top five external source hosts that generated the maximum number of logs from each category.

3.1 Top five hosts – OOS logs

In this section we would be discussing the top five external hosts (as source hosts) that generated the maximum number of OOS logs. *Note: The dates on the OOS log files do not match the time stamps on the logs within those files. The files oos_report_040308.gz to oos_report_040312.gz contained logs from March 12th to March 16th.*

Source IP	Number of logs
68.54.84.49	1107
66.225.198.20	107
62.111.194.65	106
64.91.255.232	102
67.114.19.186	79

3.1.1 68.54.84.49

An Nslookup query resolved the IP address to 'pcp0011109240pcs.elkrdg01.md.comcast.net'. The FQDN of the host indicates that the source of the logs is from a host with a broadband connection from Comcast. There were a total of 1,386 scan logs generated by this host across the five days between March 8th and March 12th; no alerts were logged from this host.

I guess all the OOS and scan logs coming from 68.54.84.49 are false positives. All packets coming from 68.54.84.49 were destined to only one IP\port pair – MY.NET.6.7:110. The port 110 is usually associated with the Post Office Protocol (POP3), a protocol used to retrieve e-mails from a mail server to the client's local drive. The timestamps on the OOS logs also indicate that new connection requests were made approximately every one minute, a behavior very similar to that of email client software like Outlook. Furthermore, the source ports were increasing in a linear and predictable manner. The fact that SYN connections, with random sequence numbers, were approximately one minute apart and that the source ports were linearly increasing indicate that this is probably not a scanning activity but rather a normal traffic from a POP3 client. All the OOS logs in this case have the TCP reserved bits and the SYN bit set and IP ID set to 0. Having the IPID set to 0 when the DF bit is set was a common issue in some older operating systems. Therefore, the presence of the reserved bits could be the reason why the packets were logged as OOS packets. There are no other abnormalities in the logs. According to the revised TCP/IP protocol [55], ECN-aware machines set both ECN and CWR bits (the two high-order bits in the 13th byte of the TCP header) during the first phase of the 3-way TCP handshake. Considering the facts discussed above, my guess is all OOS and scan logs coming from 68.54.84.49 and going to MY.NET.6.7 on port 110 are false positives. Michael Bernstein [53] has also examined these logs and derived a similar conclusion. The University IDS rules should be updated according to the new TCP/IP standard [55] where the two reserved bits are not reserved any more.

3.1.2 66.225.198.20

There were a total of 107 OOS SYN packets all going from 66.225.198.20 to MY.NET.12.6:25. The source ports were randomly distributed between ports 33276 and 60174. The two-high order bits in the 13th byte of the TCP header were set on all SYN packets sent by this host, and this might be the reason why these packets were found to be abnormal. A reverse DNS lookup did not resolve the IP address to any hostname; however, it was found that the IP

address was registered to Sever Central Network, a Web hosting company. In addition, MY.NET.12.6 was found to be the University's mail exchanger (by querying the University's name server). The Company was offering Web-based IMAP client services to its customers (as of 01/12/05). IMAP, like POP3, is a protocol used to let users read e-mails from mail servers. Someone at the university might be using this service as their mail client. I think the SYN connections were made when that user was sending emails to people at the University using the services offered by Server Central Network. Since MY.NET.12.6 was listed as the mail exchanger, any e-mail sent to the University will be going via MY.NET.12.6. As explained in section 3.1.1, the packets were probably logged because of the presence of the reserved bits. Since traffic to port 25 on MY.NET.12.6 cannot be blocked at the perimeter, the host should be hardened and should be installed with the latest patches. The IDS rules should be updated to address the fact that the two high-order bits in the 13th byte of the TCP header are not reserved any more. This will reduce false positives considerably. It is also recommended to disable Open relay on the mail exchanger to prevent spammers from using it to relay spam.

3.1.3 62.111.194.65

Again, as mentioned in the above two sections, the reason that the packets could have been logged is the presence of the reserved bits. All packets were targeted to port 6883 on MY.NET.69.226. Port 6883 is associated with BitTorrent file sharing protocol. The external IP was resolved to host-ip65-194.crowley.pl and was also found to be participating in other file sharing programs [56, 57]. Bobby Noell [58] has also taken notice of this host in his analysis expressing similar thoughts. An investigation should be conducted on MY.NET.69.226 to see if it's participating in any BitTorrent file-sharing. Traffic to ports that are not needed to be communicated from the Internet should be blocked (like 6883 in this case). The IDS rules should be updated by taking into consideration that the two high-order reserved bits in the TCP header are not reserved any more.

3.1.4 64.91.255.232

The reason we are seeing these 102 logs (64.91.255.232 as the source IP) in OOS category is because of the presence of the reserved bits. The traffic resembles that of FTP traffic in passive mode [59]. The following logs show two sets (each color represents of set) of FTP sessions.

*03/13-03:45:33.521494 64.91.255.232:**34743** -> MY.NET.24.47:**21***
*03/13-03:45:33.992122 64.91.255.232:**34744** -> MY.NET.24.47:**2573***
*03/13-03:45:38.345487 64.91.255.232:**34747** -> MY.NET.24.47:**21***
*03/13-03:45:38.802960 64.91.255.232:**34748** -> MY.NET.24.47:**2575***
...

Every set has two connections one going to port 21, the FTP command port, and the other going to an ephemeral port. Furthermore, the source port in the second connection in each set was one higher than that of the first connection in that respective set. The destination port on MY.NET.24.47 to which the external host was connecting back was also increasing by two in each set with few exceptions. Refer to Appendix A-4 for sample OOS logs.

The source IP address did not get resolved to any specific FQDN but was

registered to Liquid Web (as per the AIRN database on 01/03/05). In addition, there were a total of 42 'FTP password attempt' alerts logged all targeting MY.NET.24.47; and 21 FTP related scan logs from 64.91.255.153. These logs indicate that there's probably an FTP server running on MY.NET.24.47. Except for the question of the FTP server being legitimate or not, the OOS logs are not malicious in nature. Investigation should be conducted to determine if the FTP server on MY.NET.24.47 is an authorized server. The IDS rules should also be updated to address the fact that the two high-order bits in the13th byte of the TCP header are not reserved any more.

3.1.5 67.114.19.186

As in the previous four cases, the 79 OOS logs associated with 67.114.19.186 were logged because of the presence of the TCP reserved bits. Connections were initiating from 67.114.19.186 to MY.NET.24.44:80 almost 30 minutes past every hour every day. This activity is indicative of an automated script being employed. The IP address was resolved to "adsl-67-114-19-186.dsl.pltn13.pacbell.net" (using nslookup on 01/04/05) and MY.NET.24.44 was found to be hosting some Web services for the University. There were a total of 77 scan logs between 67.114.19.186 and MY.NET.24.44; all seemed to be normal HTTP traffic. The FQDN of the external host indicates that the traffic is coming from a DSL connection. Coen Bakkers [67] has analyzed the traffic from 67.114.19.186 and expressed similar views about these logs. The IDS rules should be updated to address the fact that the two high-ordered bits in the 13th byte of the TCP header are not reserved any more.

3.2 Top five hosts – scan logs

sourceip	Number of logs
213.157.171.109	18937
134.2.78.155	12062
202.155.56.180	11631
62.243.174.161	11558
218.197.122.51	10741

In this section we will be discussing about the top five external hosts that generated the maximum number of scan logs.

3.2.1 213.157.171.109

213.157.171.109 started "SYN scanning" the internal hosts for port 80 on March 9th at 06:53:39 (HH:MM:SS). The scan ended the same day at 07:31:04, lasting approximately 37 minutes. A total of 11,806 unique internal hosts have been scanned for port 80. The short duration of time indicates that the attacker used a program to scan the hosts. The source port was different for each SYN connection, and did not have a strict pattern. However, the (source) ports were sometimes incremented in small deltas. This might be an indication that the source host was indulging in other network activities. The IP address was resolved to a host (213-157-171-109.brasov.rdsnet.ro) in Romania. A Google cache page [61] has listed this external IP address as a malicious host.

3.2.2 134.2.78.155

The above external IP address was assigned to the Eberhard Karls University of Tübingen in Germany. An nslookup query (on 01/05/05) resolved the IP address to "pt20001.tphys.physik.uni-tuebingen.de". The external host was performing a SYN scan against port 21. The source port was increasing by one with frequent exceptions and the same destination host was scanned

multiple times in some instances. The scan started on March 9th at 05:08:54 and ended at 05:17:14, lasting approximately eight minutes. A total of 9,689 unique internal hosts were scanned. Considering the time when the scans occurred and the number of hosts that were scanned it is determined that an automated script/program was used. No references on this external host were found.

3.2.3 202.155.56.180

This external host was found to be performing a SYN scan against the University network for port 6129, a port usually associated with Dameware, a remote administration tool that is commonly used to control compromised hosts in university environments [27]. The scan started on March 12th at 22:57:18 and ended at 23:30:14 the same day, lasting approximately 32 minutes. A total of 9,496 unique hosts were scanned. As in the previous scenario, the source port was found to be increasing by one intermittently. This might be representative of the network activity on the attacking host. This IP address belonged to Indosat Internet Service Provider, in Indonesia. The attacker might be looking for already-compromised systems with Dameware installed [27], or for some vulnerable versions of Dameware itself; versions of Dameware do have a bufferoverflow vulnerability, CAN 2003-1030. Ken Connelly has reported the same scanning activity on April 8th. The scan logs can be found in the 2004-April.txt.gz file located at [63]. This IP address was also found to be blacklisted on [64]. Traffic logs should be investigated to see if any hosts responded. A response (to the SYN probe) probably means that there's a compromised host. In addition, the perimeter firewall should be configured to block traffic to unnecessary ports like this one.

3.2.4 62.243.174.161

The scan originating from this external IP address was a SYN scan targeting port 6129 as in the previous case. The scan started on March 12th at 20:13:23 and ended at 20:21:45, lasting approximately eight minutes. A total of 9,217 unique hosts were scanned. Port 6129 is used by Dameware. The attacker could be either looking for already compromised hosts or hosts with a vulnerable Dameware service running. Versions of Dameware service are known to have a bufferoverflow vulnerability, CAN-2003-1030. The host was resolved to "0x3ef3aea1.bynxx5.adsl-dhcp.tele.dk" and no reports were found on it. Full traffic logs should be investigated to see if any of the scanned hosts responded (to the SYN probes). A response means that a host on the network is probably compromised. Furthermore, firewall rules should be in place to filter traffic destined to ports that are not required to be contacted from the Internet.

3.2.5 218.197.122.51

A total of 10,741 SYN scans were logged from this host all targeting port 17300. This port is associated with a trojan called Kuang2 [65], which can be used to upload/download/delete/execute files, steal passwords and do other petty pranks. The trojan creates a backdoor listening on port 17300 on the compromised host. The SYN scan started on March 12th at 07:14:57 and ended at 07:30:08 the same day. A total of 9,382 unique hosts were scanned in approximately 15 minutes, indicating the use of an automated program. This IP address was registered to Transportational Institute,Wuhan University of

Technology, in China. Full traffic logs should be investigated to see if there were any responses. Hosts that responded, if at all there were any, should be investigated for the Kuang2 trojan. Traffic destined to this port should be blocked at the perimeter firewall.

3.3 Top five hosts- Alert logs

Unlike in the previous sections (3.1 and 3.2), we will be taking a slightly different approach in obtaining our top five hosts for alert logs. First, we will look at the top five alerts excluding the "MY.NET.30.4 activity", "MY.NET.30.3 activity" and "SMB Name Wildcard" alerts. Refer to Appendix A-5 as to why these alerts were not considered. The following table shows the top five most frequently occurred alerts.

Alert	Number of times the alert was triggered
High port 65535 tcp - possible Red Worm - traffic	6494
connect to 515 from outside	2203
EXPLOIT x86 NOOP	1353
Null scan!	1048
NMAP TCP ping!	668

In order to build our list of top five hosts for the alert logs, external hosts that generated the maximum number of each of the above five alerts were considered. Refer to the following table for the list of hosts.

Alert	Sourceip	Number of logs
High port 65535 tcp - possible Red Worm - traffic	220.37.240.35	1120
connect to 515 from outside	68.32.127.158	2195
EXPLOIT x86 NOOP	24.31.31.188	539
Null scan!	63.251.52.75	576
NMAP TCP ping!	63.211.17.228	163

3.3.1 220.37.240.35

All 1,120 alerts associated with this host seem to be false positives. The host was resolved to "YahooBB220037240035.bbtec.net" and was found to be registered to Softbank BB Corporation, a company in Japan. After visiting the Web site, www.bbtec.net, I came to believe that the Company has partnered with Yahoo in offering various broadband services. The FQDN of the host suggests that the external host in question is one such server that is hosting some broadband services. 220.37.240.35 and MY.NET.53.55 were having a two-way conversation on ports 65535 and 4576 respectively, starting on March 8th at 10:00:49 and ending at 10:14:59 the same day. The alerts were triggered because of the presence of the port 65535 on one side of the communication. Refer to Appendix A-9 for a snippet of sample logs. As per the description of the Red Worm [67] (alias adore worm), MY.NET.53.55 is not a victim unless it is being controlled by an attacker to access another infected host (If a host were compromised with adore worm, a backdoor would be listening on port 65535 on the victim; but in this scenario port 65535 was associated with the external host).

3.3.2 68.32.127.158

The packets coming from this external IP were responsible for triggering

2,195 “connect to 515 from outside” alerts. This is not a scan activity because the source port (50832) and the destination port (515) were the same in all instances. The alerts were triggered because 68.32.127.158 was sending packets to MY.NET.24.15 destined to port 515, which is usually used for print services (lpr). The external host was resolved to “pcp0011023458pcs.arlngt01.va.comcast.net” suggesting that it might be a home user trying to access some print services. However, Tillman Hodgson [68] has discussed the alerts between the same pair of hosts, but the source port involved in those alerts was 797, a privileged port. Bobby Noell [58] indicated that this traffic is not malicious in nature. But why would a privileged source port be involved if a home user is trying to access the print services? Raw packet traces should be examined to determine the nature of the traffic. Lpr printer services are known to have some security vulnerabilities (BugTrack ID: 1712, CVE #: CVE-2000-0917) that could lead to the compromise of a system. It is highly recommended that access to these services from the Internet is restricted. Policies and controls (like firewall rules) should be put in place to block in-bound traffic destined to port 515.

3.3.3 24.31.31.188

This external host was involved in attacking hosts that were running Web servers. On March 11th this host scanned about 6,786 unique hosts for port 80 using SYN scan technique. On March 12th, a total of 539 “EXPLOIT x86 NOOP”, five “MY.NET.30.3 activity” and one “MY.NET.30.4 activity” alerts were logged. Port 80 was the destination port in all cases, indicating that the attacker did his/her reconnaissance before launching the exploit code. A total of 15 unique hosts (12 of which were still running web services as of 01/08/05) were actively targeted by this external host using some exploit. Refer to Appendix A-6 for the list. Raw packet traces should be examined and the individual hosts should be investigated for signs of infection. The external IP address was registered to Comcast (formerly AT&T broadband) and was resolved to “c-24-31-31-188.mn.client2.attbi.com”. The University Web servers should be completely patched and the IDS personnel at the University should report this IP address to Comcast.

3.3.4 63.251.52.75

After examining the alert and scan logs, it is determined that this external host was involved in hostile activity. Unless there was something wrong with the host, there is no reason for it to send TCP segments with all flags (URG, ACK, PSH, RST, SYN, FIN) off. Furthermore, the scan logs show that the host was sending TCP segments with all kinds of out-of-specification combinations of TCP flags. The sheer number of such packets suggests that this external host is up to something malicious. A total of 576 alert and 163 scan logs associated with 63.251.52.75 (as source IP) were logged on March 8th, 9th, 10th and 12th. The source and destination ports were both set to 0 with occasional exceptions. A total of four unique hosts were targeted. Refer to Appendix A-7 for the hosts. The host was resolved to www.shockwave.com an online gaming site. Jorge Perez [70] and Mike Shannon [69] have discussed about the host and its malicious activity. The IP address was also listed at Dshield.org with

about 386 reports. While the real purpose of these out-of-spec packets cannot be determined, they can be used for various purposes like reconnaissance, as a decoy or, as an evasion technique. The external host should be blacklisted and blocked. To avoid such activity in the future, the perimeter firewall should be configured to drop out-of-spec packets.

3.3.5 63.211.17.228

The traffic coming from this external host (proximitycheck1.allmusic.com) seems to be benign in nature. Similar kinds of traffic from this host have been reported in [71, 73] and the host has also been listed on Dshield.org with 3768 reports. However, a closer look at the alert logs shows that there's no NMAP scanning activity as indicated by the alerts.

03/08-00:15:33.151519 [**] NMAP TCP ping! [**] 63.211.17.228:80 -> MY.NET.1.3:53
03/08-00:15:33.151529 [**] NMAP TCP ping! [**] 63.211.17.228:53 -> MY.NET.1.3:53
03/08-00:08:31.278364 [**] NMAP TCP ping! [**] 63.211.17.228:80 -> MY.NET.1.3:53
03/08-00:08:31.278386 [**] NMAP TCP ping! [**] 63.211.17.228:53 -> MY.NET.1.3:53
…
03/12-15:25:32.469223 [**] NMAP TCP ping! [**] 63.211.17.228:80 -> MY.NET.69.178:4889

The logs have two sets of packets (almost at the same time), one going from port 80 on 63.211.17.228 to port 53 on MY.NET.1.3 and the other going from port 53 on 63.211.17.228 to port 53 on MY.NET.1.3. Occasionally there were packets from port 80 on 63.211.17.228 going to an ephemeral port on other internal hosts. MY.NET.1.3 is one of the University's name servers running a Web server on port 80. As the FQDN of the host suggests, I guess this traffic is aimed at improving the quality-of-service for users of www.allmusic.com. Ashley Thomas [72] and Andrew Jones [74] have analyzed this traffic in detail and found some good evidence to support the argument that these alerts are false positives.

3.4 Top five Ports

In this section we will briefly look at the top five ports that were targeted. The list of our ports is derived by considering the top five destination ports with the maximum number of hits from external hosts. All three categories of logs were considered, but the ports with the most hits came from the scan logs. The table below shows the ports that we are going to briefly discuss about.

Destination port	Number of hits	Number of sources	Number of Targets	Type of activity	Time
6129	93549	26	15695	Scanning and active targeting*	March 8th – March 12th
80	88695	221	15651	Scanning and active targeting*	March 8th – March 12th
20168	62884	12	15370	Scanning	March 8th – March 12th
4899	49605	16	15158	Scanning	March 9th – March 12th
21	43924	31	15026	Scanning and active targeting*	March 8th – March 12th

* - It was determined whether there was active targeting or not by examining the alert logs. If there were alerts other than "MY.NET.30.3 activity" and "MY.NET.30.4 activity" associated with a particular internal host, then it was considered that, that host was actively targeted.

Port 6129, as mentioned earlier in section 3.2.4, is associated with Dameware, a tool used for remotely administering a system. Even though not intended to be used for malicious purposes, this is one of the many tools that attackers use to administer an already-compromised host [27]. Furthermore, 26 'EXPLOIT x86 NOOP' alert logs where the destination port was 6129 were captured. According to CAN 2003-1030 [75] some versions of Dameware are vulnerable to bufferoverflow attacks. Considering both the alert and the scan logs, I feel the attackers were trying to scan for either already-compromised hosts which had Dameware installed (by someone else) or exploit the Dameware service itself to gain administrative access to the hosts. Port 6129 was listed in the top 20 targeted ports at Dsheild.org on March 9th, 10th, 11th and 12th.

Port 80 was the next most hit port. Not all 221 external hosts involved were malicious. However, there were some external hosts that were involved in large scale scanning activity for port 80. There were a couple of 'EXPLOIT x86 NOOP' alerts logged which suggests that some hosts were actively targeted. Refer to [76] for the CVE numbers associated with port 80. It is important that the IT department maintains an inventory of all authorized Web servers and restricts access to other unauthorized Web servers at the perimeter using firewall rules. A process should be put in place where the IT department is notified when a new Web server is required so that access from the Internet could be granted. A process should also be put in place to make sure that up-to-date patches are installed on all authorized Web servers. The port was listed in the top 20 targeted ports on Dshield.org between March 8th and March 12th.

Port 20168 is associated with a worm called "Lovegate". The worm copies itself via network shares and mass-mails itself using its own SMTP engine. The worm may also drop a backdoor listening on port 20168 to give console access to the remote attacker [77]. Raw packet traces should be investigated to determine if any of the scanned hosts responded to the scan probes. In-bound connections to port 20168 should also be blocked at the perimeter level.

Port 4899 is associated with another tool called "Radmin", used for remotely administering Windows machines [78]. Scans against this host have also been reported by other sources [79, 80]. While the exact reason for the scans is not known, there has been speculation that there could be some unpublished exploit in the wild targeting radmin or that the attackers could be looking for default installations of the radmin service. Full packet traces should be examined to see if any of the scanned hosts responded. In-bound connections to port 4899 should be blocked at the perimeter level.

Port 21 is a port associated with the File Transfer Protocol; 21 used to transfer control commands between the client and the FTP server. Apart from the scanning logs, there were numerous "FTP password attempt" and "FTP DoS ftpd globbing" alerts which suggest that there's some active targeting going on. As FTP is one of those services which cannot be blocked completely, an inventory of authorized FTP servers should be maintained and the perimeter firewall should be configured to allow access to those authorized servers only.

A process should also be designed to regularly install up-to-date patches. Furthermore, a process should be designed where the IT department is notified before introducing new FTP servers.

4.0 Three most suspicious external hosts

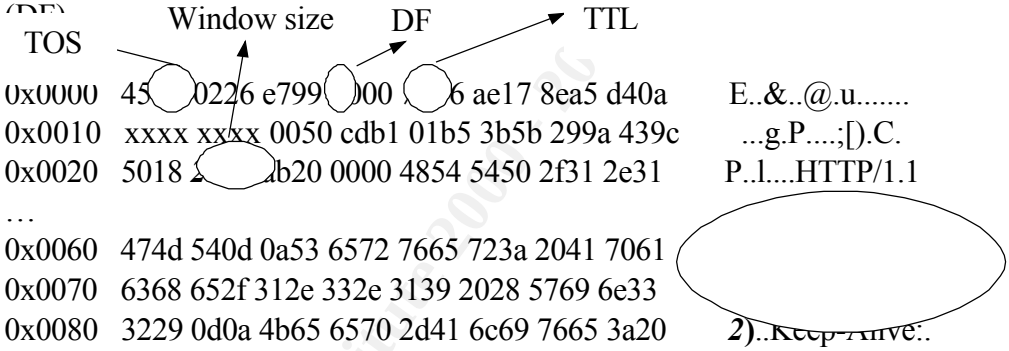
In this section we will be discussing some details about three suspicious external hosts. Note: Passive OS fingerprinting (pof) techniques are used to guess the operating systems of the remote hosts. Inasmuch as the behavior of the IP stack can be customized on many operating systems, the results obtained using pof techniques are not 100% accurate and could be misleading sometimes.

4.1 IP: 142.165.212.10

Hostname: www.infotaxi.ca

This host was chosen because there were some internal machines which responded to SYN probes from this host to port 27374 – a port associated with SubSeven trojan. Furthermore, the contents of the Web site seemed to be very benign, which lead me to believe that the external host could itself be a victim. So, I wanted to find some details about the host that could support my hunch and therefore, tried connecting to www.infotaxi.ca and queried for a non-existing Web page (test1.htm). The following is a partial packet trace of the response from the Web server (the destination IP has been masked).

16:02:11.691999 142.165.212.10.80 > xx.xx.xx.xx.52657: P 1:511(510) ack 461 win 8300 (DF)




```
0x0000   45   0226 e799  000   6 ae17 8ea5 d40a      E..&..@.u.....
0x0010   xxxx xxxx 0050 cdb1 01b5 3b5b 299a 439c      ...g.P....;[).C.
0x0020   5018 2   4b20 0000 4854 5450 2f31 2e31      P..l....HTTP/1.1
…
0x0060   474d 540d 0a53 6572 7665 723a 2041 7061
0x0070   6368 652f 312e 332e 3139 2028 5769 6e33
0x0080   3229 0d0a 4b65 6570 2d41 6c69 7665 3a20      2)..Keep-Alive:.
```

The value of the TTL field in the above packet trace was 0x75= 117. It would be reasonable to estimate that the source is 11 hops away and that the packet could have an initial TTL of 128. As shown above, the window size advertised was 0x206c=8300. The Don't fragment bit was set and the TOS field was set to zero. The ASCII text indicates that the Web server was a Windows version of Apache 1.3.19. Putting this all together and using some of the passive operating system fingerprinting [46] information, we can match the profile to that of a Win9x/NT
box. So, it is likely possible that the scans came from an already-compromised Windows 9x/NT box. The host was registered to

```
OrgName:    SaskTel
OrgID:      SASK
Address:    c/o Sasknet Policy
Address:    8th Flr 2121 Saskatchewan Dr
City:       Regina
StateProv:  SK
PostalCode: S4P-3Y2
Country:    CA
```

Full registration information can be obtained by querying for 142.165.212.10 at http://www.arin.net/whois/

4.2 63.251.52.75

Hostname: www.schockwave.com

Of the 15 external hosts that were closely examined, this was one of the two hosts that were deemed as high-risk. Using the earlier technique, it is determined that the remote operating system was a Unix machine. Below is a partial packet trace of a response to a request for a non-existing page from the host (the destination IP has been masked).

10:27:29.048122 63.251.52.75.80 > xx.xx.xx.xx.48568: P 1:525(524) ack 463 win 65160 <nop,nop,timestamp 46748617 50402200> (DF)

0x0000 45***00*** 0240 a3bc ***4***000 ***f3***06 2927 3ffb 344b E..@..@...)'?.4K
0x0010 xxxx xxxx 0050 bdb8 3569 f371 a089 1a6a E:.S.P..5i.q...j
0x0020 8018 ***fe88*** 5356 0000 0101 080a 02c9 53c9 ....SV.....S.
…
0x0060 3035 2031 353a 3330 3a34 3920 474d 540d 05.15:30:49.GMT.
0x0070 0a53 6572 7665 723a 2041 7061 6368 652f ***.Server:.Apache***/
0x0080 312e 332e 3238 2028 556e 6978 2920 5265 ***1.3.28.(Unix).***Re

The TOS filed was set to 0, the Don't fragment bit was set, the TTL was set to 243 and the window size was 65160. The TTL value suggests that the source is 12 hops away and that the initial value could be 255. The ASCII part of the trace indicates that the host is running Apache 1.3.28 on a Unix platform. Even though, the window size doesn't match, the profile closely matches to that of a Solaris 2.x host [46]. One interesting fact is that I couldn't ping the host even though I was able to access the home page on www.shockwave.com. This suggests that the host is behind a firewall/filter and that ICMP messages are being blocked. Dshield.org received about 474 records indicating that this host targeted a total of 51 other hosts. Full registration information about this host could be found by querying for 63.251.52.75 at http://www.dshield.org/ipinfo.php. The host was found to be registered to

```
CustName:    Shockwave.com
Address:     650 Townsend Street, #450
City:        San Francisco
StateProv:   CA
PostalCode:  94103
Country:     US
RegDate:     2000-08-24
Updated:     2000-08-24
```

4.3 24.31.31.188

Hostname: c-24-31-31-188.mn.client2.attbi.com

This host was selected because this was the second of the two external hosts that were deemed as high-risk to the internal network. As the FQDN suggests, it could be a home user with broadband connection. As active scanning is not permitted, ping requests were sent and the responses were analyzed. It should be kept in mind that the result may not be as accurate as in the first two cases. Following is the raw packet trace of a ping reply from 24.31.31.188

11:56:36.211127 24.31.31.188 > xx.xx.xx.xx: icmp: echo reply

```
0x0000   45**00** 0054 5280 **0**000 **70**01 7bc0 181f 1fbc      E..TR...p.{.....
0x0010   xxxx xxxx 0000 ead1 c057 0001 54fc eb41      E:.S.....W..T..A
0x0020   2794 0200 0809 0a0b 0c0d 0e0f 1011 1213      '...............
0x0030   1415 1617 1819 1a1b 1c1d 1e1f 2021 2223      ..............!"#
0x0040   2425 2627 2829 2a2b 2c2d 2e2f 3031 3233      $%&'()*+,-./0123
0x0050   3435 3637                                     4567
```

The bytes/nibbles that were bolded represent the TOS, Don't Fragment bit and the TTL fields in the IP header. These fields were set to 0, 0 and 112 respectively. Unlike in the previous two sections, there's no TCP window size as ICMP is a layer 3 protocol. The TTL filed suggests that the sender was 16 hops away and that the initial value could be 128. With the exception of the DF bit, the ones that closely match this profile are that of Netware and Windows 9x/NT/2000. Since, we guessed that this one could be a home user with a broadband connection; I am going to say that the remote OS is some flavor of Windows. The IP was registered to Comcast.

```
OrgName:    Comcast Cable Communications Holdings, Inc
OrgID:      CCCH-3
Address:    1800 Bishops Gate Blvd
City:       Mt Laurel
StateProv:  NJ
PostalCode: 08054
Country:    US
```

Full registration information can be found by querying for 24.31.31.188 at http://www.arin.net/whois/

5.0 Analysis

For the purpose of this report, I have chosen to analyze the alert, scan and OOS logs between March 8th and March 12th (but the OOS files contained logs from March 12th – March 16th). In order to successfully complete my analysis, I first used three Perl scripts, one for each of the categories, to parse the log files. The output of the Perl scripts was three SQL files that contained insert statements. Three tables were then created in an Oracle instance and the SQL files that were generated were uploaded into these tables by running them (SQL files) manually. The structure for each of the tables and the Perl scripts can be found in Appendix C. I then used MS Access, as a front-end, to query my tables. I took this approach instead of employing some tools like, SnortSnarf or ACID, because I wanted to hone my analysis skills by extracting and connecting relevant data myself. While I completely encourage use of such tools, I also feel that one should be trained to work in the absence of such tools. Working with raw data helps cultivate the needed vision and analytical ability that a true analyst requires.

References

[1] www.snort.org, 01/17/2005

[2] www.tcpdump.org, 01/17/2005

[3] Kesavamatham, SaiPrasad. Intrusion Detection and Analysis, July 7 2003, http://www.giac.org/practical/GCIA/SaiPrasad_Kesavamatham_GCIA.pdf, 01/10/17

[4] http://azureus.sourceforge.net/faq.php#17, 12/03/04

[5] http://www.aunty-spam.com/archives/2004/11/09/is-bittorrent-traffic-going-to-bring-down-the-internet, 12/05/2004

[6] http://www.p2pwatchdog.com/packet_bittorrent.html, 12/05/2004

[7] http://forums.thatcomputerguy.us/lofiversion/index.php/t7972.html, 12/05/2004

[8] http://forums.afterdawn.com/thread_view.cfm/14/42158, 12/05/2004

[9] http://azureus.sourceforge.net/faq.php#23, 12/05/2004

[10] http://www.sidewalkcrusaders.com/bthowto/btstart.html, 12/05/2004

[11] BitTorrent IRC channels, http://home.quicknet.nl/qn/prive/romeria/irc.htm, 12/05/2004

[12] http://www.emule-project.net/home/perl/news.cgi?l=1&cat_id=12, 12/05/2004

[13] http://www.emule-project.net/home/perl/help.cgi?l=1&topic_id=122&rm=show_topic, 2/05/2004

[14] http://www.g4techtv.com/screensavers/features/46941/Dark_Tip_eMule_Plus.html, 12/05/2004)

[15] http://bittorrent.com/protocol.html, 12/05/2004

[16] http://www.dslreports.com/faq/4493, 12/06/2004

[17] http://www.dshield.org/pipermail/unisog/2002-May/009063.php, 12/06/2004

[18] http://www.dshield.org/pipermail/unisog/2003-April/010349.php, 12/06/2004.

[19] Wu, Marcus. GCIA Practical assignment version 3.3, January 16 2003, http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00120.html, 12/07/2004

[20] Barrett, Ryan. GCIA Practical Assignment V 3.4 , March 24 2004, http://www.giac.org/practical/GCIA/Ryan_Barrett_GCIA.pdf, 2/07/2004

[21] http://archives.neohapsis.com/archives/snort/2002-05/0121.html, 12/06/2004

[22] IRC related Snort rules, http://coders.meta.net.nz/~perry/irc.rules, 01/17/2005

[23] IRC Basics, http://bre.klaki.net/programs/tircproxy/manual/tircproxy-5.html, 12/06/2004

[24] Klaus Zeuge, Troy Rollo and Ben Mesander, The Client-To-Client Protocol Version One, May 1993, http://www.invlogic.com/irc/ctcpprot_07.html, 12/06/2004

[25] How to use XDCC, http://debateroom.com/tutorial_xdcc.htm, 12/06/2004

[26] What is XDCC, http://groups-beta.google.com/group/alt.irc/browse_thread/thread/4d88b73c919b75f5?q=XDCC&start=3&hl=en&lr=&ie=UTF-8&rnum=4, 12/06/04

[27] TonikGin, XDCC – An .EDU Admin's Nigthmare, September 11 2002, http://www.ncsu.edu/itd/security/papers/EduHacking.html, 12/08/04

[28] Bottler – XDCC Bot Interface, http://sourceforge.net/projects/bottler/, 12/08/04

[29] http://iroffer.org/, 12/08/04

[30] Stadler, Philipp. GCIH Practical version 2.1, April 1-7 2002, http://www.giac.org/practical/Philipp_Stadler_GCIH.doc, 12/10/04

[31] Affeld, James. GCIA Practical version 3.5, June 3 2004, http://www.giac.org/practical/GCIA/James_Affeld_GCIA.pdf, 12/11/04

[32] http://archives.neohapsis.com/archives/snort/2002-12/0354.html, 12/11/04

[33] Mills, David. Network Time Protocol – Version 2, September 1989, http://www.faqs.org/ftp/rfc/rfc1119.pdf, 12/11/04

[34] Mills, David. Network Time Protocol, September 985, http://www.faqs.org/rfcs/rfc958.html, 12/11/04

[35] Mills, David. Network Time Protocol – Version 3, March 1992, http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1305.html, 12/11/04

[36] Mills, David. Network Time Protocol – Version 1 July 1988, http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1059.html, 12/11/04

[37] Frasunek, Przemyslaw. NTPD bufferoverflow exploit code, April 4 2001, http://www.securityfocus.com/archive/1/174011, 12/11/04

[38] NTPD bufferoverflow exploit code, http://www.safechina.net/www_hack_co_za/redhat/7.0/ntpdx.c, 12/11/04

[39] Rautiainen, Sami. Adore worm analysis, April 2001, http://www.f-secure.com/v-descs/adore.shtml, 12/13/04

[40] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0414, 12/13/04

[41] Turkia, Miika. GCIA Practical assignment version 2.7, January 28 2001, http://www.giac.org/practical/Miika_Turkia_GCIA.html, 12/13/2004

[42] Larratt, Glenn. GCIA Practical assignment version 3.0, http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html#NTPDX, 12/13/2004

[43] http://www.dshield.org/pipermail/intrusions/2002-September/005076.php, 12/13/2004

[44] http://cvs.sourceforge.net/viewcvs.py/snort/snort/Attic/RULES.SAMPLE?rev=1.7, 12/20/2004

[45] http://www.scn.rain.com/pub/security/checkers/snort-lib, 12/20/2004

[46] http://project.honeynet.org/papers/finger/traces.txt, 12/21/2004

[47] http://www.sans.org/y2k/092000.htm, 12/21/2004

[48] http://www.cert.org/advisories/CA-99-08-cmsd.html, 12/21/2004

[49] http://www.cert.org/advisories/CA-99-05-statd-automountd.html, 12/21/2004

[50] http://www.cert.org/advisories/CA-98.11.tooltalk.html, 12/21/2004

[51] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0003, 12/21/2004

[52] Haywood, James. GCIA Practical assignment, http://www.giac.org/practical/GCIA/James_Haywood_GCIA.pdf, 12/21/2004

[53] Bernstein, Michael. GCIA Practical assignment version 3.4, March 14 2004, http://www.giac.org/practical/GCIA/Michael_Bernstein_GCIA.pdf, 12/23/2004

[54] Sternudd, Patrik. GCIA Practical assignment version 3.4, June 24 2004, http://hem.bredband.net/flank/docs/GCIA.pdf, 12/23/2004

[55] K. Ramakrishnan, S. Floyd, D. Black. The Addition of ECN to IP, September 2001, http://www.faqs.org/rfcs/rfc3168.html, 12/30/2004

[56] http://64.233.167.104/search?q=cache:WLcjRW2GhTYJ:62.197.96.39:12836/+%2262.111.194.65%22+&hl=en, 12/30/2004

[57] http://64.233.167.104/search?q=cache:IMu7lQGvfVoJ:24.94.126.24:13766/+%2262.111.194.65%22+&hl=en, 12/30/2004

[58] Noell, Bobby. GCIA Practical assignment version 3.4, May 14 2004, http://www.giac.org/practical/GCIA/Bobby_Noell_GCIA.pdf, 12/30/2004

[59] Active FTP vs Passive FTP, http://slacksite.com/other/ftp.html, 01/03/2005

[60] Bakkers, Coen. GCIA Practical assignment version 3.5, June 29 2004, http://www.giac.org/practical/GCIA/Coen_Bakkers_GCIA.pdf, 01/04/2005

[61] http://64.233.167.104/search?q=cache:mo68xNka1FAJ:ivo.zelluf.com/log.asp+%22213.157.171.109%22&hl=en, 01/05/2005

[62] http://www.nmx.fromtheshadows.net/blockedips.txt, 01/05/2005

[63] http://www.dshield.org/pipermail/intrusions/, 01/05/2005

[64] http://lists.virus.org/bugtraq-0204/msg00152.html, 01/05/2005

[65] http://www.757.org/~joat/ports.php?action=view&program_no=40, 01/05/2005

[66] Kroeger, Tim. GCIA Practical assignment version 3.4, May 18 2004, http://www.giac.org/practical/GCIA/Tim_Kroeger_GCIA.pdf, 01/07/2005

[67] Adore worm, http://www.sophos.com/virusinfo/analyses/linuxadore.html, 01/07/2005

[68] Hodgson, Tillman. GCIA Practical assignment version 3.4, June 10 2004, http://www.giac.org/practical/GCIA/Tillman_Hodgson_GCIA.pdf, 01/08/2005

[69] Shannon, Mike. GCIA Practical assignment version 3.4, February 19 2004, http://www.giac.org/practical/GCIA/Mike_Shannon_GCIA.pdf, 01/09/2005

[70] Perez, Jorge. GCIA Practical assignment version 3.3, http://www.giac.org/practical/GCIA/Jorge_Perez_GCIA.pdf, 01/09/2005

[71] http://linuxgw.phj.hu/acidlab/acid_qry_main.php?new=1&sig%5B0%5D=%3D&sig%5B1%5D=103&sig_type=1&submit=Query+DB&num_result_rows=-1, 01/09/2005

[72] Thomas, Ashley. GCIA Practical assignment version 3.3, March 17 2003, http://www.giac.org/practical/GCIA/Ashley_Thomas_GCIA.pdf, 01/10/2005

[73] http://seclists.org/lists/incidents/2002/Dec/0084.html, 01/10/2005

[74] Jones, Andrew. GCIA Practical assignment version 3.3, http://www.giac.org/practical/GCIA/Andrew_Jones_GCIA.pdf, 01/10/2005

[75] http://www.saintcorporation.com/cgi-bin/demo_tut.pl?tutorial_name=Dameware_vulnerabilities.html&fact_color=doc&tag=, 01/10/2005

[76] http://www.colasoft.com/resources/port.php?id=80, 01/10/2005

[77] W32.Lovegate worm, April 01 2003, http://www.k7computing.com/newsinfo/LoveF.htm, 01/10/2005

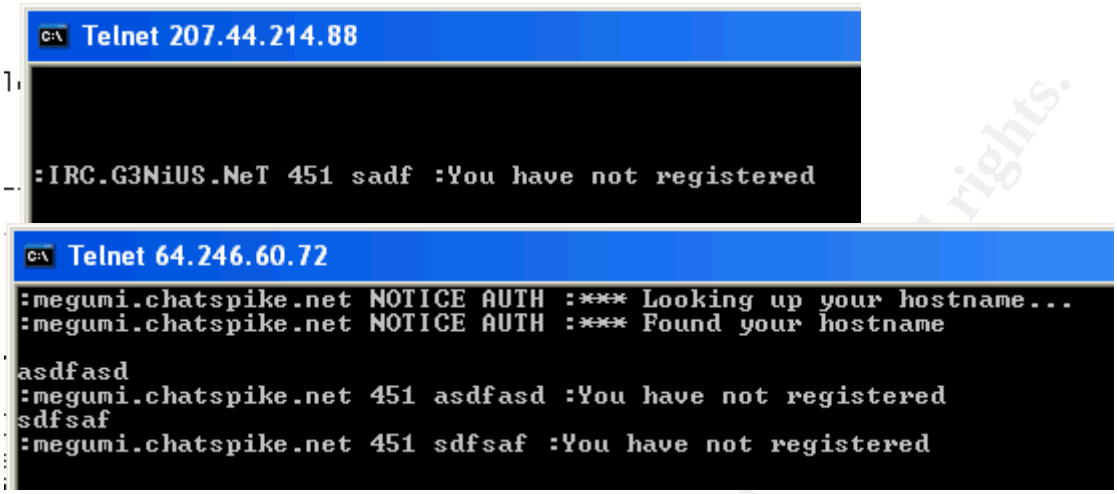
[78] Scheidell, Michael. Radmin Default install options vulnerability, September 2 2002, http://www.securityfocus.com/archive/1/290099/2002-09-01/2002-09-07/0, 01/10/2005

[79] http://www.secnap.com/security/radmin001.html, 01/10/05

[80] http://seclists.org/lists/incidents/2004/Apr/0018.html, 01/10/05

Appendix A

Item 1:



Item 2:

SYN packets to MY.NET.66.29:

Mar 8 00:34:36 81.112.172.203:3156 -> MY.NET.66.29:7755 SYN ******S*
Mar 8 00:34:39 81.112.172.203:3156 -> MY.NET.66.29:7755 SYN ******S*
Mar 8 01:36:04 62.23.19.42:3006 -> MY.NET.66.29:20168 SYN ******S*
Mar 8 02:23:23 83.135.68.141:1846 -> MY.NET.66.29:4898 SYN ******S*
Mar 8 07:40:59 129.24.232.173:4960 -> MY.NET.66.29:4000 SYN ******S*
Mar 8 07:41:53 80.48.119.3:3283 -> MY.NET.66.29:20168 SYN ******S*
Mar 8 08:10:36 68.114.42.4:220 -> MY.NET.66.29:6129 SYN ******S*
Mar 9 00:41:27 143.239.181.85:4802 -> MY.NET.66.29:80 SYN ******S*
Mar 9 05:11:40 134.2.78.155:2428 -> MY.NET.66.29:21 SYN ******S*
Mar 9 08:30:46 211.237.212.40:4657 -> MY.NET.66.29:4489 SYN ******S*
Mar 9 09:23:52 200.251.242.151:4099 -> MY.NET.66.29:80 SYN ******S*
Mar 9 09:55:38 66.124.32.2:1388 -> MY.NET.66.29:3000 SYN ******S*
Mar 9 10:45:05 61.240.36.137:4474 -> MY.NET.66.29:4899 SYN ******S*
Mar 9 18:39:37 170.94.47.66:14807 -> MY.NET.66.29:6129 SYN ******S*
Mar 9 18:54:05 68.185.128.118:3128 -> MY.NET.66.29:80 SYN ******S*
Mar 9 18:54:08 68.185.128.118:3128 -> MY.NET.66.29:80 SYN ******S*
Mar 9 20:10:48 80.19.53.202:3455 -> MY.NET.66.29:6129 SYN ******S*
Mar 10 00:01:35 129.255.144.162:4219 -> MY.NET.66.29:20168 SYN ******S*
Mar 10 06:09:14 212.100.97.10:1530 -> MY.NET.66.29:8080 SYN ******S*
Mar 10 09:26:21 130.191.237.82:3374 -> MY.NET.66.29:6129 SYN ******S*
Mar 10 11:56:10 4.22.114.47:2579 -> MY.NET.66.29:1257 SYN ******S*
Mar 10 16:55:58 67.100.216.5:220 -> MY.NET.66.29:21 SYN ******S*
Mar 10 18:43:27 81.48.25.9:4171 -> MY.NET.66.29:6129 SYN ******S*
Mar 11 08:51:58 24.107.132.7:220 -> MY.NET.66.29:6129 SYN ******S*
Mar 12 00:28:09 213.135.252.86:220 -> MY.NET.66.29:21 SYN ******S*
Mar 12 07:24:07 218.197.122.51:1809 -> MY.NET.66.29:17300 SYN ******S*
Mar 12 08:04:52 68.77.58.92:1215 -> MY.NET.66.29:4899 SYN ******S*
Mar 12 23:05:09 202.155.56.180:2522 -> MY.NET.66.29:6129 SYN ******S*

SYN packets to MY.NET.109.86:

Mar 8 00:36:29 81.112.172.203:1564 -> MY.NET.109.86:7755 SYN ******S*
Mar 8 01:26:02 132.254.57.159:1833 -> MY.NET.109.86:6129 SYN ******S*
Mar 8 04:32:53 81.208.106.64:4939 -> MY.NET.109.86:7100 SYN ******S*
Mar 8 14:54:22 66.114.254.117:3695 -> MY.NET.109.86:80 SYN ******S*
Mar 8 21:28:34 212.64.200.62:1407 -> MY.NET.109.86:20168 SYN ******S*
Mar 8 23:35:12 218.165.104.239:1872 -> MY.NET.109.86:17300 SYN ******S*
Mar 9 01:08:37 83.121.102.116:3525 -> MY.NET.109.86:20168 SYN ******S*
Mar 9 02:56:32 162.40.123.222:3057 -> MY.NET.109.86:80 SYN ******S*
Mar 9 05:06:16 64.166.194.201:1181 -> MY.NET.109.86:3000 SYN ******S*

Mar 9 05:13:34 134.2.78.155:2465 -> MY.NET.109.86:21 SYN ******S*
Mar 9 07:15:03 213.157.171.109:4083 -> MY.NET.109.86:80 SYN ******S*
Mar 9 07:14:18 213.157.171.109:4159 -> MY.NET.109.86:80 SYN ******S*
Mar 9 08:32:49 211.237.212.40:4746 -> MY.NET.109.86:4489 SYN ******S*
Mar 9 09:24:36 200.251.242.151:2559 -> MY.NET.109.86:80 SYN ******S*
Mar 9 10:52:30 147.94.116.130:1567 -> MY.NET.109.86:20168 SYN ******S*
Mar 9 11:37:37 211.22.194.61:2318 -> MY.NET.109.86:4899 SYN ******S*
Mar 9 22:24:22 218.85.85.97:4224 -> MY.NET.109.86:17300 SYN ******S*
Mar 10 01:58:50 218.58.155.178:3672 -> MY.NET.109.86:4899 SYN ******S*
Mar 10 03:09:52 163.25.148.155:1938 -> MY.NET.109.86:20168 SYN ******S*
Mar 10 06:10:53 212.100.97.10:3370 -> MY.NET.109.86:8080 SYN ******S*
Mar 10 12:02:41 195.226.52.1:2282 -> MY.NET.109.86:6129 SYN ******S*
Mar 10 12:02:44 195.226.52.1:2282 -> MY.NET.109.86:6129 SYN ******S*
Mar 11 00:48:07 24.202.208.210:1689 -> MY.NET.109.86:4899 SYN ******S*
Mar 11 03:09:01 213.35.181.187:4731 -> MY.NET.109.86:6129 SYN ******S*
Mar 11 05:17:27 80.131.248.55:3481 -> MY.NET.109.86:80 SYN ******S*
Mar 11 05:51:11 137.204.213.227:2482 -> MY.NET.109.86:8150 SYN ******S*
Mar 11 07:48:12 24.31.31.188:1437 -> MY.NET.109.86:80 SYN ******S*
Mar 11 08:34:25 202.40.176.20:1802 -> MY.NET.109.86:4899 SYN ******S*
Mar 12 06:16:12 66.117.30.203:36058 -> MY.NET.109.86:21 SYN ******S*
Mar 12 07:26:04 218.197.122.51:2733 -> MY.NET.109.86:17300 SYN ******S*
Mar 12 08:05:37 68.77.58.92:4549 -> MY.NET.109.86:4899 SYN ******S*
Mar 12 17:28:53 66.188.24.1:3217 -> MY.NET.109.86:20168 SYN ******S*
Mar 12 18:19:24 213.168.217.148:3361 -> MY.NET.109.86:6129 SYN ******S*
Mar 12 20:18:00 62.243.174.161:3036 -> MY.NET.109.86:6129 SYN ******S*
Mar 12 23:10:20 202.155.56.180:2131 -> MY.NET.109.86:6129 SYN ******S*

SYN packets to MY.NET.53.204

Mar 8 00:34:04 81.112.172.203:3624 -> MY.NET.53.204:7755 SYN ******S*
Mar 8 01:21:03 132.254.57.159:3236 -> MY.NET.53.204:6129 SYN ******S*
Mar 8 02:21:47 83.135.68.141:3687 -> MY.NET.53.204:4898 SYN ******S*
Mar 8 04:32:35 81.208.106.64:4919 -> MY.NET.53.204:7100 SYN ******S*
Mar 8 07:40:25 129.24.232.173:1523 -> MY.NET.53.204:4000 SYN ******S*
Mar 8 07:41:18 80.48.119.3:2286 -> MY.NET.53.204:20168 SYN ******S*
Mar 9 05:03:31 64.166.194.201:1554 -> MY.NET.53.204:3000 SYN ******S*
Mar 9 05:11:09 134.2.78.155:2946 -> MY.NET.53.204:21 SYN ******S*
Mar 9 07:03:40 213.157.171.109:3630 -> MY.NET.53.204:80 SYN ******S*
Mar 9 07:04:19 213.157.171.109:3571 -> MY.NET.53.204:80 SYN ******S*
Mar 9 08:30:12 211.237.212.40:1432 -> MY.NET.53.204:4489 SYN ******S*
Mar 9 08:30:13 211.237.212.40:1432 -> MY.NET.53.204:4489 SYN ******S*
Mar 9 09:23:40 200.251.242.151:3590 -> MY.NET.53.204:80 SYN ******S*
Mar 9 09:54:57 66.124.32.2:2001 -> MY.NET.53.204:3000 SYN ******S*
Mar 9 15:26:02 68.63.203.236:1852 -> MY.NET.53.204:5900 SYN ******S*
Mar 9 18:39:33 170.94.47.66:11677 -> MY.NET.53.204:6129 SYN ******S*
Mar 9 18:53:50 68.185.128.118:4929 -> MY.NET.53.204:80 SYN ******S*
Mar 9 22:05:43 67.170.105.177:4193 -> MY.NET.53.204:4899 SYN ******S*
Mar 9 22:58:41 217.229.150.28:61841 -> MY.NET.53.204:80 SYN ******S*
Mar 10 00:01:02 129.255.144.162:1049 -> MY.NET.53.204:20168 SYN ******S*
Mar 10 03:07:21 163.25.148.155:4121 -> MY.NET.53.204:20168 SYN ******S*
Mar 10 04:18:32 128.134.66.112:4581 -> MY.NET.53.204:80 SYN ******S*
Mar 10 09:21:02 65.66.71.124:220 -> MY.NET.53.204:21 SYN ******S*
Mar 10 11:55:36 4.22.114.47:3373 -> MY.NET.53.204:1257 SYN ******S*
Mar 11 07:45:43 24.31.31.188:1316 -> MY.NET.53.204:80 SYN ******S*
Mar 11 07:45:44 24.31.31.188:1316 -> MY.NET.53.204:80 SYN ******S*
Mar 11 08:33:26 202.40.176.20:3152 -> MY.NET.53.204:4899 SYN ******S*
Mar 11 13:11:30 62.251.180.2:12364 -> MY.NET.53.204:20168 SYN ******S*
Mar 12 00:39:46 213.135.252.86:220 -> MY.NET.53.204:21 SYN ******S*
Mar 12 03:55:42 69.68.84.181:220 -> MY.NET.53.204:21 SYN ******S*
Mar 12 05:26:36 159.149.208.253:1914 -> MY.NET.53.204:4000 SYN ******S*
Mar 12 05:29:44 193.77.153.106:2897 -> MY.NET.53.204:4899 SYN ******S*
Mar 12 07:23:32 218.197.122.51:3617 -> MY.NET.53.204:17300 SYN ******S*
Mar 12 07:23:33 218.197.122.51:3617 -> MY.NET.53.204:17300 SYN ******S*
Mar 12 08:04:40 68.77.58.92:4783 -> MY.NET.53.204:4899 SYN ******S*
Mar 12 18:16:41 213.168.217.148:3823 -> MY.NET.53.204:6129 SYN ******S*
Mar 12 20:15:38 62.243.174.161:3502 -> MY.NET.53.204:6129 SYN ******S*
Mar 12 20:15:40 62.243.174.161:3502 -> MY.NET.53.204:6129 SYN ******S*

Mar 12 19:59:02 64.229.20.163:4810 -> MY.NET.53.204:2745 SYN ******S*
Mar 12 22:30:26 80.81.125.227:45300 -> MY.NET.53.204:6129 SYN ******S*
Mar 12 23:03:37 202.155.56.180:3186 -> MY.NET.53.204:6129 SYN ******S*

SYN packets to MY.NET.69.211:

Mar 8 00:34:45 81.112.172.203:1684 -> MY.NET.69.211:7755 SYN ******S*
Mar 8 00:34:47 81.112.172.203:1684 -> MY.NET.69.211:7755 SYN ******S*
Mar 8 01:36:46 62.23.19.42:3960 -> MY.NET.69.211:20168 SYN ******S*
Mar 8 02:23:53 83.135.68.141:4799 -> MY.NET.69.211:4898 SYN ******S*
Mar 8 04:32:39 81.208.106.64:4907 -> MY.NET.69.211:7100 SYN ******S*
Mar 8 07:41:59 80.48.119.3:4838 -> MY.NET.69.211:20168 SYN ******S*
Mar 8 07:42:01 80.48.119.3:4838 -> MY.NET.69.211:20168 SYN ******S*
Mar 8 21:26:43 212.64.200.62:2358 -> MY.NET.69.211:20168 SYN ******S*
Mar 9 00:54:48 83.121.102.116:3893 -> MY.NET.69.211:20168 SYN ******S*
Mar 9 02:55:42 162.40.123.222:4841 -> MY.NET.69.211:80 SYN ******S*
Mar 9 05:11:51 134.2.78.155:3379 -> MY.NET.69.211:21 SYN ******S*
Mar 9 07:06:42 213.157.171.109:4415 -> MY.NET.69.211:80 SYN ******S*
Mar 9 07:07:22 213.157.171.109:4350 -> MY.NET.69.211:80 SYN ******S*
Mar 9 08:05:08 62.181.222.203:1524 -> MY.NET.69.211:1524 SYN ******S*
Mar 9 08:30:58 211.237.212.40:1843 -> MY.NET.69.211:4489 SYN ******S*
Mar 9 14:06:52 140.112.248.65:4000 -> MY.NET.69.211:4899 SYN ******S*
Mar 9 15:26:43 68.63.203.236:2284 -> MY.NET.69.211:5900 SYN ******S*
Mar 9 15:26:44 68.63.203.236:2284 -> MY.NET.69.211:5900 SYN ******S*
Mar 9 20:10:57 80.19.53.202:4403 -> MY.NET.69.211:6129 SYN ******S*
Mar 9 22:21:48 218.85.85.97:3103 -> MY.NET.69.211:17300 SYN ******S*
Mar 9 22:24:05 217.153.88.103:3256 -> MY.NET.69.211:6129 SYN ******S*
Mar 10 04:19:20 128.134.66.112:4795 -> MY.NET.69.211:80 SYN ******S*
Mar 10 04:19:23 128.134.66.112:4795 -> MY.NET.69.211:80 SYN ******S*
Mar 10 04:22:01 218.26.225.174:4180 -> MY.NET.69.211:4899 SYN ******S*
Mar 10 12:00:49 195.226.52.1:4028 -> MY.NET.69.211:6129 SYN ******S*
Mar 11 00:47:28 24.202.208.210:1927 -> MY.NET.69.211:4899 SYN ******S*
Mar 11 05:49:29 137.204.213.227:2718 -> MY.NET.69.211:8150 SYN ******S*
Mar 11 07:46:25 24.31.31.188:1925 -> MY.NET.69.211:80 SYN ******S*
Mar 12 00:40:14 213.135.252.86:220 -> MY.NET.69.211:21 SYN ******S*
Mar 12 05:29:55 193.77.153.106:1142 -> MY.NET.69.211:4899 SYN ******S*
Mar 12 07:24:18 218.197.122.51:3383 -> MY.NET.69.211:17300 SYN ******S*
Mar 12 07:24:19 218.197.122.51:3383 -> MY.NET.69.211:17300 SYN ******S*
Mar 12 15:05:20 66.191.136.10:42113 -> MY.NET.69.211:21 SYN ******S*
Mar 12 17:25:07 66.188.24.1:4574 -> MY.NET.69.211:20168 SYN ******S*
Mar 12 18:17:27 213.168.217.148:4255 -> MY.NET.69.211:6129 SYN ******S*
Mar 12 18:17:28 213.168.217.148:4255 -> MY.NET.69.211:6129 SYN ******S*
Mar 12 20:02:40 64.229.20.163:4163 -> MY.NET.69.211:2745 SYN ******S*
Mar 12 23:05:31 202.155.56.180:3475 -> MY.NET.69.211:6129 SYN ******S*

Item 3:

Destination IP	# of ports	Destination IP	# of ports
MY.NET.190.0	91	MY.NET.70.107	94
MY.NET.190.1	149	MY.NET.70.108	79
MY.NET.190.102	137	MY.NET.70.109	50
MY.NET.190.202	129	MY.NET.70.114	94
MY.NET.190.203	151	MY.NET.70.118	85
MY.NET.190.92	140	MY.NET.70.128	86
MY.NET.190.93	133	MY.NET.70.133	91
MY.NET.190.95	143	MY.NET.70.135	85
MY.NET.190.97	147	MY.NET.70.139	92
MY.NET.24.20	52	MY.NET.70.148	86
MY.NET.5.13	78	MY.NET.70.156	80

MY.NET.6.15	83	MY.NET.70.157	81
MY.NET.70.0	52	MY.NET.70.162	83
MY.NET.70.1	91	MY.NET.70.163	85
MY.NET.70.105	90	MY.NET.70.164	78

Destination IP	# of ports	Destination IP	# of ports
MY.NET.70.169	99	MY.NET.70.210	90
MY.NET.70.170	101	MY.NET.70.216	93
MY.NET.70.172	92	MY.NET.70.225	94
MY.NET.70.175	98	MY.NET.70.232	106
MY.NET.70.177	94	MY.NET.70.237	85
MY.NET.70.18	88	MY.NET.70.238	76
MY.NET.70.185	82	MY.NET.70.247	104
MY.NET.70.191	102	MY.NET.70.248	93
MY.NET.70.196	92	MY.NET.70.252	105
MY.NET.70.197	93	MY.NET.70.27	75
MY.NET.70.202	83	MY.NET.70.35	90
MY.NET.70.203	108	MY.NET.70.37	69
MY.NET.70.205	94	MY.NET.70.38	77
MY.NET.70.207	101	MY.NET.70.40	86
MY.NET.70.209	87	MY.NET.70.41	89

MY.NET.70.42	92	MY.NET.70.9	87
MY.NET.70.43	88	MY.NET.70.90	86
MY.NET.70.48	93	MY.NET.70.93	96
MY.NET.70.5	90	MY.NET.70.94	90
MY.NET.70.50	84	MY.NET.71.0	52
MY.NET.70.52	88	MY.NET.71.1	95
MY.NET.70.53	96	MY.NET.71.230	85
MY.NET.70.63	103	MY.NET.71.237	79
MY.NET.70.66	91	MY.NET.71.248	82
MY.NET.70.69	98		
MY.NET.70.70	97		
MY.NET.70.72	96		
MY.NET.70.73	95		
MY.NET.70.75	98		
MY.NET.70.88	100		

Item 4:
Sample OOS logs from 64.91.255.232

=+=

03/13-03:45:33.521494 64.91.255.232:34743 -> MY.NET.24.47:21
TCP TTL:51 TOS:0x0 ID:37478 IpLen:20 DgmLen:60 DF

12****S* Seq: 0x16DC93E6 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 45744512 0 NOP WS: 0

=+=
03/13-03:45:33.992122 64.91.255.232:34744 -> MY.NET.24.47:2573
TCP TTL:51 TOS:0x0 ID:8581 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x1694202A Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 45744559 0 NOP WS: 0
=+=
03/13-03:45:38.345487 64.91.255.232:34747 -> MY.NET.24.47:21
TCP TTL:51 TOS:0x0 ID:45730 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x1782B1C0 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 45744994 0 NOP WS: 0

=+=
03/13-03:45:38.802960 64.91.255.232:34748 -> MY.NET.24.47:2575
TCP TTL:51 TOS:0x0 ID:34954 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x171C8B1A Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 45745040 0 NOP WS: 0

Item 5:
Alerts Not considered:
MY.NET.30.4 activity:- I guess the rule that triggered this alert was a custom one. While it is not possible to guess the exact rule and the purpose of the rule, I think the rule triggers this alert if there's a packet bound to this IP. It could be a misconfiguration in the IDS rules or a highly sensitive system. Tim Kroeger [66] has also expressed similar opinion about this alert.
MY.NET.30.3 acitivity:- Same as above
As the above two alerts do not give any specific details, I did not consider these alerts in my top five alerts in section 3.3.
SMB Name Wildcard:- The traffic that triggered all these alerts was originated internally. Since, my main criteria is it focus on external hosts as source addresses, I did not consider this alert in my top five alerts in section 3.3.

Item 6:
Web servers that were targeted:

IP Address	Web Status
MY.NET.111.72	YES
MY.NET.150.101	YES
MY.NET.150.44	YES
MY.NET.29.8	YES
MY.NET.30.3	YES
MY.NET.30.4	YES
MY.NET.5.20	YES
MY.NET.5.25	YES

MY.NET.5.44	YES
MY.NET.5.45	YES
MY.NET.5.46	YES
MY.NET.5.67	YES
MY.NET.5.92	NO
MY.NET.5.95	NO
MY.NET.75.13	NO

Item 7:
Hosts scanned by 63.251.52.75

- MY.NET.110.9
- MY.NET.121.30
- MY.NET.66.31
- MY.NET.80.148

Item 8:
Format of the log files:
Alert files:
Alert files contain all alerts generated by Snort. Depending on the type of alert mode used, the degree of granularity of the information that is logged into the files varies. Refer to the Snort user manual [1] for the different alert modes available. The following is an example record from alert.040308.

03/08-00:00:53.118990 [**] MY.NET.30.3 activity [**] 68.55.250.229:1646 ->MY.NET.30.3:524

The above alert was generated with the alert mode set to "Fast". In the "Fast" mode, the alert contains the timestamp, alert message, source and destination IPs and ports [Snort Users Manual]. Here's a break down of the above example:

Timestamp: *03/08-00:00:53.118990*
Month: 03
Date: 08
Hour: 00
Minute: 00
Second: 53
Millisecond: 118990

Alert: *MY.NET.30.3 activity*

Sourceip:sourceport: *68.55.250.229:1646*

Destinationip:destinationportt: *MY.NET.30.3:524*

Note: *The alert logs do not include the year in which the log was generated. It has to be captured by using a different process. In this case, the year was incorporated in the naming convention.*

OOS files:
The logs in the OOS files were triggered by packets that are unusual. These alerts are very subjective in nature. For example, the following entry was taken from oos_report_040308. This packet was considered unusual by the people who captured this traffic because of the presence of the special bits, ECN and CWR bits. This was logged because snort was told to log packets like this.

03/12-00:05:39.554159 66.225.198.20:58503 -> MY.NET.12.6:25
TCP TTL:52 TOS:0x0 ID:41836 IpLen:20 DgmLen:60 DF

*12****S* Seq: 0x73FB4A17 Ack: 0x0 Win: 0x16D0 TcpLen: 40*
TCP Options (5) => MSS: 1460 SackOK TS: 66840955 0 NOP WS: 0

Please refer to the dissection in the above section for the anatomy of the first line. The following tables show the various fields from the IP and TCP headers in the next three lines.

TCP TTL:52 TOS:0x0 ID:41836 IpLen:20 DgmLen:60 DF

Field	Description	Relative offset
TCP	*Protocol*	*IP[9]*
TTL: 52	*Time To Live*	*IP[8]*
TOS: 0x0	*Type of Service*	*IP[1]*
ID: 41836	*IP ID*	*IP[4:2]*
IpLen: 20	*IP Header Length*	*IP[0] & 0x0f*
DgmLen: 60	*Total Datagram Length*	*IP[2:2]*
DF	*Fragment bit*	*IP[6] & 0x60*

*12****S* Seq: 0x73FB4A17 Ack: 0x0 Win: 0x16D0 TcpLen: 40*
TCP Options (5) => MSS: 1460 SackOK TS: 66840955 0 NOP WS: 0

Field	Description	Relative offset
*12****S**	*TCP Flags*	*TCP[13]*
Seq: 0x73FB4A17	*TCP Seq Number*	*TCP[4:4]*
Ack: 0x0	*TCP Ack Number*	*TCP[8:4]*
Win: 0x16D0	*TCP receive window size*	*TCP[14:2]*
TcpLen: 40	*TCP Header Length*	*TCP[12] & 0xf0*
TCP Option	*TCP Option*	*TCP[20:20]*

Scan files:

The scan files were generated by using the protscan preprocecssor. Refer to the Snort user manual that can be downloaded from [1] for details on how to use this configuration. The following entry was taken from scans.040308

Mar 8 00:06:56 68.54.84.49:57895 -> 130.85.6.7:110 SYN 12****S* RESERVEDBITS

The log shows the timestamp (Mar 8 00:06:56), the source IP:source port (68.54.84.49:57895) and destination IP: destination port (130.85.6.7:110). The log also gives information about the transport protocol involved; in this case, the presence of TCP flags (SYN and the reserved bit) indicates that it is TCP. If the protocol were UDP, the destination port would be followed by the word “UDP”.

Item 9:

03/08-10:00:49.634863 [**] High port 65535 tcp - possible Red Worm - traffic [**] 220.37.240.35:65535 -> MY.NET.53.55:4576
03/08-10:00:50.932070 [**] High port 65535 tcp - possible Red Worm - traffic [**] 220.37.240.35:65535 -> MY.NET.53.55:4576
03/08-10:00:51.646114 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.53.55:4576 -> 220.37.240.35:65535
03/08-10:00:51.868488 [**] High port 65535 tcp - possible Red Worm - traffic [**] 220.37.240.35:65535 -> MY.NET.53.55:4576
03/08-10:00:51.869319 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.53.55:4576 -> 220.37.240.35:65535
03/08-10:00:51.882971 [**] High port 65535 tcp - possible Red Worm - traffic [**] 220.37.240.35:65535 ->

MY.NET.53.55:4576
03/08-10:00:52.048469 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.53.55:4576 -> 220.37.240.35:65535
03/08-10:00:52.600869 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.53.55:4576 -> 220.37.240.35:65535
03/08-10:00:52.953952 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.53.55:4576 -> 220.37.240.35:65535
03/08-10:00:53.302415 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.53.55:4576 -> 220.37.240.35:65535
03/08-10:00:54.015286 [**] High port 65535 tcp - possible Red Worm - traffic [**] 220.37.240.35:65535 -> MY.NET.53.55:4576
03/08-10:00:54.027725 [**] High port 65535 tcp - possible Red Worm - traffic [**] 220.37.240.35:65535 -> MY.NET.53.55:4576
03/08-10:00:54.069051 [**] High port 65535 tcp - possible Red Worm - traffic [**] 220.37.240.35:65535 -> MY.NET.53.55:4576

Item 10:

There were two sets of SYN scans (total of 32 SYNs) coming from 207.44.214.88 on March 8th 2004 - one set starting at 00:58:10 and the other set starting at 00:58:13. The interesting observation is that the source port-destination port combinations were almost same in both the sets. There was just one destination port that appeared only in the first set and one destination port that appeared only in the second set.

Mar 9 00:58:10 207.44.214.88:57491 -> MY.NET.42.5:5104 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57492 -> MY.NET.42.5:5113 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57490 -> MY.NET.42.5:4438 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57493 -> MY.NET.42.5:5262 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57496 -> MY.NET.42.5:6561 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57494 -> MY.NET.42.5:5634 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57495 -> MY.NET.42.5:6552 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57498 -> MY.NET.42.5:7810 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57499 -> MY.NET.42.5:8130 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57497 -> MY.NET.42.5:7464 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57503 -> MY.NET.42.5:9100 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57500 -> MY.NET.42.5:8148 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57504 -> MY.NET.42.5:9186 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57501 -> MY.NET.42.5:8520 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57502 -> MY.NET.42.5:8814 SYN ******S*
Mar 9 00:58:10 207.44.214.88:57506 -> MY.NET.42.5:9578 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57491 -> MY.NET.42.5:5104 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57489 -> MY.NET.42.5:9036 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57492 -> MY.NET.42.5:5113 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57490 -> MY.NET.42.5:4438 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57493 -> MY.NET.42.5:5262 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57496 -> MY.NET.42.5:6561 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57494 -> MY.NET.42.5:5634 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57495 -> MY.NET.42.5:6552 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57498 -> MY.NET.42.5:7810 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57499 -> MY.NET.42.5:8130 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57497 -> MY.NET.42.5:7464 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57503 -> MY.NET.42.5:9100 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57500 -> MY.NET.42.5:8148 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57504 -> MY.NET.42.5:9186 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57502 -> MY.NET.42.5:8814 SYN ******S*
Mar 9 00:58:13 207.44.214.88:57501 -> MY.NET.42.5:8520 SYN ******S*

Likewise, there were two sets of SYN scans (total of 55 SYNs) coming from 66.235.194.217. One set starting at March 8th 17:51:39 and the other set starting at March 9th 01:07:17. Again, the destinations ports that were being scanned were the same in both the sets. There was only one port that was scanned on March 8th that wasn't scanned on March 9th. Interestingly, the

destination ports that were being scanned were almost the same as the ones in the above bullet. Refer to Appendix A for the scans.

```
Mar  8 17:51:39 66.235.194.217:41671 -> MY.NET.42.5:81 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41676 -> MY.NET.42.5:8081 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41672 -> MY.NET.42.5:6588 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41668 -> MY.NET.42.5:8000 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41669 -> MY.NET.42.5:8001 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41677 -> MY.NET.42.5:4914 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41679 -> MY.NET.42.5:7198 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41680 -> MY.NET.42.5:7366 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41682 -> MY.NET.42.5:4438 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41678 -> MY.NET.42.5:6826 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41681 -> MY.NET.42.5:9036 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41684 -> MY.NET.42.5:5113 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41675 -> MY.NET.42.5:8080 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41686 -> MY.NET.42.5:5634 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41685 -> MY.NET.42.5:5262 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41687 -> MY.NET.42.5:6552 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41688 -> MY.NET.42.5:6561 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41689 -> MY.NET.42.5:7464 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41690 -> MY.NET.42.5:7810 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41683 -> MY.NET.42.5:5104 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41692 -> MY.NET.42.5:8148 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41691 -> MY.NET.42.5:8130 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41694 -> MY.NET.42.5:8814 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41695 -> MY.NET.42.5:9100 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41696 -> MY.NET.42.5:9186 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41697 -> MY.NET.42.5:9447 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41693 -> MY.NET.42.5:8520 SYN ******S*
Mar  8 17:51:39 66.235.194.217:41698 -> MY.NET.42.5:9578 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52319 -> MY.NET.42.4:81 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52321 -> MY.NET.42.4:8000 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52322 -> MY.NET.42.4:8001 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52320 -> MY.NET.42.4:6588 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52324 -> MY.NET.42.4:8081 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52323 -> MY.NET.42.4:8080 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52325 -> MY.NET.42.4:4914 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52326 -> MY.NET.42.4:6826 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52327 -> MY.NET.42.4:7198 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52328 -> MY.NET.42.4:7366 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52329 -> MY.NET.42.4:9036 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52330 -> MY.NET.42.4:4438 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52332 -> MY.NET.42.4:5113 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52331 -> MY.NET.42.4:5104 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52333 -> MY.NET.42.4:5262 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52335 -> MY.NET.42.4:6552 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52334 -> MY.NET.42.4:5634 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52336 -> MY.NET.42.4:6561 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52337 -> MY.NET.42.4:7464 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52338 -> MY.NET.42.4:7810 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52339 -> MY.NET.42.4:8130 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52340 -> MY.NET.42.4:8148 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52341 -> MY.NET.42.4:8520 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52342 -> MY.NET.42.4:8814 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52343 -> MY.NET.42.4:9100 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52344 -> MY.NET.42.4:9186 SYN ******S*
Mar  9 01:07:17 66.235.194.217:52346 -> MY.NET.42.4:9578 SYN ******S*
```

Item 11:
OOS log where the packet is coming from MY.NET.12.6 with a TTL value of 255.

```
03/12-02:11:01.018860 MY.NET.12.6:25 -> 208.55.43.103:1660
TCP TTL:255 TOS:0x0 ID:9233 IpLen:20 DgmLen:40
```

12***R** Seq: 0x64146E39 Ack: 0x0 Win: 0x0 TcpLen: 20

Appendix B – Supporting data for Executive Summary

- Internal host participating in IRC/BitTorrent
  MY.NET. 42.3
- Internal host that could be participating in BitTorrent file-sharing.
  MY.NET. 69.226
- Host that could have got compromised due to a vulnerability in a service that is used to synchronize time between hosts
  MY.NET.66.29
- Hosts that could have been infected with SubSeven Trojan
  MY.NET.190.202
  MY.NET.190.102
  MY.NET.190.203
  MY.NET.190.1
  MY.NET.190.93
  MY.NET.190.97
  MY.NET.190.92
- Host that should be examined for VNC
  MY.NET.70.156
- Four hosts should be examined for Dameware – remote administration tool
  MY.NET.27.103
  MY.NET.66.32
  MY.NET.75.6
  MY.NET.84.145
- Web servers that were attacked
  MY.NET.111.72
  MY.NET.150.101
  MY.NET.150.44
  MY.NET.29.8
  MY.NET.30.3
  MY.NET.30.4
  MY.NET.5.20
  MY.NET.5.25
  MY.NET.5.44
  MY.NET.5.45
  MY.NET.5.46
  MY.NET.5.67

© SANS Institute 2000 - 2005 Author retains full rights.

Appendix C

Item 1:

Oracle Table structures:
The following table contained SCAN logs

```
create table newscans (
Year Varchar2(100),
Month varchar2(100),
Day varchar2(100),
Hour varchar2(100),
Minute varchar2(100),
Second varchar2(100),
SourceIP varchar2(200),
SourcePort varchar2(200),
DestinationIP varchar2(200),
DestinationPort varchar2(200),
Protocol varchar2(50),
flags varchar2(150),
descr varchar2(2000),
scan varchar2(4000)
);
```

The following table contained alert logs:

```
create table newalerts (
Year Varchar2(100),
Month varchar2(100),
Day varchar2(100),
Hour varchar2(100),
Minute varchar2(100),
Second varchar2(100),
MilliSecond varchar2(300),
Alert varchar2(2000),
SourceIP varchar2(200),
SourcePort varchar2(200),
DestinationIP varchar2(200),
DestinationPort varchar2(200),
fullalert varchar2(4000));
```

The following table contained OOS logs:

```
create table newoos (
Year Varchar2(100),
Month varchar2(100),
Day varchar2(100),
Hour varchar2(100),
Minute varchar2(100),
```

```
Second varchar2(100),
MilliSecond varchar2(200),
SourceIP varchar2(200),
SourcePort varchar2(200),
DestinationIP varchar2(200),
DestinationPort varchar2(200),
proto varchar2(200),
ttl varchar2(100),
id varchar2(300),
dgmlen varchar2(200),
frag varchar2(200),
flags varchar2(200),
SeqNo varchar2(200),
AckNo varchar2(200),
Win varchar2(200),
oos varchar2(4000));
```

Item 2A:

The following Perl script used to parse scan logs and generate an output file with insert statements.

```
#############################################
## Program: parsescans.pl
## Author: Mohan Chirumamilla
## To parse and generate SQL inserts
## Even though the scripts were almost rewritten, the main idea was from
##  Mr. SaiPrasad Kesavamatham
#############################################

$Year = 2004;
$MYNET = "130.85.";

my %Months = ('Jan'=>'01','Feb'=>'02', 'Mar'=>'03','Apr'=>'04','May'=>'05',
'Jun'=>'06','Jul'=>'07','Aug'=>'08','Sep'=>'09','Oct'=>10,'Nov'=>11,'Dec'=>12);


open(F,"<$ARGV[0]") || die "Cannot open $ARGV[0]\n";
@scans = <F>;
close(F);
open(F,">>$ARGV[1]") || die "Cannot open $ARGV[1]\n";

foreach $aline (@scans)
{
   chomp($aline);
```

```
   #initialize everything to null
   $Month = "";
   $Day = "";
   $Hour = "";
   $Minute = "";
   $Second = "";
   $SourceIP = "";
   $SourcePorrt = "";
   $DestinationIP = "";
   $DestinationPort = "";
   $Protocol = "";
   $descr = "";
   $flags = "";
   $aline =~ s/$MYNET/MY\.NET\./g;

   @Fields = split(/\s+/,$aline);
   $Month = $Months{$Fields[0]};
   $Day = $Fields[1];
   ($Hour,$Minute,$Second) = split(/\:/,$Fields[2]);
   ($SourceIP,$SourcePort) = split(/\:/,$Fields[3]);
   ($DestinationIP,$DestinationPort) = split(/\:/,$Fields[5]);
   if ($Fields[6] eq "UDP")
   {
      $Protocol = "UDP";
   }
   elsif ($Fields[7])
   {
      $Protocol = "TCP";
      $descr = $Fields[6]." ".$Fields[8];
      $flags = $Fields[7];
   }

   $sqlstatement = "insert into newscans values
(\'$Year\',\'$Month\',\'$Day\',\'$Hour\',\'$Minute\',\'$Second\',\'$SourceIP\',\'$SourcePort\',\'
$DestinationIP\',\'$DestinationPort\',\'$Protocol\',\'$flags\',\'$descr\',\'$aline\')\;";
   print F "$sqlstatement\n";


}

close(F);
```

Item 2B:
The following Perl script used to parse alert logs and generate an output file with insert statements.

```
##############################################
## Program: parsealerts.pl
## Author: Mohan Chirumamilla
## To parse and generate SQL inserts for alert logs
## Even though the scripts were almost rewritten, the main idea was from
##  Mr. SaiPrasad Kesavamatham
##############################################

$Year = 2004;
open(F,"<$ARGV[0]") || die "Cannot open $ARGV[0]\n";
@lines = <F>;
close(F);
open(F,">>$ARGV[1]") || die "Cannot open $ARGV[1]\n";


foreach $aline (@lines)
{
   chomp($aline);

   #initialize everything to null
   $Month = "";
   $Day = "";
   $Hour = "";
   $Minute = "";
   $Second = "";
   $SourceIP = "";
   $SourcePorrt = "";
   $DestinationIP = "";
   $DestinationPort = "";
   $Protocol = "";
   $descr = "";
   $flags = "";

   ($TimeStamp, $Alert, $Address) = split (/\[\*\*\]/, $aline);
   ($Month, $Day, $Time) = split (/\/|\-/, $TimeStamp);
```

```
   ($Hour, $Minute, $Second, $MilliSecond) = split (/\:|\./, $Time);
   $Alert =~ s/^\s+//; #cut leading space off alert
   $Alert =~ s/\s+$//; #cut trailing space off alert
   $MilliSecond =~ s/\s+$//;


   if ($Alert && $Alert !~ /spp_portscan/)
   {
      ($SourceAddress, $DestinationAddress) = split (/\->/, $Address);
      ($SourceIP, $SourcePort) = split (/\:/, $SourceAddress);
      $SourcePort =~ s/\s+$//;
      $SourceIP =~ s/^\s+//;
      ($DestinationIP, $DestinationPort) = split (/\:/, $DestinationAddress);
      $DestinationIP =~ s/^\s+//;
#       print F "$Year $Month $Day $Hour $Minute $Second $MilliSecond $Alert
$SourceIP $SourcePort $DestinationIP $DestinationPort\n";
      $sqlstatement = "insert into newalerts values
(\'$Year\',\'$Month\',\'$Day\',\'$Hour\',\'$Minute\',\'$Second\',\'$MilliSecond\',\'$Alert\',\'$S
ourceIP\',\'$SourcePort\',\'$DestinationIP\',\'$DestinationPort\',\'$aline\')\;";
         print F "$sqlstatement\n";
   }
}

close(F);
```

Item 2C:

The following Perl script used to parse OOS logs and generate an output file with insert statements.

```
##############################################
## Program: parseoos.pl
## Author:  Mohan Chirumamilla
## To parse and export oos logs to a database
## Even though the scripts were almost rewritten, the main idea was from
## Mr. SaiPrasad Kesavamatham
##############################################

$Year = "2004";

$push = 0;
open(F,"<$ARGV[0]\n") || die "Cannot open $ARGV[0]\n";
@lines = <F>;
close(F);
open(F,">>$ARGV[1]\n") || die "Cannot open $ARGV[1]\n";
```

```
foreach $line (@lines)
{
   chomp($line);


   if ($line && $line !~ /\=\+/ && $line =~ /(\w\w\s)+/)
   {

         if ($line =~ /->/)
         {
            if($push == 1)
            {
               #print F "$Month $Day $Hour $Second $MilliSecond $SourceIP
$SourcePort $DestinationIP $DestinationPort $proto $ttl $id $dgmlen $frag\n";
               $sqlstatement = "insert into newoos values
(\'$Year\',\'$Month\',\'$Day\',\'$Hour\',\'$Minute\',\'$Second\',\'$MilliSecond\',\'$SourceIP\',
\'$SourcePort\',\'$DestinationIP\',\'$DestinationPort\',\'$proto\',\'$ttl\',\'$id\',\'$dgmlen\',\'$fr
ag\',\'$Flags\',\'$SeqNo\',\'$AckNo\',\'$Win\',\'$line\')\;";
               print F "$sqlstatement\n";
                     $Month = "";
                        $Day = "";
                        $Hour = "";
                        $Minute = "";
                        $Second = "";
                        $SourceIP = "";
                        $SourcePorrt = "";
                        $DestinationIP = "";
                        $DestinationPort = "";
                        $proto = "";
                        $ttl = "";
                        $tos = "";
                        $id = "";
                        $iplen = "";
                        $dgmlen = "";
                        $frag = "";
                        $Flags = "";
                        $SeqNo = "";
                        $AckNo = "";
                           $Win = "";


            }
            @Fields = split(/\s+/, $line);
            ($Month, $Day, $Time) = split (/\/|\-/, @Fields[0]);
```

```
            ($Hour, $Minute, $Second, $MilliSecond) = split (/\:|\./, $Time);
            ($SourceIP, $SourcePort) = split (/\:/, @Fields[1]);
            ($DestinationIP, $DestinationPort) = split (/\:/, @Fields[3]);
            $push = 1;
         }
         elsif ($line =~ /^TCP/ && $line !~ /options/i)
         {
            @misc = split(/\s+/,$line);
            $proto = $misc[0];
            $misc[1] =~ m/TTL:(\d+)/;
            $ttl = $1;
            $misc[2] =~ /TOS:(\S+)/;
            $tos = $1;
            $misc[3] = ~ m/ID:(\d+)/;
            $id = $1;
            $misc[4] =~ m/IpLen:(\d+)/;
            $iplen = $1;
            $misc[5] =~ m/DgmLen:(\d+)/;
            $dgmlen = $1;
            if($misc[6])
            {
               $frag = $misc[6];
            }
         }
         elsif ($line =~ /Seq:/)
         {
            @Options = split(/\s+/, $line);
            $Flags = @Options[0];
            $SeqNo = @Options[2];
            $AckNo = @Options[4];
            $Win = @Options[6];
            # Get the same format as a parsed Alert file for database import


#'null','$Year$Month$Day$Hour$Minute$Second','$MilliSecond','OOS','$SourceIP',

#'$SourcePort','$DestinationIP','$DestinationPort','$Flags','$SeqNo','$AckNo','$Win
            #'
         }
   }

}

close (F);
```