Global Information Assurance Certification Paper

GCIA Intrusion Detection In-Depth
SANS Parliament Square, London
June 20 – 23, 2001
Version 2.9

Christine Chan

Table of Contents

Assignment 1: Network Detects

Submit five networks detect, with analysis.

1.1 Network Detect 1 – DNS Chaos Lookup

1.1.1 Detect

```
16:37:57.360618 adsl-65-65-110-235.dsl.austtx.swbell.net.1737 > my.network.domain: . ack
2391842779 win 32120 <nop,nop,timestamp 118558991 156124585> (DF)
16:37:58.420407 adsl-65-65-110-235.dsl.austtx.swbell.net.1858 > my.network.domain:
32600+ TXT CHAOS)? VERSION.BIND. (30)
```

1.1.2 Source of Detect

My network

1.1.3 Detect Generation

Detect was generated by TCPDump v3.6.

1.1.4 Probability Source Address was Spoofed

The beginning of the conversation from the TCPDump log is missing, but because the first line of the log is an ACK packet, and that the protocol being used is TCP, we can assume that we are witnessing the end of the three-way handshake. Therefore since a TCP connection has been established, the source address is highly unlikely to be spoofed.

1.1.5 Description of Attack

The TCPDump log was triggered by the following alert from PortSentry:

```
Aug  2 16:37:56 fw portsentry[6659]: attackalert: Connect from host:
65.65.110.235/65.65.110.235 to UDP port: 53
```

```
Aug  2 16:37:57 fw kernel: Portsentry: dropping: IN=ppp0 OUT= MAC= SRC=65.65.110.235
DST=my.network LEN=52 TOS=0x00 PREC=0x00 TTL=43 ID=30586 DF PROTO=TCP SPT=1737 DPT=53
WINDOW=32120 RES=0x00 ACK URGP=0
Aug  2 16:37:58 fw kernel: Portsentry: dropping: IN=ppp0 OUT= MAC= SRC=65.65.110.235
DST=my.network LEN=58 TOS=0x00 PREC=0x00 TTL=43 ID=30606 PROTO=UDP SPT=1858 DPT=53 LEN=38
```

TCPDump only logged the traffic after the alert has been generated (as instructed in a shell script to be executed on alert being triggered), so the beginning of the conversation is missing from the log.

Looking at the PortSentry alert, it seems that host adsl-65-65-110-235.dsl.austtx.swbell.net triggered the alert by attempting a connection to UDP port 53 (at 4.37:56pm), maybe to verify there is an active DNS server listening on port 53. Then immediately a TCP connection is attempted (at 4:37:57pm), followed by another UDP one (at 4:37:58pm).

From the TCPDump log, it can be seen that at the time of the second UDP connect, the version of BIND is queried, as shown in the TCPDump log:

```
16:37:58.420407 adsl-65-65-110-235.dsl.austtx.swbell.net.1858 > my.network.domain:
32600+ TXT CHAOS)? VERSION.BIND. (30)
```

The BIND DNS server has a feature where its database contains a CHAOS/TXT record with the name "VERSION.BIND". If somebody queries this record, the version of the BIND software will be returned. This event is triggered whenever anybody does such a lookup. This is not an attack itself, but a simple reconnaissance scan. However, if the returned version number is something like "4.9.6-REL" or "8.2.1", then it indicates that one of the known version of BIND is running, and that it can be broken into with a buffer overflow exploit, which is what the attacker may have been looking to find out.

1.1.6 Attack Mechanism

The BIND version can be queried using the following commands:

dig @target.network version.bind chaos txt

or

my.server> /usr/sbin/nslookup
>set q=txt
>set class=chaos
>server target.dns.server
Default Server: target.dns.server
Address: xx.xx.xx.2

>version.bind

1.1.7 Correlation

There are some examples of DNS Chaos Lookup on various mailing lists, such as Linuxarkivet (http://www.linuxarkivet.nu/), Neohapsis (http://www.neohapsis.com/), The Mail Acrhive, and websites such as http://www.incidents.org/.

An example of this type of activity was extracted from http://www.mail-archive.com/comp-protocols-dns-bind@moderators.isc.org/msg00378.html:

In my newly set-up DNS server which is the SOA for some public IN domains, I found the following line in the logging of bind (named): XX+/212.68.193.196/version.bind/TXT/CHAOS. Has someone any idea why this query was sent to my DNS server? Should I be worried about it?

Vik Heyndrickx posted the above on 27 Jan 2001.

1.1.8 Evidence of Active Targeting

Because the attack is a reconnaissance scan, this shows that the attacker had chosen the target intentionally, hence this is active targeting.

1.1.9 Severity

Criticality – DNS Server (5)
Lethality – Reconnaissance scan (2)
System Countermeasures – Modern OS, some patches missing (4)
Network Countermeasures – Validated restrictive firewall (5)

Severity = (Criticality + Lethality) – (System + Network Countermeasures)
= (5+2) – (4+5)
= -2

1.1.10 Defensive Recommendation

Try not to give any information away by using fake BIND version information in the banners or preventing BIND from showing the version. The AUSCERT recommended fix involve adding a special "bind" zone definition to named.conf and adding a master zone file for the "bind" zone.

Also, the following rule can be added to Snort to help catch these attacks in the future:

alert udp $EXTERNAL any -> $INTERNAL 53 (msg: "IDS278 - SCAN - named Version Probe"; content: "|07|version|04|bind|00 0010 0003|"; nocase; offset: 12; depth: 18;)

Above Snort rule was extracted from http://archives.neohapsis.com/archives/snort/2001-02/0192.html

1.1.11 Multiple Choice Test Question

Q. When would DNS traffic use TCP?

1. DNS will choose to use UDP and TCP randomly
2. For zone transfer and when the response packet size exceeds a certain size
3. DNS only uses TCP
4. DNS never uses TCP, only UDP

A. The answer is 2: DNS will use TCP for zone transfers and when the response is too large for UDP.

1.2 Network Detect 2 - RST/ACK Stealth Scan

1.2.1 Detect

```
02:23:31.406132 192.31.21.163.6635 > x.x.128.70.6635: S 1329421741:1329421741(0) win 25446
02:23:31.528424 192.31.21.163.6635 > x.x.133.100.6635: S 1329421741:1329421741(0) win 25446
02:23:31.529020 x.x.133.100.6635 > 192.31.21.163.6635: R 0:0(0) ack 1329421742 win 0
02:23:31.552094 192.31.21.163.6635 > x.x.133.62.6635: S 1329421741:1329421741(0) win 25446
02:23:31.553319 x.x.133.62.6635 > 192.31.21.163.6635: R 0:0(0) ack 1329421742 win 0
02:23:31.555878 192.31.21.163.6635 > x.x.133.115.6635: S 1329421741:1329421741(0) win 25446
02:23:31.558264 x.x.133.115.6635 > 192.31.21.163.6635: R 0:0(0) ack 1329421742 win 0
02:23:31.570929 192.31.21.163.6635 > x.x.134.90.6635: S 1329421741:1329421741(0) win 25446
02:23:31.576215 192.31.21.163.6635 > x.x.134.93.6635: S 1329421741:1329421741(0) win 25446
02:23:31.578252 x.x.134.93.6635 > 192.31.21.163.6635: R 0:0(0) ack 1329421742 win 0
02:23:32.197527 192.31.21.163.6635 > x.x.128.246.6635: S 1329421741:1329421741(0) win 25446
```

1.2.2 Source of Detect

Detect was taken from http://www.incidents.org/archives/intrusions/msg01536.html

1.2.3 Detect Generation

Detect was generated by TCPDump.

1.2.4 Probability Source Address was Spoofed

From first glance, it seems the source IP is legitimate since this is a TCP communication and using a spoofed IP will not allow the real source to see any return packets. But this activity seems to be a stealth scan, which would mean that the source IP could be spoofed to avoid exposing the real identity of the attacker.

1.2.5 Description of Attack

Studying the destination IPs, it seems this is some kind of reconnaissance activity. The source IP is sending a SYN packet to different destinations, all with the destination port of 6635. What is strange is that the source port is also 6635 and this is static for each connection, which is not normal TCP/IP behavior.

Also the sequence number is the same for each connection and the window size is rather large. This all leads me to conclude that the packets are crafted. But why?

The port 6635 are not defined and do not seem to be used by any trojans. As started before, this is most probably a scan to detect live hosts, since if the host is alive and the port is opened, then on receiving a SYN packet, the host will respond with a SYN+ACK. However, if the host is alive and the port is closed, then on receipt of a SYN, the host will respond with a RST+ACK.

If a host is not alive, then an ICMP error message will be sent back to the source IP. From the trace, there is no evidence of any ICMP messages, but this maybe that the destination hosts are located behind a firewall, which prohibits outbound ICMP messages.

From the scan, an attacker can effectively map out the victim's network.

Because of the short interval between each connection, this scan may have been conducted by an automated script.

1.2.6 Attack Mechanism

The above trace can be attributed to a TCP SYN Scan (or Half-Open Scanning):

A SYN Scanner, which did not establish a complete TCP connection, is used to perform these scans. These kinds of port scanners remain undetectable by only sending the first single TCP Packet containing the SYN flag and establishing a half TCP Connection.

Steps:-

1. The SYN Scanner sends the first TCP packet containing the SYN flag (which in turn contains the port number) to the remote host
2. The remote system replies with either a SYN/ACK or a RST/ACK
3. If the client receives a SYN/ACK from the server, then it means that the port is in listening state. However, if the client receives a RST/ACK then it means that the port is not listening or in other words there is no service running on that particular remote port
4. When the SYN Scanner receives one of the above responses, it knows whether the respective port is open or not and whether a daemon is ready listening for connections

TCP SYN Scans or Half-Open Scanning is a stealth method of port scanning, because a full TCP three-way handshake does not take place. Thus, they are less detectable as compared to traditional TCP connect scans.

1.2.7 Correlation

Matt Franz described a similar activity to the one in the above trace in http://lists.gnac.net/pipermail/firewalls/1998-May/040650.html.

The trace below was extracted from the message posted by Matt Franz

```
SYN SCAN
PORT 21 (Open)
10:22:45.030552 192.168.0.2.49724 > 192.168.0.3.21: S 2421827136:2421827136(0)
10:22:45.030552 192.168.0.3.21 > 192.168.0.2.49724: S 4046313668:4046313668(0) ack
2421827137
10:22:45.030552 192.168.0.2.49724 > 192.168.0.3.21: R 2421827137:2421827137(0)
PORT 22 (Closed)
10:22:45.050552 192.168.0.2.49724 > 192.168.0.3.22: S 2418821749:2418821749(0)
10:22:45.050552 192.168.0.3.22 > 192.168.0.2.49724: R 0:0(0) ack 2418821750
```

1.2.8 Evidence of Active Targeting

Because this seems to be hosts scan using RST+ACK, this shows the attacker is interested in the victim's network, hence this implies that this trace is the result of active targeting.

1.2.9 Severity

Criticality – Any systems (5)
Lethality – Reconnaissance scan (2)
System Countermeasures – Modern OS, some patches missing (4)
Network Countermeasures – Validated restrictive firewall (5)

Severity = (Criticality + Lethality) – (System + Network Countermeasures)
= (5+2) – (4+5)
= -2

1.2.10 Defensive Recommendation

This scan uses a RST+ACK to detect wherever a port is listening or not, or wherever a host is alive or not. To prevent this type of stealth scan, use firewalls to control access to ports on internal hosts. Any ports that should not be opened must be closed, the firewalls must not allow any access to these ports.

The ports that need to be opened must have access to them controlled at the firewalls and routers.

By controlling port access, this will minimize the probability of random scans from attackers, wherever it is direct or indirect.

1.2.11 Multiple Choice Test Question

Q. If the following pattern was seen in one of the network logs from your network, what could this mean?

```
02:23:31.529020 x.x.x.1.6635 > my.net.x.10.7000: R 0:0(0) ack 1329421742 win 0
02:23:31.553319 x.x.x.1.6636 > my.net.x.10.7001: R 0:0(0) ack 1329421743 win 0
02:23:31.558264 x.x.x.1.6637 > my.net.x.10.7002: R 0:0(0) ack 1329421744 win 0
02:23:31.578252 x.x.x.1.6638 > my.net.x.10.7003: R 0:0(0) ack 1329421745 win 0
```

Please note that this trace is not real

1. This is normal activity, seeing uninitiated RST+ACK packets should not raise any cause for concerns
2. The host my.net.x.10 is the victim of source IP spoofing
3. The host x.x.x.1 are trying to scan the host my.net.x.10
4. This is the activity of a denial-of-service attack

A. The answer is 2: An attacker has sent SYN packets to the destination hosts, using my.net.x.10 as the spoofed IP. The log shows the destination host responding to my.net.x.10 with the correct RST+ACK for closed ports. This can be an indirect stealth port scan.

   By using a spoofed IP, the attacker can hide its real identity from detection. To perform the scan, the attacker first pings the spoofed IP to obtain the IP ID. Then it will send a crafted packet with the spoofed IP as the source IP to the victim host, who will reply and this response can be observed by the attacker, who will then know what ports are open on the victim host without giving away its identity.

1.3 Network Detect 3 – Broadcast Destination Addresses

1.3.1 Detect

```
TCP scan underway from 64.4.58.204: 04/04/01 15:27 GMT:
From firewall logs: (times are Eastern Daylight Savings Time)
Apr 4 06:34:06 Deny TCP 64.4.58.204:56974 MY.NET.71.0:6606
Apr 4 06:39:36 Deny TCP 64.4.58.204:33642 MY.NET.3.0:13037
Apr 4 07:11:07 Deny TCP 64.4.58.204:7162 MY.NET.71.0:10780
Apr 4 07:36:07 Deny TCP 64.4.58.204:31099 MY.NET.2.0:16496
Apr 4 07:42:37 Deny TCP 64.4.58.204:60857 MY.NET.71.0:1168
Apr 4 08:51:08 Deny TCP 64.4.58.204:27153 MY.NET.3.0:21971
Apr 4 10:09:10 Deny TCP 64.4.58.204:51486 MY.NET.3.0:53326
Apr 4 11:25:41 Deny TCP 64.4.58.204:17920 MY.NET.3.0:8452
```

1.3.2 Source of Detect

Detect was taken from http://www.sans.org/y2k/040601.htm

1.3.3 Detect Generation

This is some log from some firewall.

1.3.4 Probability Source Address was Spoofed

The source address 64.4.58.204 is registered to MS Hotmail in US (www.hotmail.com), it seems unlikely that a host on the Hotmail network will conduct any malicious activity, hence it is most likely that this source address is spoofed.

But since Dec 2000, Microsoft IIS has been vulnerable to the LPRng vulnerability (http://www.kb.cert.org/vuls/id/382365), and if the servers have not been patched, then it may

be likely that the host 64.4.58.204 had been compromised and the source address is not spoofed.

This is based on the assumption that Hotmail uses IIS as their web servers.

1.3.5 Description of Attack

Using x.x.x.0 as a destination address is not normal, which is what probably triggered the firewall to deny and log these packets.

The times between each attempt are quite far apart and at irregular intervals, hence these attempts are probably conducted manually. Looking at the source ports, they are not sequential and this can be because the host is initiating many connections. One reason for this (apart from the fact that the host is a server for Hotmail) is that it may also be scanning other sites, and interleaving its activities to avoid detection.

The .0 host part of an IP address denotes the network or the broadcast address on older systems. New systems will ignore and drop the packets.

Why would someone send packets to this address? One possibility springs to mind, which is OS fingerprinting. Since only old systems will understand and recognize the x.x.x.0 address, then if a response is received from a host, this will effectively tell the attacker the version of the system running on the host.

1.3.6 Attack Mechanism

From an attacking host, just initiate a connection to the x.x.x.0 address (telnet, ftp, etc or a TCP ping).

1.3.7 Correlation

A similar trace can be found at http://www.sans.org/y2k/031301.htm, below is an extract from it:

```
security.214:Feb 13 22:35:31 Deny TCP 140.118.107.12:5818 my.net.3.0:98

security.214:Feb 13 22:36:26 Deny TCP 140.118.107.12:4203 my.net.13.0:98
security.214:Feb 13 22:36:26 Deny TCP 140.118.107.12:4203 my.net.13.0:98

security.214:Feb 13 22:39:56 Deny TCP 140.118.107.12:22453 my.net.71.0:98
security.214:Feb 13 22:39:56 Deny TCP 140.118.107.12:22453 my.net.71.0:98


security.0306-1600:Mar 6 11:30:42 Deny ICMP:8.0 204.19.37.24 my.net.13.0
security.0306-1600:Mar 6 11:30:42 Deny ICMP:8.0 204.19.37.24 my.net.13.255
security.0306-1600:Mar 6 11:30:42 Deny ICMP:8.0 204.19.37.24 my.net.71.0
security.0306-1600:Mar 6 11:30:42 Deny ICMP:8.0 204.19.37.24 my.net.71.255
```

The ICMP traffic in the above trace may be attributed to Smurf activities.

1.3.8 Evidence of Active Targeting

The same source address targeted some of the network/broadcast address on my.net.x.x several times, each with a different destination port. Because of this pattern, this leads me to believe that this is active targeting, since if it were random, then different unique destinations would be in the trace, not repeated addresses.

1.3.9 Severity

Criticality – Any systems (5)
Lethality – Attack is unlikely to succeed (1)
System Countermeasures – Modern OS, some patches missing (4)
Network Countermeasures – Validated restrictive firewall (5)

Severity = (Criticality + Lethality) – (System + Network Countermeasures)
= (5+1) – (4+5)
= -3

1.3.10 Defensive Recommendation

Broadcast addresses must not be forwarded by any routers or routing hosts, blocking this at the routers will avoid situations where packets aimed for broadcast addresses will enter the network.

1.3.11 Multiple Choice Test Question

Q. How does the Smurf attack work?

1. Is an email virus – it arrives in the victim's inbox as a attachment, once the victim executes the attachment, the virus will execute its payload and make various changes to the system. Then it will emails itself out to everyone in the victim's address book
2. Is a Internet worm – it probes random IIS servers for exploits, once a successful connect as been made, the worm proceeds to infect the computer (by exploiting a specific vulnerability) and then use the computer as a base to infect other systems
3. Is a Distributed Denial of Service attack (DDoS) – the attacker uses a spoofed source address (which is the victim's address) and sends an ICMP request to a broadcast address. The router will send this to all hosts on the network and each host will send a ICMP reply to the victim's machine
4. Is a Denial of Service attack – by sending large ICMP request packets to victims from remote machines, causing a buffer overflow on the systems and leading to system panic

A. The answer is 3: Smurf attacks are a brute-force attack targeted at a feature in the IP specification known as direct broadcast addressing. A Smurf hacker flood the routers with ICMP echo request packets (pings) and since the destination IP address of each packet is the broadcast address of the network, the router will broadcast the ICMP

echo request packet to all hosts on the network. If there are numerous hosts, this will create a large amount of ICMP echo request and response traffic.

If the source IP address of the ICMP echo request packet is spoofed, then the resulting ICMP traffic will not only clog up the network--the "intermediary" network--but will also congest the network of the spoofed source IP address--known as the "victim" network.

1.4 Network Detect 4 – Code Red II

1.4.1 Detect

```
[**] IDS552/web-iis_IIS ISAPI Overflow ida [**]
08/06-10:13:55.165739 210.209.11.12:4715 -> my.network:80
TCP TTL:109 TOS:0x0 ID:43198 IpLen:20 DgmLen:576
***A**** Seq: 0xCB1D4D1C  Ack: 0xE4E9D43F  Win: 0x4470  TcpLen: 20
47 45 54 20 2F 64 65 66 61 75 6C 74 2E 69 64 61  GET /default.ida
3F 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  ?XXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58  XXXXXXXXXXXXXXXX
58 25 75 39 30 39 30 25 75 36 38 35 38 25 75 63  X%u9090%u6858%uc
62 64 33 25 75 37 38 30 31 25 75 39 30 39 30 25  bd3%u7801%u9090%
75 36 38 35 38 25 75 63 62 64 33 25 75 37 38 30  u6858%ucbd3%u780
31 25 75 39 30 39 30 25 75 36 38 35 38 25 75 63  1%u9090%u6858%uc
62 64 33 25 75 37 38 30 31 25 75 39 30 39 30 25  bd3%u7801%u9090%
75 39 30 39 30 25 75 38 31 39 30 25 75 30 30 63  u9090%u8190%u00c
33 25 75 30 30 30 33 25 75 38 62 30 30 25 75 35  3%u0003%u8b00%u5
33 31 62 25 75 35 33 66 66 25 75 30 30 37 38 25  31b%u53ff%u0078%
75 30 30 30 30 25 75 30 30 3D 61 20 20 48 54 54  u0000%u00=a  HTT
50 2F 31 2E 30 0D 0A 43 6F 6E 74 65 6E 74 2D 74  P/1.0..Content-t
79 70 65 3A 20 74 65 78 74 2F 78 6D 6C 0A 43 6F  ype: text/xml.Co
6E 74 65 6E 74 2D 6C 65 6E 67 74 68 3A 20 33 33  ntent-length: 33
37 39 20 0D 0A 0D 0A C8 C8 01 00 60 E8 03 00 00  79 ........`....
00 CC EB FE 64 67 FF 36 00 00 64 67 89 26 00 00  ....dg.6..dg.&..
E8 DF 02 00 00 68 04 01 00 00 8D 85 5C FE FF FF  .....h......\...
50 FF 55 9C 8D 85 5C FE FF FF 50 FF 55 98 8B 40  P.U...\...P.U..@
10 8B 08 89 8D 58 FE FF FF FF 55 E4 3D 04 04 00  .....X....U.=...
00 0F 94 C1 3D 04 08 00 00 0F 94 C5 0A CD 0F B6  ....=...........
C9 89 8D 54 FE FF FF 8B                          ...T....
```

1.4.2 Source of Detect

My network

1.4.3 Detect Generation

Detect was generated by Snort v1.7.

1.4.4 Probability Source Address was Spoofed

From this packet alone, it is not really possible to determine whether the source IP was spoofed or not. But because this is a TCP communication, using a spoofed IP would not be of much benefit to the attacker, since they will not receive any response.

1.4.5 Description of Attack

This trace was triggered by the 'web-iis_IIS ISAPI Overflow ida' rule, which was implemented to catch the 'Buffer Overflow In IIS Indexing Service DLL' (http://www.cert.org/advisories/CA-2001-13.html) exploit. This exploit takes advantage of a buffer overflow in one of the ISAPI extensions installed with most versions of IIS, which can lead to attackers gaining control over the server.

From initial observation of the trace, the pattern looks very similar to the one left by the Code Red Worm (CRv1 and CRv2 – http://www.cert.org/advisories/CA-2001-19.html). Below is an example of CRv2 activity (extracted from IIS logs):

```
2001-08-01 17:06:02 209.27.247.5 - GET /default.ida
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u90
90%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u
9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a 200 171 4039 94
80 HTTP/1.0 - - -
```

The worm tries to connect to port 80 on randomly chosen targets, upon a successful connection, the attacking host sends a crafted HTTP GET request to the target, which attempts to exploit the buffer overflow in the Indexing Service. This can be seen in the extracted IIS log (having this string in the log does not mean the server have been compromised, only that an infection was attempted), by the long string of Ns in the GET request.

As can be seen, the detection from Snort strikes a big resemblance to the above log extract. To further show the similarities, here are the attack signatures of both CRv2 and the new worm:

CRv2

```
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3
%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b0
0%u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0
```

The new worm sends a very similar header

```
/default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd3
%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b0
0%u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0
```

The difference is that the latter uses X instead of N as its filler character. Because of this resemblance, I was at first be inclined to dismiss this as a new variant of the Code Red Worm.

But, after several references to people observing the same pattern on their servers, the new worm is in fact not a variant of the Code Red Worm. This worm takes advantage of the same vulnerability as the Code Red Worm, but the code is different. The author of the new worm has embedded the string "CodeRedII" inside the code, hence leading to conclusions that this is in response to, or inspired by, the original Code Red Worm.

Code Red II will only exploit Windows 2000 web servers because it overwrites EIP with a jmp that is only correct under Windows 2000. Under other Windows OS, that offset is different so, the process will simply crash instead of allowing the worm to infect the system and spread.

This worm has a more malicious payload, a backdoor access method. It leaves a copy of cmd.exe, named root.exe, in a location accessible to the web server. If successful, this will allow any attacker (not just the worm author) access to the victim's web server at a later date.

1.4.6 Attack Mechanism

Code Red II generates a random IP for their next target and then proceeds to infect it. The process for propagation is:

Extracted from Eeye's analysis of Code Red II
*(*http://www.eeye.com/html/Research/Advisories/AL20010804.html*)*

A. Setup local IP_STORAGE variable. This is used for worm propagation functionality and to make sure not to re-infect the local system
B. Sleep for 64h milliseconds
C. Get local system time. The worm checks to see if it the year is less than 2002 or if the month is less than 10. If the date is beyond either of those, then the worm reboots the local system. That basically limits the worm to 10/01 for its spreading (in a perfect world.)
D. Setup SockAddr_in. This will reference the GET_IP section
E. Setup Socket: This performs a Socket(), stores the handle, then makes it a non-blocking socket (this is important for speed dealing with connect() calls)
F. Connect to the remote host, if it returns a connect right away, go to Step H
G. Do a select to get the handle. If no handle is returned, then go to Step K
H. Set socket to Blocking. This is so select isn't required after the connect
I. Send a copy of the worm
J. Do a recv. This is not actually used anywhere
K. Close the socket and loop to Step A

1.4.7 Correlation

There are many examples of Code Red Worm activity (of all versions) on the Web (namely http://www.sans.org) and from mailing lists such as the Incident mailing list from SecurityFocus (http://www.securityfocus.com/templates/archive.pike?list=75).

1.4.8 Evidence of Active Targeting

The Code Red Worm generates a random list of IP to target from each compromised machine, hence each IP has a 'random' chance of being hit. Hence Code Red II infects its victims at random and does not perform active targeting.

1.4.9 Severity

Criticality – Web Server (5)
Lethality – Backdoor (5)
System Countermeasures – Modern OS, some patches missing (4)
Network Countermeasures – Validated restrictive firewall (5)

Severity = (Criticality + Lethality) – (System + Network Countermeasures)
= (5+5) – (4+5)
= 1

1.4.10 Defensive Recommendation

It is recommended that all relevant patches be applied immediately to IIS on Windows NT/2000. In addition, SecurityFocus analysts recommend that as much of the following hardening/checklist document by implemented as possible:

IIS4:
http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/columns/questions/iischeck.asp

IIS5:
http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/iis/deploy/depovg/securiis.asp

There is also an IIS5 Hotfix Checking Tool to check for patches that have not been installed. You can get it from Microsoft Technet at:
http://www.microsoft.com/Downloads/Release.asp?ReleaseID=24168

1.4.11 Multiple Choice Test Question

Q. What is a backdoor?

5. A door that leads into the garden
6. A door on the desktop/tower, that opens up the casing to expose the internal components of a computer
7. A entry point on a system that the owner purposefully created, to allow everyone access to their systems

8. A entry point left on a system, such that it enables users to gain entry to the system without the knowledge of the owner

A. The answer is 4: By definition a back door is a mechanism for circumventing or disabling system security. Originally, back doors were presumed to be 'justified' because they offered system access to technicians and other administrators, and were often deliberately included with applications. But recently, searching for (and finding) back doors are a common and fairly effective attack technique used by attackers, and more often, attackers will write or utilizes specific codes (trojans) that will compromise a system and leave a means of entry to the system (i.e. open a new port to listen for connections).

1.5 Network Detect 5 – Port 3879 Scan

1.5.1 Detect

```
Source: 146.96.242.15 (146.96.242.15)
Destination port: 3879 () SYN flags: ******S* Count: 79
Times in PDT (UTC -0700)
Apr 3 00:49:14 146.96.242.15:4954 -> 142.90.100.4:3879 SYN ******S*
Apr 3 00:49:39 146.96.242.15:3794 -> 142.90.100.51:3879 SYN ******S*
etc.
Source: 195.223.184.81 (195.223.184.81)
Destination port: 3879 () SYN flags: ******S* Count: 56
Times in PDT (UTC -0700)
Apr 2 09:02:11 195.223.184.81:1059 -> 142.90.100.2:3879 SYN ******S*
Apr 2 08:57:28 195.223.184.81:2181 -> 142.90.100.1:3879 SYN ******S*
etc.
Source: 203.86.3.94 (203.86.3.94)
Destination port: 3879 () SYN flags: ******S* Count: 50
Times in PDT (UTC -0700)
Apr 3 00:02:07 203.86.3.94:2675 -> 142.90.100.54:3879 SYN ******S*
Apr 3 00:06:16 203.86.3.94:3828 -> 142.90.107.6:3879 SYN ******S*
etc.
Source: 210.161.41.56 (210.161.41.56)
Destination port: 3879 () SYN flags: ******S* Count: 47
Times in PDT (UTC -0700)
Apr 2 20:02:21 210.161.41.56:1937 -> 142.90.100.1:3879 SYN ******S*
Apr 2 20:01:19 210.161.41.56:4794 -> 142.90.100.2:3879 SYN ******S*
etc.
Source: adsl-63-195-2-66.dsl.chic01.pacbell.net (63.195.2.66)
Destination port: 3879 () SYN flags: ******S* Count: 44
Times in PDT (UTC -0700)
Apr 2 23:56:05 63.195.2.66:1425 -> 142.90.100.1:3879 SYN ******S*
etc
```

1.5.2 Source of Detect

Detect taken from http://www.sans.org/y2k/040401.htm

1.5.3 Detect Generation

Detect was generated from Snort and logged by Snort's Portscan.

1.5.4 Probability Source Address was Spoofed

It is not likely that the source addresses are spoofed since the trace does not look like a 'denial of service' attack, and using spoofed addresses for information gathering would not benefit the attackers.

1.5.5 Description of Attack

This alert was triggered because this trace is a port scan. Snort's Portscan will register traffic as port scans if a certain amount of it is sent to the network within a certain time limit.

From the trace, various source addresses sends a SYN packet to port 3879 on hosts on the 142.90.100.x network.

There are too many source addresses and only one port targeted for this to be an information gathering port scan. The time for each scan from each source address are quite close together, indicating that this is the result of some automated script. Having several hosts running the same automated script and targeting the same port number brings me to conclude that this is the activity of a worm.

Doing some research on the Web, port 3879 seems to be a standard for Linux exploits. Extracted from http://maclux-rz.uibk.ac.at/~maillists/focus-ms/msg00669.shtml, Daniel Martin made this comment on 4 Apr 2001:

> *Check message <01040408575000.03578@localhost.localdomain> no the incidents list; it has captured packets that show an exploit attempt against port 515. The exploit, if successful, binds a shell to port 3879. That is, this port has no significance beyond this particular worm, but the connection to 3879 is how the worm checks that the LPR exploit was successful and presumably delivers its payload.*

The trace above seems to match the comments made by Daniel Martin, but further analysis of logs from the same day will need to be made, to verify that port 515 on hosts on the target's network has also been scanned and that this trace is related to the LPRng exploit.

1.5.6 Attack Mechanism

The following assumes that the above trace was created by worms exploiting the LPRng Buffer Overflow vulnerability:

The attacking host performs a fast scan of TCP port 515 on the target network, any address that responds then gets hit by hundreds of exploit attempts and connection attempts to 3879.

The exploit used was a LPRng Buffer Overflow to a Unix machine (possibly RedHat 7.0). The overflow spawns a shell on port 3879 that is used to install the following onto the compromised system:

- /usr/bin/atm - a trojaned SSHD 1.2.27 binary which listens on TCP port 23132. The SSH keys and conf file are placed in the directory /usr/man/man2/.man8
- /bin/netstat is replaced
- /bin/dmesg is replaced
- Located in /usr/man/man8/.man8/rtk are the lpdscan and lpdx binaries; lpdx is the actual LPD exploit.

An install shell script could be used for the exploit.

1.5.7 Correlation

There are many examples of this type of activity, one such trace can be seen at http://www.sans.org/y2k/012201.htm (LPRng trace), I've included a extract from the log below:

```
The attacker starts a connection to port 515 (printer):

18:52:08.574822 24.147.188.237.4796 > 216.164.26.224.515: S
551120663:551120663(0) win 32120 (DF) (ttl 48, id 63337)
18:52:08.574962 216.164.26.224.515 > 24.147.188.237.4796: S
1939527328:1939527328(0) ack 551120664 win 32120 (DF) (ttl 64, id 780)

18:52:08.625678 24.147.188.237.4796 > 216.164.26.224.515: . 1:1(0)
ack 1 win 32120 (DF) (ttl 48, id 63338)


The attacker tries to start a connection to port 3879:

18:52:08.730624 24.147.188.237.4797 > 216.164.26.224.3879: S
546052661:546052661(0) win 32120 (DF) (ttl 48, id 63341)
```

The mailing lists on SecurityFocus also have many examples of traces concerning the LPRng exploit and port 3879.

1.5.8 Evidence of Active Targeting

If this trace is the activity of a worm, then each target will be randomly chosen from the attacking host. The fact that several hosts on the same network was targeted by various source addresses is not enough to conclude that this is active targeting, it can just be that the 'random seed' in the worm is not so random afterall.

1.5.9 Severity

Criticality – Unix systems (3)
Lethality – Exploit can cause various effects (4)
System Countermeasures – Modern OS, some patches missing (4)
Network Countermeasures – Validated restrictive firewall (5)

Severity = (Criticality + Lethality) – (System + Network Countermeasures)
= (3+4) – (4+5)
= -2

1.5.10 Defensive Recommendation

LPR is the print service on Unix machines, if this service is not needed then it should be disabled. The servers where such services are required must be patched with the latest updates.

1.5.11 Multiple Choice Test Question

Q. What three known vulnerabilities do the Ramen Worm exploits?

1. Wu-ftpd, LPRng, rpc.statd
2. BIND, NTP, telnetd
3. Apache, Samba, man
4. Sendmail, LPRng, BIND

A. The answer is 1: Ramen is a Linux-based Internet worm that targets Red Hat 6.2 and Red Hat 7.0, using three security breaches associated with the packages nfs-utils, wu-ftpd, and LPRng.

   Once the worm detects the vulnerabilities, it exploits them, copies itself onto the server, and takes over root access rights.

   The three security breaches sought out by the Ramen worm are as follows:

   a. Wu-ftpd: Buffer overrun; due to improper bounds checking, SITE EXEC may enable remote root execution, without having any local user account required
   b. nfs-utils: Flaws in the rpc.statd daemon can lead to remote root break in
   c. LPRng: Vulnerability due to incorrect usage of the syslog() function. Local and remote users can send string-formatting operators to the printer daemon to corrupt the daemon's execution, potentially gaining root access.

Assignment 2: Describe the State of Intrusion Detection

Write a white paper on any single intrusion detection technology or challenge. This can be any IDS, IDS technology or approach, or network pattern.

2.1 Introduction

Most IDS are implemented on large networks, or on machines that are permanently connected to the Internet. What if the average home user wish to utilize the facilities of a IDS on their machine that they use to connect to the Internet with?

Most home users operate some Win32 operating system, which can include any version of Windows 9x, Windows ME, Windows NT or Windows 2000, with a modem connection to the Internet. They usually only have one machine and do not have consistent IP addresses.

The objective here is to deploy Snort onto Windows operating systems, which does not have the same computing power as most servers on networks do. The idea is to have a simple way to run Snort, such that even users who are not familiar with computers can do this.

Snort is not going to be running continuously, and because I am assuming most users will only wish to see what type of traffics are attempting connections during their time online, but not really wishing to be alerted each time, the traffic is going to be stored in log form for analysis at a later time. Hence in effect, I am building a *flight recorder*.

The analysis of the logs will be via another application, producing output that is readable and can be understood by even novice's home users.

2.2 The Environment

This implementation has been tested on Windows 98, Windows NT 4.0 and Windows 2000.

All Windows machines use a non-permanent PPP connection to the Internet via a modem. There are no intermediate firewalls between any of the Win32 machines and the ISPs the PPP connections are made to, so in effect all the Win32 machines are in the 'firing range' and are unprotected.

Snort is installed and deployed on each of the Win32 machines, and it is only executed upon a PPP connection. Once the PPP connection is terminated, Snort can be turned off. Logs are created and stored, but the output is not printed to the screen.

2.3 Standard Procedures

Snort for Win32 can be obtained from http://www.snort.org. I downloaded the following version:

snort-1.7-win32-static.zip

Apart from the Snort executable, all Win32 operating systems also requires the Winpcap driver (obtained from http://netgroup-serv.polito.it/winpcap/).

Winpcap

WinPcap is an architecture for packet capture and network analysis for Win32 platforms. It includes a kernel-level packet filter, a low-level dynamic link library (packet.dll), and a high-level and system-independent library (wpcap.dll, based on libpcap version 0.5).

- The packet filter is a device driver that adds to Windows 95, Windows 98, Windows ME, Windows NT and Windows 2000 the ability to capture and send raw data from a network card, with the possibility to filter and store in a buffer the captured packets.
- Packet.dll is an API that can be used to access directly the functions of the packet driver, offering a programming interface independent from the Microsoft OS.
- Wpcap.dll exports a set of high-level capture primitives that are compatible with libpcap. These functions allow capturing packets in a way independent from the underlying network hardware and operating system.

2.3.1 Installation

1. Install the Winpcap driver (the computer may need to be restarted afterwards)
2. Unzip the Snort package into h:\apps\snort\
3. Rename the folder the Snort executable is unzipped to, to bin (i.e. h:\apps\snort\bin)
4. Create the following folders
    - H:\apps\snort\logs
    - H:\apps\snort\alerts
    - H:\apps\snort\etc
5. Add h:\apps\snort\bin to your path – this can be done from the Control Panel
6. Download the Snort rules from http://www.whitehats.com/ids/index.html, and place this into h:\apps\snort\bin (replacing the default snort.conf file)

It is possible to further configure the snort.conf file, but the rule sets from http://www.whitehats.com is quite inclusive already and should not really need any further rules added. The network host IP can be hard-coded into the snort.conf file, but since the IP will be different for each connection, specifying the IP at the command line will be more efficient.

2.3.2 Retrieving the IP

Because on a non-permanent PPP connection, the IP address of the machine is not static, it is different with each connection. Hence if we wish to use the IP address of the current connection analysis the Snort logs, then we will need to capture the IP details for each connection.

This can be done by using a batch script called get-ip.bat, containing only one line:

ipconfig > "h:\apps\snort\etc\ip-address.txt"

The script needs to be executed at the beginning, after the machine has established a connection with the ISP and have obtained it IP for that session.

2.3.3 Flight Recorder Mode

Upon connection of the PPP link, the IP address needs to be captured using the script created in the above section. Then start Snort to capture all traffic (no filtering is done here, I want to see everything).

snort -D -l "h:\apps\snort\logs" -b -i 2

D - This will start Snort in daemon mode.
l – This informs Snort where to place the logs
b – This instructs Snort to log in tcpdump binary format
i – This informs Snort which interface to listen on

To find out what interface Snort needs to listen on, use

Snort –W

Performing this will cause the following:

- A Snort process will appear in the Task Manager (on Windows NT and Windows 2000)
- A log file will be created in the log directory. The log file will have the following format:

Snort-mmdd@hhmm.log

Where:

mm (1st) – month
dd – day
hh – hour
mm (2nd) – minutes

Note: Make sure that the log directory is on a separate disk from the system, log files can grow to an large size, and if they are on the same disk as the system, this can cause the system to crash due to lack of disk space.

2.3.4 Analysis Mode

The logs can be analyzed using the following Snort command:

snort -d -h <host> -A full -N -c <rules> -r <log> -l "h:\apps\snort\alerts"

d – this instructs Snort to dump the application layer
h – this informs Snort what the IP address of the current host is, overriding the value in the snort.conf file
A full – this instructs Snort to switch on full alert
N – this instructs Snort to turn off logging (but alerts are still enabled)
c – this informs Snort what rules file to use
r – this informs Snort what logs file to analyze
l – this informs Snort where to place the logs

This will trawl through the binary logs created in the last section, and using the snort.conf rules set, it will generate alerts on any activity that triggers the rules set. These alerts are stored in h:\apps\snort\alerts.

The alerts will be stored in *alert.ids*.

2.3.5 Log Parser

To help understand what alerts have been generated, the following tool was used:

WinSnort2Html: This takes the alert log files and parses them into an HTML page.

WinSnort2Html is written in Visual Basic, hence the program requires VB 5.0 or later runtime libraries.

WinSnort2Html can be obtained from http://home.earthlink.net/~ckoutras.

Installation

The binary is in compressed form, just uncompress it using WinZip into a chosen directory.

How it Works

The program works with "alert.ids" files created by Snort using either "full" or "fast" alerts.

The program looks at the first line of the "alert.ids" file to determine the alert type. If the first line starts with a date the alert type for the entire file is assumed to be fast. However, if the first line starts with "[**]" and ends with "[**]", then the alert type for the entire files is assumed to be "full". Since the program only checks the first line, the alert file must only contain one type of alert. If the HTML page is blank or some of the table elements are missing, than the file most likely contains mixed alerts. Additionally, the program does not work with full alerts that contain MAC address information.

Full alert log sample:

```
[**] IDS234 Test Alert [**]
01/20-05:52:07.932471 x.x.x.x:61067 -> y.y.y.y:80
TCP TTL:128 TOS:0x0 ID:11270  DF
*****PA* Seq: 0x1EF5DC   Ack: 0x9C3FDAD3   Win: 0x860

[**] CVE-1999-0175 - WEB-MISC-convert.bas Attempt [**]
01/22-15:40:42.683373 a.a.a.a:1026 -> b.b.b.b:1043
TCP TTL:41 TOS:0x0 ID:30254  DF
```

******A* Seq: 0x11D06557 Ack: 0x15A69C Win: 0x7FB8

[**] CAN-1999-0229 - IIS WEB-Attack [**]
01/24-18:27:59.193523 c.c.c.c:80 -> d.d.d.d:1137
TCP TTL:50 TOS:0x0 ID:53997 DF
******A* Seq: 0x5CC103ED Ack: 0x118A9A Win: 0x7FB8

Fast Alert Sample:

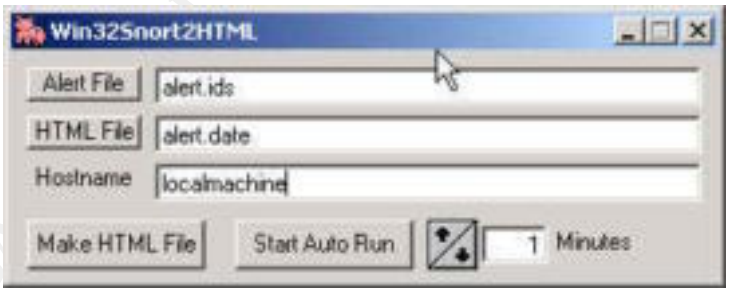
02/07-15:38:49.111693 [**] CVE-1999-0175 - WEB-MISC-convert.bas Attempt [**] a.a.a.a:80 -> b.b.b.b:1331
02/07-15:38:49.176369 [**] CAN-1999-0229 - IIS WEB-Attack [**] c.c.c.c:80 -> d.d.d.d:1331

All preferences are saved into the winsnort2html.ini file, which is located in the c:\windows\system directory.

Each alert line in the HTML page has several hyperlinks. If the alert text begins with either CVE or CAN, the text is hyperlinked to search the Common Vulnerabilities and Exploits database maintained by Mitre Corporation. This database can be found at http://cve.mitre.org. Alerts that begin with IDS are hyperlinked to search the White Hats database, this database can be found at http://www.whitehats.com. Source and destination IP addresses are hyperlinked to the Arin Whois database. The port numbers are hyperlinked to the port database maintained by http://www.snort.org.

Execution

The tool has the following appearance:



The input file to be parsed is entered into the first field, the output file is entered into the second file and the name of the machine the alert file is stored on, is entered into the last field.

Depending on the length of the alert file, it may take a few minutes before the HTML file is produced.

2.3.6 Logs Storage

The binary logs will be quite large in size, so after they have been passed through the Analysis Mode, they should be deleted from the computer. Since we are dealing primary with home users, who usually do not have large storage facilities, storing logs will not be very feasible.

2.4 The Snort Rules Set

The rules are used to define what Snort should consider as hostile traffic. They define everything from 'who' is involved (source and destination) to 'what' is considered as hostile (i.e. invalid TCP flag settings).

Rules can be written to be very specific. Looking for particular payloads and packets attributes, or they can be very general, specifying only a single IP or port.

Each rule has two parts: the rule header and the rule options. The rule headers are required, but the rule options are not.

For example, below is an extract from the rules set obtained from http://www.whitehats.com:

alert TCP $EXTERNAL any -> $INTERNAL 32771:34000 (msg: "IDS242/rpc_rpc.ttdbserv-solaris-overflow"; dsize: >999; flags: A+; content: "|C0 22 3F FC A2 02 20 09 C0 2C 7F FF E2 22 3F F4|";)
alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS430/web-cgi_http-php_strings_exploit-portal-tf8"; flags: A+; content: "?STRENGUR ";)
alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS301/web-misc_http-nessus-404-check"; flags: A+; content: "/nessus_is_probing_you_"; depth: 32;)

Rule Headers

The rule headers from the first rule consist of:

alert TCP $EXTERNAL any -> $INTERNAL 32771:34000

The rule header defines the 'who':

- Protocol – this tells Snort what type of traffic the rule applies to
- Source and destination IPs – this specifies where the hostile traffic originated from and where it is targeted to
- Source and destination ports – this specifies what port the hostile traffic originated from and where it is targeted to
- Direction of traffics – this defines what direction the packets must be travelling in

Snort's detection engine breaks the comparison of a packet into two parts, corresponding to the parts of a rule. The first comparison compares the rule header to the packet, if the packet does not fit the profile of one of the rule headers in a rules set, the detection engine moves onto the next packet. If the packet fits the profile, then the detection engine continues on to test the rule options.

The first field of the rule headers is the *action* field, this instructs Snort as to what it is supposed to do if the rule is triggered. The value for the field can be one of three options:

- Alert – this instructs Snort to create an entry in the "alert.ids" file and to log the packet as well.
- Log – this instructs Snort to only make a log entry
- Pass – this instructs Snort to drop the packet and not to do any further processing of it

Rule Options

The rule options from the first rule consist of:

(msg: "IDS242/rpc_rpc.ttdbserv-solaris-overflow"; dsize: >999; flags: A+; content: "|C0 22 3F FC A2 02 20 09 C0 2C 7F FF E2 22 3F F4|";)

The rule options defines the 'what':

- It tells Snort what packet attributes must be inspected
- It forms a signature defining a specific attack or probe

The rule options is enclosed in parentheses ('(','')'), it must start and end with parenthesis. The syntax used in the rule options is the same for both the packet attributes and actions, it generally consists of an attribute or action keyword, followed by a value. Everything appears in pairs and uses a simple syntax structure that is followed by every item.

Each attribute is separated by a semi-colon (;) and the last attribute in the rule option section must be terminated with a semi-colon. Each attribute has the following format:

msg: "IDS242/rpc_rpc.ttdbserv-solaris-overflow"

- Msg – this is the keyword
- "IDS242/rpc_rpc.ttdbserv-solaris-overflow" – this is the value

A colon (:) separates the keyword and value for each attribute.

Summary

Once a rule has been triggered, no more processing is done on the packet, hence all generic rules need to be placed at the top of the rules set.

The rules set from http://www.whitehat.com is updated on a regular basis and contains rules to capture all the latest attacks. Obtaining a ready-made rules set will enable home users to deploy Snort without having to write their own rules, or to analyse what attacks to be on the alert for.

Assignment 3: "Analyze This" Scenario

Analyze a set of Snort logs for a University. The Snort system uses a fairly standard rulebase.

Produce an analysis report based on the data.

3.1 Data Overview

Data was taken for a period of seven days, from the 2 July 2001 to the 8 July 2001. The set of data consisted of:

Alert Logs	Scan Logs	OOS Logs
Alertalert.010702.clean	Scansscans.010702.clean	Oos_Jul.02.01
Alertalert.010703.clean	Scansscans.010703.clean	Oos_Jul.03.01
Alertalert.010704.clean	Scansscans.010704.clean	Oos_Jul.04.01
Alertalert.010705.clean	Scansscans.010705.clean	Oos_Jul.05.01
Alertalert.010706.clean	Scansscans.010706.clean	Oos_Jul.06.01
Alertalert.010707.clean	Scansscans.010707.clean	Oos_Jul.07.01
Alertalert.010708.clean	Scansscans.010708.clean	Oos_Jul.08.01

3.2 Detect Analysis and Summary

This section represents the analysis results in tabular and graphical forms.

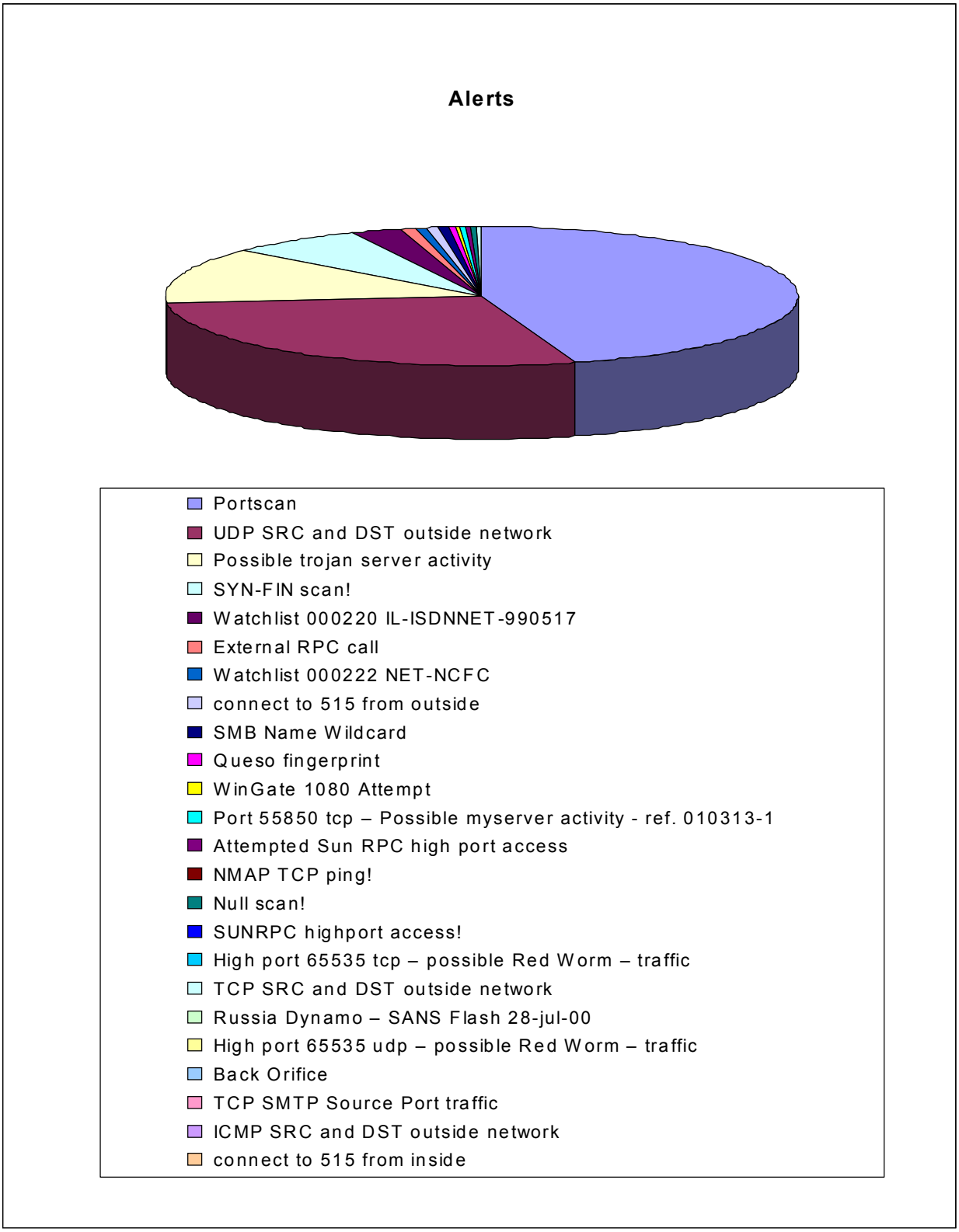
3.2.1 Alerts Logs

There were 118,834 alerts generated over the seven days, with ***Portscans*** making up the majority of the alerts.

This is summarised in the following table and pie chart. Then each alert type is investigated further, each time listing (if possible) the 'top talkers', the most common ports, further information on the top source IP (both external and internal) and any correlation if possible. And finally, security recommendations will be given on how to minimise the risk from each alert.

Type of Alert	Occurrences
Portscan	53720
UDP SRC and DST outside network	33306
Possible trojan server activity	15496
SYN-FIN scan!	8521
Watchlist 000220 IL-ISDNNET-990517	2952
External RPC call	1018
SMB Name Wildcard	644
Watchlist 000222 NET-NCFC	623
connect to 515 from outside	608
Queso fingerprint	439

WinGate 1080 Attempt	351
Port 55850 tcp – Possible myserver activity – ref. 010313-1	273
SUNRPC highport access!	193
Attempted Sun RPC high port access	165
Null scan!	131
NMAP TCP ping!	115
High port 65535 tcp – possible Red Worm – traffic	90
TCP SRC and DST outside network	80
Russia Dynamo – SANS Flash 28-jul-00	64
High port 65535 udp – possible Red Worm – traffic	31
ICMP SRC and DST outside network	6
Back Orifice	3
TCP SMTP Source Port traffic	3
connect to 515 from inside	2
TOTAL	**118834**



Portscans

There were a total of 53,720 port scans detected over the week. The port scans can be separated into the following:

Internal (My.Net.x.x) -> External = 34,297
External -> Internal (My.Net.x.x) = 19,423

External to Internal port scans are used to identify open ports on the target system/network, this can lead to information being gathered that can be used for further attacks.

As for Internal to External port scans, this can be someone on the internal network scanning some external sites. Or a compromised host, being used as an intermediate station for scanning.

Top ten internal port scanning hosts:

IP	Occurrences
MY.NET.160.114	19599
MY.NET.218.214	827
MY.NET.98.127	623
MY.NET.217.10	608
MY.NET.100.230	601
MY.NET.75.196	403
MY.NET.97.200	398
MY.NET.98.133	335
MY.NET.98.246	322
MY.NET.98.205	288

Looking at host My.Net.160.114, which generated the most port scan alerts:

Extracted from scan log of 2 July 2001

```
Jul  2 04:29:57 MY.NET.160.114:777 -> 24.43.12.34:27005 UDP
Jul  2 04:29:57 MY.NET.160.114:777 -> 24.101.13.103:64473 UDP
Jul  2 04:29:56 MY.NET.160.114:777 -> 202.132.49.15:1630 UDP
Jul  2 04:30:00 MY.NET.160.114:777 -> 24.43.12.34:27005 UDP
Jul  2 04:30:00 MY.NET.160.114:777 -> 24.101.13.103:64473 UDP
```

This host appears many times throughout the week, always displaying the same pattern: scanning for UDP ports, and using a static source port of 777. The UDP port 777 is used by Multiling HTTP, so this may be a false positive. What we are seeing may be responses to requests, an example of such a stimulus can be:

```
22.43.12.34:27005 -> My.Net.160.114:777 UDP
```

Port scans will normally be used to scan different targets, having so many connects to the same target in such a short space of time is not normal behavior of a port scan. May be this is the result of a badly configured network, where the sensors are only seeing traffic in one direction.

Top ten external port scanning hosts:

IP	Occurrences
205.188.233.153	2336
205.188.233.121	2270
205.188.244.249	1745
128.143.75.164	1331
205.188.246.121	1204
205.188.233.185	1146
205.188.244.121	916
64.37.156.9	483
61.222.34.170	308
148.223.228.15	307

IP 205.188.233.153 resolves to the hostname g2lb5.spinner.com, and is registered to Spinner Networks, Inc in the US.

Extracted from scan log of 2 July 2001

```
Jul  2 14:43:51 205.188.233.153:22028 -> MY.NET.110.33:6970 UDP
Jul  2 14:43:51 205.188.233.153:7182 -> MY.NET.180.76:6970 UDP
Jul  2 14:43:51 205.188.233.153:21870 -> MY.NET.145.197:6970 UDP
Jul  2 14:43:51 205.188.233.153:18302 -> MY.NET.70.92:6972 UDP
Jul  2 14:43:51 205.188.233.153:18144 -> MY.NET.71.248:6970 UDP
Jul  2 14:43:51 205.188.233.153:26204 -> MY.NET.111.30:6970 UDP
Jul  2 14:43:51 205.188.233.153:9628 -> MY.NET.109.62:6970 UDP
Jul  2 14:43:51 205.188.233.153:10252 -> MY.NET.108.13:6970 UDP
```

This trace seems to show a normal port scan. The host 205.188.233.153 is probably trying to map the internal network. The port 6970 is used by the trojan GateCrasher on protocol TCP, but whether the trojan is used on the UDP port or not will require further investigation.

Security Recommendation

Any unnecessary ports on any Internet facing servers should be hidden from external hosts. Or if they cannot be hidden because it is necessary for them to be accessible from the Internet, then access to these ports needs to be properly controlled at the firewalls.

All internal servers should be checked for any abnormal ports that are listening on them. These could be trojans, and in which case, any servers that have been compromised must be taken offline immediately.

All internal hosts that performed port scanning, should be taken offline and investigated to check whether they have been compromised.

TCP/UDP/ICMP SRC & DST Outside Network

These alerts are triggered by traffic with external source IPs attempting connections to external destination IPs. This is not normal network behaviour, unless the organisation's routers and servers are being used as proxies or relays.

With such abnormal packets, it is highly likely that the source IP of the packets is spoofed. Such behaviour can be seen in the mstream DDOS tool (http://www.cert.org/incident_notes/IN-2000-05.html), which sends out packets to targets with randomly spoofed source IP addresses.

UDP SRC

Top ten source IP:

IP	Occurrences
63.250.213.73	25821
63.250.213.124	2331
169.254.161.0	2083
63.250.213.26	1788
63.250.213.100	173
167.102.5.44	173
169.254.101.152	143
192.168.0.102	138
18.236.0.28	101
169.254.192.111	83

Top ten destination IP:

IP	Occurrences
233.28.65.227	25821
233.40.70.50	2331
233.28.65.164	1788
130.132.143.42	1067
130.132.143.43	1016
233.28.65.61	173
167.102.7.105	161
162.129.20.10	138
206.27.242.2	58
18.70.0.160	55

Destination ports:

Port	Occurrences
5779	30113
137	2744
53	438
67	7
138	4

TCP SRC

Top ten source IP:

IP	Occurrences
24.180.140.132	48
192.168.1.1	8
129.105.28.214	4
172.142.95.96	3
172.142.77.105	3
172.186.126.23	3
172.130.50.110	2
172.141.113.131	2
172.146.90.11	2
172.139.47.194	2

Top ten destination IP:

IP	Occurrences
213.122.166.185	37
192.168.1.3	8
64.12.163.199	4
213.1.130.184	4
24.176.87.45	3
24.0.28.71	3
216.164.58.132	3
199.174.143.34	2
213.122.170.205	2
209.155.224.20	2

Top ten destination ports:

Port	Occurrences
1243	8
9898	4
1736	3
21579	3
3392	3
21599	3
1214	3
21617	3
21196	2
2173	2

ICMP SRC

Source IP:

IP	Occurrences

172.165.110.185	4
172.128.86.85	1
172.139.126.250	1

Destination IP:

IP	Occurrences
64.198.236.197	1
65.33.239.57	1
24.23.237.125	2
193.252.183.44	2

There were only 4 occurrences of ICMP traffic flowing between external hosts. The first occurrence was on 4 July 2001 at 3:25pm and the last occurrences was on 8 July 2001.

```
07/04-15:35:36.328300  [**] ICMP SRC and DST outside network [**] 172.128.86.85 ->
64.198.236.197
07/07-17:53:17.829372  [**] ICMP SRC and DST outside network [**] 172.139.126.250 ->
65.33.239.57
07/08-14:58:11.225708  [**] ICMP SRC and DST outside network [**] 172.165.110.185 ->
24.23.237.125
07/08-14:58:17.139285  [**] ICMP SRC and DST outside network [**] 172.165.110.185 ->
24.23.237.125
07/08-15:21:36.113252  [**] ICMP SRC and DST outside network [**] 172.165.110.185 ->
193.252.183.44
07/08-15:24:26.410863  [**] ICMP SRC and DST outside network [**] 172.165.110.185 ->
193.252.183.44
```

One possible explanation for such a behavior pattern is that some internal host is performing a Denial of Service attack, using spoofed ICMP packets (i.e. ICMP echo requests). The internal host may have been compromised, and being used as a DDOS agent.

Security Recommendation

Packets with source IP that do not belong to the internal network address range, but originate internally, and destined for IP external to the University, should not be allowed to pass through the firewalls.

In general, traffic should not be allowed to flow from external hosts to external hosts, through the network components of the University.

Possible Trojan Server Activity

There were 15,496 occurrences of this type of alerts over the seven days. These alerts warn that external IP may be connecting to trojans running on internal machines, or vice versa.

Internal (My.Net.x.x) -> External = 2,367
External -> Internal (My.Net.x.x) = 13,129

Taking a closer look, here is an extract from the alert log of 2 Jul 2001 (alertalert.010702.clean):

```
07/02-00:03:22.492439  [**] Possible trojan server activity [**] 216.104.129.2:3922 ->
MY.NET.15.69:27374
07/02-00:03:22.493778  [**] Possible trojan server activity [**] MY.NET.15.69:27374 ->
216.104.129.2:3922
07/02-00:03:22.969441  [**] Possible trojan server activity [**] 216.104.129.2:3922 ->
MY.NET.15.69:27374
07/02-00:03:22.969693  [**] Possible trojan server activity [**] MY.NET.15.69:27374 ->
216.104.129.2:3922
```

These alerts clearly shows communication between an external IP (216.104.129.2) and an internal IP (My.Net.15.69) on port 27374, which happens to be the port used by the SubSeven trojan. This brings me to conclude that either the server with the IP My.Net.15.69 has been compromised, or a legitimate program is using that particular port for communication and this is a false positive.

Another example is also taken from the same day, but shows a different behavior:

```
07/02-02:10:53.739048  [**] Possible trojan server activity [**] 193.158.154.68:3142 ->
MY.NET.120.183:27374
07/02-02:10:56.753826  [**] Possible trojan server activity [**] 193.158.154.68:3142 ->
MY.NET.120.183:27374
07/02-02:11:02.796930  [**] Possible trojan server activity [**] 193.158.154.68:3142 ->
MY.NET.120.183:27374
07/02-02:11:14.856737  [**] Possible trojan server activity [**] 193.158.154.68:3142 ->
MY.NET.120.183:27374
```

With this set of alerts, this shows an attempt to connect from 193.158.154.68 to an internal IP (My.Net.120.183) on the SubSeven port. The external IP attempts to connect four times, all within a few seconds and maintaining a static source port. This is the typical behaviour of TCP, which will attempt to connect four times, before it gives up. Hence from this extract, I can conclude that at 2:10am, the host 193.158.154.68 attempted a connection with My.Net.120.183 and the connection was unsuccessful.

Later in the day, at around 7pm, the same host attempts again to connect the SubSeven port again on a different target. Below are the offending logs:

```
07/02-19:23:27.466049  [**] Possible trojan server activity [**] 193.158.154.68:1295 ->
MY.NET.6.64:27374
07/02-19:23:33.498311  [**] Possible trojan server activity [**] 193.158.154.68:1295 ->
MY.NET.6.64:27374
```

There are only two attempted connections here, may be the attacker terminated the attempts before the connection retried for the third time. Because the same source host has been seen attempting connections to various internal machines, and because the time between each attempt are far apart, this leads me to conclude that these are all manual attempts.

The source IP 193.158.154.68 resolves to the hostname LAPTOP_1, the netblock 193.0.0.0 – 193.255.255.255 is assigned to European users and is maintained by RIPE.

Security Recommendations

Users should be made aware of the danger of downloading programs from the Internet and opening attachments in emails. By making users aware, this will minimise the chance of malicious programs (trojans) entering the internal network.

Also, it seems that all alerts of this type were triggered by hosts attempting connections to the SubSeven port. It may be useful to investigate the internal hosts that are being targeted, especially the ones that actually responded to the probes.

SYN-FIN Scan

These are most likely crafted packets with both the SYN and FIN flags set, and are used to bypass any IDS that have not been configured to recognise them. These scans are largely used to identify target operating systems, or to locate active TCP ports on the target's network. Hence the source IP is highly unlikely to be spoofed, since these scans are designed to elicit response (thus these alerts can be used to reverse track the attackers) and each connection is an intentional probe.

For example:

A SYN + FIN sent to an open or closed on LINUX (Kernel 2.2.x) will receive a RST + ACK in reply. Whilst a SYN + FIN sent to an open or closed port on Windows NT Workstation 4 with SP6a, will receive a SYN + ACK in reply.

From the logs:

IP	Source Port	Destination port	Occurrences
213.255.24.48	21	21	8247
211.180.236.194	111	111	269
212.185.222.86	1214	1	1
24.200.179.202	4098	5631	1
62.149.150.37	8192	2048	1
64.196.112.126	32808	259	1
64.198.134.34	32808	259	1

The host generating the most traffic is 213.255.24.48, which resolves to the hostname h255-24-48.pd1.albacom.net, and registered to Albacom S.p.a., in Rome (Italy). The host uses a source port of 21 and scanning for port 21 on the destination host:

```
07/06-08:52:03.415713  [**] SYN-FIN scan! [**] 213.255.24.48:21 -> MY.NET.1.5:21
07/06-08:52:03.564820  [**] SYN-FIN scan! [**] 213.255.24.48:21 -> MY.NET.1.13:21
07/06-08:52:03.601885  [**] SYN-FIN scan! [**] 213.255.24.48:21 -> MY.NET.1.15:21
07/06-08:52:03.850787  [**] SYN-FIN scan! [**] 213.255.24.48:21 -> MY.NET.1.26:21
07/06-08:52:03.970637  [**] SYN-FIN scan! [**] 213.255.24.48:21 -> MY.NET.1.32:21
```

The above is an extract from the alert logs of 6 July 2001

It can be seen that these packets are crafted, since the source port is static for all connections, even though the destination host is different for each one.

From a similar analysis performed by **PJ Goodwin** (Analyst ID 305; http://www.sans.org/y2k/practical/PJ_Goodwin_GCIA.doc), it can be seen that port 21 (FTP) is a popular port scanned by attackers. From PJ Goodwin's results, he observed 19,613 probe to port 21 over the period of two months. Here is a extract taken from his report:

> *According to SAN's Griffin List for 1/3/01 – 0900, Port 21 was the top port scanned by attackers.*
> *This can be interpreted as a hostile attempt to find open FTP servers. This could be a precursor to FTP abuse or pre-planning of maliciousness.*
>
> *It also could be an attempt to fingerprint each system in an attempt to gather more information about the targeted network. In this case, it could be a precursor to an attack on the target.*

Security Recommendation

Since the source IP is unlikely to be spoofed in such an attack, the host may be compromised and is unknowingly being used as a tool. Contact the system administrator for the company where the IP is registered to, and have them investigate the host further.

Also, all unnecessary traffic should be denied access to the network. Occasionally such traffic will be required, and in these cases, access must be properly controlled at the firewalls and routers.

Watchlist 000220 IL-ISDNNET-990517/ Watchlist 000222 NET-NCFC

These alerts involve IPs from Israel and China. The breakdown is as follows:

Watchlist 000220 IL-ISDNNET-990517 –

Percentage of occurrence: 2%
IP Range: 212.179.xxx.xxx
Geographical: Israel

Top Five Hosts:

IP	Occurrences
212.179.34.114	573
212.179.27.6	220
212.179.22.99	191
212.179.41.235	170
212.179.33.1	97

Watchlist 000222 NET-NCFC –

Percentage of occurrence: 0.57%
IP Range: 159.226.xxx.xxx
Geographical: China

Top Five Hosts:

IP	Occurrences
159.226.41.166	525
159.226.114.1	39
159.226.63.190	13
159.226.120.17	12
159.226.67.61	11

Destination ports from all location:

Port	Occurrences
1214	2596
32825	525
1372	129
25	103
6346	98
11622	41
9549	28
4253	17
8765	9
113	7
6347	7
4617	5
4256	2
37213	1
37662	1
38580	1
39555	1
39601	1
48971	1
55294	1
64006	1

Analyzing the destination ports for these alerts, it seems some of them are connections to ports used by possible trojans. For example:

```
07/07-00:31:17.383809  [**] Watchlist 000222 NET-NCFC [**] 159.226.5.222:1797 ->
MY.NET.100.230:113
07/07-00:33:47.320947  [**] Watchlist 000222 NET-NCFC [**] 159.226.5.222:1806 ->
MY.NET.100.230:113
```

From the above extract (taken from the alert logs of 7 July 2001), the destination port of the alerts is 113. This is the known port used by ident, an Authentication Service. But it is also the port used by the Invisible Identd Daemon trojan!

However there may be some false positives. For example, the ports 6346 and 6347 are used by the Gnutella application, which is a MP3 sharing program similar to Napster. The fact that some host from either Israel or China connected to those ports within the My.Net.x.x network does not necessarily mean it is malicious activity. Some host within My.Net.x.x may be using Gnutella and are sharing files externally.

These types of alerts have appeared many times before, and according to **David Singer** (Analyst ID 353; http://www.sans.org/y2k/practical/David_Singer_GCIA.doc), "*these alerts have been removed from the current rulebase and should not be of further concern to us*".

Security Recommendation

Even though it has been pointed out that these alerts can be ignored, if you still feel that they are of concern, then just block any packets arriving from these IP ranges at the firewalls.

Also, as general security practice, unnecessary ports should be disabled and hidden from the Internet and external file sharing should also be disabled.

External RPC Call

These alerts detect external connections to the TCP/UDP port of 111 on some internal machine. This can be attributed to scanning for portmap on Unix systems.

RPC is a networking technology developed by Sun Microsystems. It is used on most UNIX machines, and is a popular way of building networked applications. It allows programs on local UNIX systems to execute commands on remote UNIX systems.

Its popularity translates into lots of programs that may have holes. Scanning for RPC is the first stage in looking for those particular programs. Once it is identified that RPC is running on a system, the intruder will then perform a RPC portmap dump, which would list all the RPC programs on the target and tell the intruder if there are any exploits that can be used.

Top ten source IP:

IP	Occurrences
199.84.54.32	311
204.117.207.245	114
216.21.132.81	95
203.186.220.10	90
170.211.172.90	70
211.21.44.101	59
195.46.96.102	55
61.218.145.218	54
216.21.159.88	51
152.101.24.249	45

Looking at host 199.84.54.32 (registered to CA*NET Network Operations Centre), this gives clear evidence of portmap scanning:

Extracted from alert log of 2 July 2001

```
07/02-09:08:18.099722  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.1:111
07/02-09:08:18.122793  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.3:111
07/02-09:08:18.221180  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.12:111
07/02-09:08:18.240969  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.14:111
07/02-09:08:18.261594  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.16:111
07/02-09:08:18.279660  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.18:111
07/02-09:08:18.304633  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.20:111
07/02-09:08:18.332141  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.22:111
07/02-09:08:18.353208  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.24:111
07/02-09:08:18.411274  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.30:111
07/02-09:08:18.425018  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.32:111
07/02-09:08:18.512703  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.40:111
07/02-09:08:18.646718  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.54:111
07/02-09:08:18.670974  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.56:111
07/02-09:08:18.687435  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.58:111
07/02-09:08:18.702961  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.60:111
07/02-09:08:18.908727  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.80:111
07/02-09:08:18.929605  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.82:111
07/02-09:08:18.948252  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.84:111
07/02-09:08:18.970067  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.86:111
07/02-09:08:18.990292  [**] External RPC call [**] 199.84.54.32:111 -> MY.NET.132.88:111
```

Because of the static source port, these packets are most likely crafted. The alert logs do not tell us what protocol is being used, so looking at the scan log for the same day, this shows us that the protocol is TCP:

Extracted from scan log of 2 July 2001 (in correlation with the internal host My.Net.132.88 from the above alert log extract)

```
Jul  2 09:08:18 199.84.54.32:111 -> MY.NET.132.88:111 SYN **S*****
```

Apart from searching for vulnerable portmaps, the Sadmind worm also uses the same port. Since host 199.84.54.32 is scanning TCP port 111, may be it is the activity of the Sadmind worm. In this case, it is highly likely that host 199.84.54.32 has been also compromised and being used as an intermediate attacking platform.

Security Recommendation

If RPC is not required on the Unix servers, then they should be disabled. If they are necessary for the server, then RPC should not be made available externally by blocking it at the firewalls.

SMB Name Wildcard

This is information gathering. Windows machines often exchange these queries as a part of the file sharing protocol to determine NetBIOS names when only IP addresses are known.

An attacker could use this same query to extract useful information such as workstation name, domain, and users currently logged in, by performing a deliberate scan for port 137.

There were a total of 644 alerts generated for this, 642 are from external sites to internal, and 2 are from internal sites to external.

From the logs (top ten external IP):

IP	Occurrences
130.13.135.239	15
130.13.64.30	9
207.136.38.129	9
130.13.138.244	7
130.54.113.11	7
130.113.224.125	6
130.13.64.211	6
130.13.79.197	6
130.64.43.125	6
200.59.34.140	6

The hosts from the list above, and all the other external hosts that triggered this alert, have attempted connection with port 137 on some internal host. But a probe of port 137 by itself is not enough evidence of an attack, however a simultaneous connection to port 139 could indicate that someone is trying to connect t and access shared resources. Hence these alerts can be seem to be harmless.

The internal host that triggered this alert is MY.NET.162.199. This may be a false positive, where the host is performing legitimate communications with the destination host MY.NET.50.154. Both hosts are in based internally, and even though they are both in different address space, having SMB traffic flowing between them can be attributed to normal business traffic.

Security Recommendation

Ports 135 - 139 must be blocked at the firewalls, so that they cannot be accessed externally. Blocking NetBIOS traffic from leaving the network will minimise the chances of attackers obtaining useful information, which can be used for further attacks.

Connect to 515 from Outside/Inside

The Unix LPR service runs on port 515, and these alerts detects hosts scanning for the LPRng vulnerability. It contains a potential vulnerability that may allow root compromise from both local and remote systems. The vulnerability is due to incorrect usage of the syslog(3) function. Local and remote users can send string-formatting operators to the printer daemon to corrupt the daemon's execution, potentially gaining root access

Extract taken from http://www.firewall-1.org/2001-04/msg00360.html

These alerts can also be attributed to the spread of the Adore Worm, which scans target Linux hosts for the LRRng vulnerability.

There were 608 alerts triggered from connections from external sources. These alerts were triggered by 4 different hosts:

IP	Occurrences
165.132.31.137	432
210.103.58.65	113
217.96.133.163	62
255.255.255.255	1

The final host on the list (255.255.255.255) is of particular interest:

```
07/03-05:37:56.209410  [**] connect to 515 from outside [**] 255.255.255.255:31337 ->
MY.NET.135.58:515
```

This is the broadcast address and the source port is that used by Back Orifice. This is unusual and the target host should need to be investigated, since broadcast addresses should never be seen as a source address. Also, broadcast addresses should not be forwarded on from routers. Because of this abnormal pattern, this leads me to conclude that this packet is crafted and its purpose is malicious.

As for the IP 165.132.31.137, this is registered to Yonsei University in Korea. It has made many connections to port 515 on various internal hosts during 4 July 2001:

```
07/04-11:14:48.296332  [**] connect to 515 from outside [**] 165.132.31.137:3024 ->
MY.NET.133.222:515
07/04-11:14:48.414477  [**] connect to 515 from outside [**] 165.132.31.137:3032 ->
MY.NET.133.230:515
07/04-11:14:48.443853  [**] connect to 515 from outside [**] 165.132.31.137:3034 ->
MY.NET.133.232:515
07/04-11:14:48.466308  [**] connect to 515 from outside [**] 165.132.31.137:3036 ->
MY.NET.133.234:515
07/04-11:14:48.496351  [**] connect to 515 from outside [**] 165.132.31.137:3038 ->
MY.NET.133.236:515
07/04-11:14:48.996891  [**] connect to 515 from outside [**] 165.132.31.137:2267 ->
MY.NET.132.20:515
```

And here is the same host in the scan logs of the same day:

```
Jul  4 11:14:45 165.132.31.137:2252 -> MY.NET.132.5:515 SYN **S*****
Jul  4 11:14:45 165.132.31.137:2253 -> MY.NET.132.6:515 SYN **S*****
Jul  4 11:14:45 165.132.31.137:2255 -> MY.NET.132.8:515 SYN **S*****
Jul  4 11:14:45 165.132.31.137:2259 -> MY.NET.132.12:515 SYN **S*****
Jul  4 11:14:45 165.132.31.137:2264 -> MY.NET.132.17:515 SYN **S*****
```

From the alert and scan extracts, it is evident that the host 165.132.31.137 is scanning for the LPR service.

Because this host belongs to a University, it may not be so strange that it is connecting to a print service in our University (but this is still not normal), since some student may be

trying to send data to our students. But what is intriguing, is that the host made 430 connections to port 515 on hosts on the My.Net.x.x network, in less than one minute. This is evidence of an automated script at work, hence it is highly likely that the host 165.132.31.137 has been compromised. Because the protocol used is TCP and this is a scan, it is unlikely for the source IP to be spoofed, unless the attacker is not expecting to receive any response from the targets.

There were 2 connections to port 515 from an internal host:

```
07/05-22:08:31.834644  [**] connect to 515 from inside [**] MY.NET.179.78:51422
-> 24.13.123.8:515
07/08-21:32:37.731932  [**] connect to 515 from inside [**] MY.NET.179.78:34101
-> 24.13.123.8:515
```

The IP 24.13.123.8 is registered to Home Network based in the US. It is interesting that My.Net.179.78 is attempting to connect to some external site for the print service.

To investigate whether this pattern of activity has occurred before or not, I read through some reports produced by other analysts. **Beck Bogle** (Analyst ID 339; http://www.sans.org/y2k/practical/Becky_Bogle_GCIA.doc) observed a similar occurrence from her logs:

> *Destination 212.187.65.135 is registered to Nijmegen Cablemodems in the Netherlands. MY.NET.70.38 connected with port 515 on this host 3 times on 01/04. It seems suspicious that a box on my.net would need to connect to a print service in the Netherlands. Secondly, destination 148.243.214.7 is registered to Coordinacion Nacional de Progresa in Mexico. Host MY.NET.163.17 connected with port 515 on this external host on 12/20 at 21:58. Further analysis shows that on 12/15 (five days earlier) host 141.211.176.99 (registered to University of Michigan) scanned over 2200 boxes for port 515 on MY.NET including MY.NET.163.17. The fact that 141.211.176.99 is a university IP address suggests that it may be a compromised box. There is a possibility that MY.NET.163.17 has also been compromised, and this host needs to be examined more closely.*

I have not witness My.Net.179.78 as a destination for external scans, but my logs only spanned a period of seven days. It is highly possible that this box was compromised at an earlier date. A detailed analysis of earlier logs will be needed to verify this possibility.

Security Recommendation

If the Print Service is not required, then it should be stopped. If it is necessary, then ensure that the port is not visible externally. Internal hosts should not need to connect to some external host for the print service, so traffic flowing in this direction should also be blocked.

Also, monitor vulnerabilities for this service and apply the newest patches when they become available.

Queso Fingerprint

Queso is a port scanner tool similar to Nmap, in which it is used to fingerprint the operating system of the target machines.

Top ten source IP:

IP	Occurrences
199.183.24.194	246
193.226.113.248	110
209.150.103.212	9
63.212.189.228	8
192.117.120.140	6
158.75.57.4	6
209.10.41.242	5
133.127.86.112	5
128.61.38.150	4
141.157.90.81	4

Top ten destination ports:

Port	Occurrences
25	254
1214	113
6346	33
113	13
6347	6
22	5
23	4
1448	3
6355	2
34721	1

The IP 199.183.24.194 resolves to hostname vger.kernel.org, and is registered to Transmeta Corporation in the US. This host performed 246 scans using Queso within seven days, this is a rather high volume of traffic from some external host. Looking deeper:

```
07/02-03:34:43.164039  [**] spp_portscan: PORTSCAN DETECTED from 199.183.24.194 (STEALTH)
[**]
07/02-03:25:56.166349  [**] Queso fingerprint [**] 199.183.24.194:36210 ->
MY.NET.253.41:25
07/02-03:34:44.104707  [**] spp_portscan: portscan status from 199.183.24.194: 1
connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
07/02-03:34:45.128675  [**] spp_portscan: End of portscan from 199.183.24.194 (TOTAL
HOSTS:1 TCP:1 UDP:0) [**]
```

The same pattern is repeated at regular intervals and quite frequently throughout the seven days. Each time the destination is one of three hosts within the My.Net.253.x network (.41, .42 and .43). And each time, the destination port is always port 25.

From the traces, it is evident that host 199.183.24.194 is performing intentional scanning on the My.Net.253.x subnet. A host scanning the same target once or twice can be

attributed to coincidence, but with the same pattern appearing frequently throughout each day and through the week is very suspicion.

But what is strange is that the targets are always the same, which seems to imply that this is a false positive. Since it should not really take so many attempts to fingerprint a group of systems. Looking at the time-span, each scan is quite far apart, and not at regular intervals, which can lead us to discard automated scanning. Maybe this is legitimate SMTP traffic between the host 199.183.24.194 and the 3 hosts on the My.Net.253.x subnet.

In either case, these internal machines need further investigation to decide whether the traffic is legitimate or not.

PJ Goodwin (Analyst ID 305; http://www.sans.org/y2k/practical/PJ_Goodwin_GCIA.doc) also observed a similar pattern in his logs:

```
09/29-00:22:54.391105 [**] spp_portscan: PORTSCAN DETECTED from 24.3.161.193 (STEALTH)
[**]
09/29-00:11:58.584512 [**] Queso fingerprint [**] 24.3.161.193:32811 -> MY.NET.145.9:110
09/29-00:22:56.486852 [**] spp_portscan: portscan status from 24.3.161.193: 1 connections
across 1 hosts: TCP(1), UDP(0) STEALTH [**]
09/29-00:22:59.244973 [**] spp_portscan: End of portscan from 24.3.161.193 (TOTAL HOSTS:1
TCP:1 UDP:0) [**]
```

Below is PJ Goodwin's analysis:

> *It is possible that the number one top talker 24.3.161.193 is causing a false positive. All 43 alerts were directed at the same MY.NET.145.9. The same is true in the correlations. An OS fingerprint of one system would not take 43 attempts spread over the time period of 09/26-04:27:59.343599 - 10/08-17:45:37.010287 11/11-13:25:31.250967.*

Security Recommendation

It is hard to prevent tools such as Queso from fingerprinting our network, but generally all unnecessary traffic to and from My.Net.x.x should be disallowed.

Also in the future, this rule may have to be modified due to the implementation of Explicit Congestion Notification (ECN - http://www.sans.org/y2k/ecn.htm). ECN is a standard proposed by the IETF that will cut down on network congestion and routers dropping packets by using the two reserve bits in the TCP header (bits 8 & 9).

WinGate 1080 Attempt

These are attempts to connect to the WinGate port 1080, and can be alerts that someone is scanning the network for possible WinGate Servers to relay or redirect traffic on.

In particular, the following exploits should be of concern:

- CVE-1999-0290 - The WinGate telnet proxy allows remote attackers to cause a denial of service via a large number of connections to localhost

- CVE-1999-0291 - The WinGate proxy is installed without a password, which allows remote attackers to redirect connections without authentication

- CVE-1999-0441 - Remote attackers can perform a denial of service in WinGate machines using a buffer overflow in the Winsock Redirector Service

- CVE-1999-0494 - Denial of service in WinGate proxy through a buffer overflow in POP3

- CAN-2000-1048 - ** CANDIDATE (under review) ** Directory traversal vulnerability in the logfile service of Wingate 4.1 Beta A and earlier allows remote attackers to read arbitrary files via a .. (dot dot) attack via an HTTP GET request that uses encoded characters in the URL

Above list extracted from http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=wingate

However, some of these alerts may be false positives, since IRC chat servers will also scan clients for open WinGate SOCKS servers.

Information obtained from http://www.whitehats.com/info/IDS175

Top ten source IP:

IP	Occurrences
216.15.205.2	65
217.10.143.54	53
195.159.0.151	32
130.227.3.123	23
195.66.170.8	18
209.217.52.231	15
209.212.128.47	15
161.58.185.242	13
209.116.7.97	11
209.249.9.118	9

The IP 216.15.205.2 resolves to hostname 216.15.205.2, and is registered to Maverick Networks in the US.

Extracted from alert log of 5 July 2001

```
07/05-07:39:31.979907  [**] WinGate 1080 Attempt [**] 216.15.205.2:49260 ->
MY.NET.98.198:1080
07/05-07:39:45.622206  [**] WinGate 1080 Attempt [**] 216.15.205.2:49265 ->
MY.NET.98.198:1080
```

WinGate attempts are very common, for example **PJ Goodwin** (Analyst ID 305; http://www.sans.org/y2k/practical/PJ_Goodwin_GCIA.doc) reported observing 4,802 entries in his logs. Below is a snapshot from his logs:

```
./SnortAle.txt:08/11-01:27:06.939036 [**] WinGate 1080 Attempt [**] 216.179.0.37:2940 ->
MY.NET.60.8:1080
./SnortAle.txt:08/11-01:39:58.051582 [**] WinGate 1080 Attempt [**] 216.67.82.19:2743 ->
MY.NET.98.138:1080
./SnortAle.txt:08/11-01:39:59.043073 [**] WinGate 1080 Attempt [**] 216.67.82.19:2743 ->
MY.NET.98.138:1080
./SnortAle.txt:08/11-01:39:59.940287 [**] WinGate 1080 Attempt [**] 216.67.82.19:2743 ->
MY.NET.98.138:1080
./SnortAle.txt:08/11-01:40:00.744956 [**] WinGate 1080 Attempt [**] 216.67.82.19:2743 ->
MY.NET.98.138:1080
```

Security Recommendation

WinGate is a program that lets user share the usage of their dial-up services. If possible, this service should be disabled from all servers that have implemented it. The port 1080 (and 8080) should be blocked at the firewalls, since traffic from external sources should not be allowed to connect to this port on internal hosts.

Attempted Sun RPC High Port Access/ SUNRPC Highport Access

Solaris rpcbind, which belongs on UDP port 111, is also found on UDP ports above 32770. Thus many packet filters are not effective, resulting in the ability to access rpcbind.

Top ten source IP:

IP	Occurrences
12.25.141.32	42
216.143.37.89	27
216.143.37.155	24
208.171.80.202	18
216.143.36.186	16
65.9.177.233	13
66.26.252.85	12
216.143.37.2	12
207.172.73.101	6
65.8.46.199	4

IP 12.25.141.32 resolves to hostname dewarfi.cm.gscyclone.com, which is registered to GS Net.Works in the US. This host has made 42 attempted connection to RPC services, below is a example of its activity:

```
07/04-11:53:12.061297  [**] SUNRPC highport access! [**] 12.25.141.32:1041 ->
MY.NET.217.6:32771
07/04-11:53:12.194125  [**] SUNRPC highport access! [**] 12.25.141.32:1041 ->
MY.NET.217.6:32771
07/04-11:55:26.447696  [**] SUNRPC highport access! [**] 12.25.141.32:1041 ->
MY.NET.217.6:32771
```

```
07/04-11:55:26.519796  [**] SUNRPC highport access! [**] 12.25.141.32:1041 ->
MY.NET.217.6:32771
```

Port 32771 is a port that is sometimes used by rusersd. The above is just a sample, in the logs the host makes many more attempted connections to this port on the target host. Each time using the same static source port of 1041.

A similar pattern can be seen in the analysis complied by **Shong Chong** (Analyst ID 283; http://www.sans.org/y2k/practical/Shong_Chong_GCIA.doc):

```
09/06-23:10:10.012419 [**] SUNRPC highport access! [**] 193.64.205.17:56880->
My.Net.211.2:32771
09/06-23:10:10.159763 [**] SUNRPC highport access! [**] 193.64.205.17:56880->
My.Net.211.2:32771
09/06-23:10:10.302667 [**] SUNRPC highport access! [**] 193.64.205.17:56880->
My.Net.211.2:32771
```

And here is Shong Chong's analysis of this behavior:

> *Looks like this server has been probed for port 32771 quite a bit. Port 32771 is SUN RPC high port. It is usually reserved for use inside a LAN. It could be one of the following 4 activities, but there are no corresponding detail log available for these alerts.*
>
> *IDS26/nfs-showmount [TCP any -> 32771:] CAN-1999-0631*
> *IDS429/portmap-listing-32771 [TCP any -> 32771] CAN-1999-0632*
> *IDS241/rpc.ttdbserv-solaris-kill [TCP any -> 32771:34000] CVE-1999-0003*
> *IDS242/rpc.ttdbserv-solaris-overflow [TCP any -> 32771:34000] CVE-1999-0003*

Security Recommendation

All traffic to and from the My.Net.x.x network should be controlled at the firewalls. In particular, RPC ports should not be accessible externally, since RPC exploits are one of the most popular means of attacks.

Port 55850 TCP - Possible MyServer Activity

These alerts detect connections to port 55850, which is known to be used by the MyServer DDOS agent.

Top ten internal source IP:

IP	Occurrences
MY.NET.253.24	48
MY.NET.217.154	26
MY.NET.5.29	17
MY.NET.253.41	16
MY.NET.253.52	12
MY.NET.100.230	11
MY.NET.6.34	9
MY.NET.70.97	3
MY.NET.139.36	3

MY.NET.1.10	3

Top ten external source IP:

IP	Occurrences
199.4.19.2	27
128.42.5.4	24
208.33.217.101	16
128.100.132.4	15
205.188.156.249	12
171.64.14.58	7
63.90.54.167	5
152.163.225.103	4
4.18.92.27	3
192.87.16.130	3

Below is an extract from the alert logs of 3 July 2001

```
07/03-10:15:35.414324  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:35.414431  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:35.414534  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:35.418126  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:35.418189  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:35.498774  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:45.764710  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:45.767496  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:45.767898  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:45.767996  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:45.772007  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:45.775604  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:45.776092  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:45.777829  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:45.779892  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:45.838338  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:45.838388  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:45.878184  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:45.883122  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:45.889417  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:45.889463  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
07/03-10:15:45.889514  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
```

```
07/03-10:15:45.889604  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.52:55850 -> 205.188.156.249:25
07/03-10:15:45.890392  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 205.188.156.249:25 -> MY.NET.253.52:55850
```

This may be a false positive, since this looks very much like normal SMTP traffic.

Below is an extract from the alert logs of 3 July 2001. The host My.Net.243.41 may be home to the MyServer DDOS agent, and it is attacking the external IP 206.117.161.71.

```
07/03-00:08:32.796276  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] MY.NET.253.41:55850 -> 206.117.161.71:113
```

Below is an extract from the alert logs of 3 July 2001. The internal host My.Net.6.47 may be under attack from the external IP 152.163.225.103.

```
07/03-23:47:32.265719  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 152.163.225.103:55850 -> MY.NET.6.47:25
07/03-23:47:32.281884  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 152.163.225.103:55850 -> MY.NET.6.47:25
07/03-23:47:37.311952  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 152.163.225.103:55850 -> MY.NET.6.47:25
07/03-23:47:37.312152  [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1
[**] 152.163.225.103:55850 -> MY.NET.6.47:25
```

Security Recommendation

Scan all internal machines for opened 55850 ports, those that comes back positive should be investigated further.

Also block port 55850 at the firewalls. In general, if a port is not required, then disable it.

NMAP TCP Ping

This event indicates that a remote user has used the NMAP port-scanning tool to probe the server. An NMAP TCP ping (instead of ICMP echo) was sent to determine if a host is reachable or not.

Top ten source IP:

IP	Occurrences
207.238.101.253	39
204.167.220.253	31
202.187.24.3	14
199.197.130.21	6
208.9.199.244	4
63.117.235.7	3
211.152.3.40	3
193.144.127.9	3
193.41.181.254	2
212.150.43.130	2

Top ten destination ports:

Port	Description	Occurrences
53	DNS	74
80	HTTP	26
1095	NICELink/RAT Trojan	2
1130		2
1465	Pipe Platform	2
6346	Gnutella	1
21	FTP	1
1387	Computer Aided Design Software Inc	1
3946		1
34022		1

Looking at IP 207.238.101.253, which is registered to Business Internet, Inc in the US, it performed the most NMAP scans over the seven days:

Extracted from alert log of 7 July 2001

```
07/07-12:18:36.350293  [**] NMAP TCP ping! [**] 207.238.101.253:80 -> MY.NET.1.8:53
07/07-12:18:36.351430  [**] NMAP TCP ping! [**] 207.238.101.253:53 -> MY.NET.1.8:53
07/07-12:18:46.377051  [**] NMAP TCP ping! [**] 207.238.101.253:80 -> MY.NET.1.8:53
07/07-12:18:46.377149  [**] NMAP TCP ping! [**] 207.238.101.253:53 -> MY.NET.1.8:53
```

Extracted from alert log of 5 July 2001

```
07/05-02:26:21.257451  [**] NMAP TCP ping! [**] 207.238.101.253:80 -> MY.NET.1.8:53
07/05-02:26:21.258263  [**] NMAP TCP ping! [**] 207.238.101.253:53 -> MY.NET.1.8:53
```

Extracted from alert log 2 July 2001

```
07/02-13:17:48.752986  [**] NMAP TCP ping! [**] 207.238.101.253:443 ->
MY.NET.153.191:1095
```

The host 207.238.101.253 also makes appearances in the other days, each time always using port 443, port 53 or port 80 as its source port and either port 53, port 1165 or port 1465 as its destination port.

But despite the amount of traffic, this may be a false positive. **PJ Goodwin** (Analyst ID 305; http://www.sans.org/y2k/practical/PJ_Goodwin_GCIA.doc) also observed a high volume of NMAP TCP Ping alerts in his logs, but after further investigation, he gave this verdict:

> *Due to the evidence from the correlation, it appears that the majority of the NMAP TCP ping! alerts were generated by DNS load balancing.*

Security Recommendation

Outbound ICMP unreachable messages should be blocked at the routers to keep from providing too much information to remote systems.

Null Scan

Null scans are where there are no flags set at all on the TCP packets. These are obviously crafted packets, since null flags are not normal TCP traffic. Null scans are used to evade packet filters and firewalls that may be watching for SYN packets directed toward restricted ports. The scan should return a RST for closed ports, whereas open ports should drop the packet.

Because of this, it is also possible to use null scans for fingerprinting operating systems, since they only work with non-Microsoft systems.

Top ten source IP:

IP	Occurrences
61.116.124.245	10
213.66.109.65	6
202.92.71.141	3
202.92.71.208	3
65.26.36.91	3
202.92.68.165	2
202.92.70.253	2
202.92.71.186	2
24.160.115.104	2
62.252.43.61	2

Top ten destination port:

IP	Occurrences
1214	70
40195	11
6346	11
4063	3
1107	2
1446	1
2446	1
3605	1
6699	1
4159	1

The IP 61.116.124.245 resolves to the hostname n56ch-01p245.ppp11.odn.ad.jp, which is registered to some organization in Japan.

Extracted from scan log of 7 July 2001

```
Jul  7 16:03:53 61.116.124.245:1992 -> MY.NET.218.214:40195 NULL ********
```

Extracted from alert log of 7 July 2001

```
07/07-16:08:50.884706  [**] Null scan! [**] 61.116.124.245:1992 -> MY.NET.218.214:40195
07/07-16:19:43.596148  [**] Null scan! [**] 61.116.124.245:1992 -> MY.NET.218.214:40195
```

Null scans are intentional probes and this is obvious from the extracts, where the source port is static, even though the attempts are 11 minutes apart. It is a high port to a high port and the flags are all null.

John Garris (Analyst ID 330; http://www.sans.org/y2k/practical/John_Garris_GCIA.doc) also observed a number of null scans.

> ***MY.NET.253.105***: *As noted in the first table above, those machines which were most active during the last two analyses, have shown little change that should trigger a great deal of concern. During this period, MY.NET.253.105 received a number of null scans from 216.51.104.65.*

Security Recommendations

Firewall rules sets should be examined and all unnecessary traffic to and from the My.Net.x.x network should be disallowed.

High Port 65535 UDP/TCP - Possible Red Worm

The Red Worm, a variant of the Ramen and Lion worm, is also known as the Adore worm. The Adore worm scans the Internet checking Linux hosts to determine whether they are vulnerable to any of the following well-known exploits: LPRng, rpc-statd, wu-ftpd and BIND. The Adore worm replaces one system binary (ps) with a trojaned version, then sends an email to the following addresses: adore9000@21cn.com, adore9000@sina.com, adore9001@21cn.com, adore9001@sina.com.

The Adore worm sends some sensitive information about the system in the email, and then runs a package called icmp, which will set the default port to listen on, and the packet length to watch for. Once it receives the information, it then sets a rootshell to allow connections, it adds a cronjob and then removes all traces of its existence and reboots the system. But the system is now compromised, with an open port listening for connections.

External source IP:

IP	Occurrences
211.7.36.113	9
216.136.129.16	7
64.50.191.56	5
207.200.6.75	2
207.224.211.188	1
207.69.200.57	1
216.34.232.64	1
62.23.138.20	1

Internal source IP:

IP	Occurrences
MY.NET.60.38	31
MY.NET.5.29	13
MY.NET.253.51	11
MY.NET.100.230	7
MY.NET.6.47	1

Looking at the internal host My.Net.60.38, it makes several appearances on 6 July 2001, each time displaying the following pattern:

Extracted from alert log of 6 July 2001

```
07/06-12:49:54.064926  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.60.38:23 -> 144.243.4.2:65535
07/06-12:49:54.753894  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.60.38:23 -> 144.243.4.2:65535
07/06-12:49:54.949023  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.60.38:23 -> 144.243.4.2:65535
07/06-12:49:55.001991  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.60.38:23 -> 144.243.4.2:65535
07/06-12:49:55.120810  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.60.38:23 -> 144.243.4.2:65535
07/06-12:49:55.361549  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.60.38:23 -> 144.243.4.2:65535
07/06-12:49:55.398871  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.60.38:23 -> 144.243.4.2:65535
07/06-12:49:55.693922  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.60.38:23 -> 144.243.4.2:65535
07/06-12:49:55.804505  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.60.38:23 -> 144.243.4.2:65535
```

The host My.Net.60.38 attempts repeatingly to connect to IP 144.243.4.2, using source port 23 (telnet).

Security Recommendation

Scan all internal machines for opened 65535 ports, those that comes back positive should be investigated further.

Also block port 65535 at the firewalls. In general, if a port is not required, then disable it.

Russia Dynamo - SANS Flash 28-Jul-00

SANS recommended that traffic to or from the Russian IP range 194.87.6.x be blocked in a Flash Report on 28 July 2000 (http://archives.neohapsis.com/archives/sans/2000/0068.html). This was due to unusual activity consisting of Internet wide port scanning for proxy servers, with the information being sent back to a Russian IP address.

There were 64 occurrences of this alert type in the logs, showing communication between 3 My.Net.x.x hosts and the external IP 194.87.6.129.

External source IP:

IP	Occurrences
194.87.6.129	24
194.87.6.131	4

Internal source IP:

IP	Occurrences
MY.NET.150.133	15
MY.NET.70.97	15
MY.NET.75.145	6

Top ten destination port:

Port	Occurrences
1214	28
1376	6
1097	5
1784	3
2946	3
2938	3
1582	3
2878	3
1988	3
2746	2

Extracted from alert logs

```
07/05-10:22:01.195356  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.70.97:1214 -
> 194.87.6.131:2437
07/05-10:56:23.413446  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] 194.87.6.131:2878 -
> MY.NET.75.145:1214
07/05-10:56:23.414662  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.75.145:1214
-> 194.87.6.131:2878
07/05-10:56:24.218847  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] 194.87.6.131:2878 -
> MY.NET.75.145:1214
07/05-10:56:24.218894  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.75.145:1214
-> 194.87.6.131:2878
07/05-10:56:25.052683  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] 194.87.6.131:2878 -
> MY.NET.75.145:1214
07/05-10:56:25.052733  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.75.145:1214
-> 194.87.6.131:2878
07/05-11:12:01.686264  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.75.145:1214
-> 194.87.6.131:2938
07/05-11:12:02.215302  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.70.97:1214 -
> 194.87.6.131:2946
07/05-11:12:02.700465  [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.75.145:1214
-> 194.87.6.131:2938
```

This may just be a false positive, since there is no indication of any malicious activity here. If scanning was taking place, then more hosts would be targeted and the pattern would be more uniform.

Security Recommendation

If IP ranges from Russia is of concern, then these ranges can be blocked at the firewall. The 3 internal hosts should be investigated to determine why they are communicating with the 2 hosts in Russia.

Back Orifice

These alerts detect probes for the Back Orifice trojan program, which is a remote administration tool used to control Windows machines. The server listens on port 31337 and these probes are to locate possible compromised hosts on the internal network.

Back Orifice is a tool consisting of two main pieces, a client application and a server application. The client application, running on one machine, can be used to monitor and control a second machine running the server application

There were only 3 alerts generated for Back Orifice. The culprit is the external host 207.41.14.111, scanning internal host My.Net.60.39.

```
07/03-14:31:32.863038  [**] Back Orifice [**] 207.41.14.11:1765 -> MY.NET.60.39:
31337
07/03-14:31:59.231822  [**] Back Orifice [**] 207.41.14.11:1755 -> MY.NET.60.39:
31337
07/03-15:01:42.643866  [**] Back Orifice [**] 207.41.14.11:3697 -> MY.NET.60.39:
31337
```

There is no resolved hostname belonging to this IP, but the IP block 207.40.0.0 – 207.43.255.255 is registered to Sprint, which is based in the US. It seemed 207.41.14.111 attempted to connect twice in succession, then once more about thirty minutes later. This brings me to conclude that this is an intended attack, performed manually. My guess is the host 207.41.14.111 has also been compromised and is being used to attempt connection to My.Net.60.39 by the attacker. May be the attacker is verifying that the Back Orifice trojan have been successfully delivered to My.Net.60.39.

Security Recommendation

The host My.Net.60.39 should be taken offline and investigated further to ensure it has not been compromised.

Generally all traffic to and from the My.Net.x.x network should be monitored and properly controlled.

TCP SMTP Source Port Traffic

This event indicates that an attacker is making a connection to a privileged port using the source port 25 (smtp). This should not normally occur. Old or mis-configured packet filters may allow the connection if they allow all SMTP traffic.

This alert is similar to the TCP FTP-Data Source Port (http://www.whitehate.com/info/IDS6) and TCP DNS Source Port (http://www.whitehats.com/info/IDS7).

There were 3 alerts, triggered by 2 external hosts:

External IP	Destination Host	Destination Port	Occurrences
129.43.100.100	My.Net.253.52	583	2
207.88.135.158	My.Net.5.73	807	1

```
07/03-11:16:43.255384  [**] TCP SMTP Source Port traffic [**] 207.88.135.158:25
-> MY.NET.5.73:807
07/05-08:45:43.767416  [**] TCP SMTP Source Port traffic [**] 129.43.100.100:25
-> MY.NET.253.52:583
07/05-08:47:04.770142  [**] TCP SMTP Source Port traffic [**] 129.43.100.100:25
-> MY.NET.253.52:583
```

There is no DNS information for neither external IPs.

IP 207.88.135.158 only makes one single attempt, but it may have made other attempts earlier and logs prior to 2 July 2001 will need to be examined to confirm this. Port 807 is not a known port, and neither is it high enough to be discarded as the end of some SMTP communication.

As for IP 129.43.100.100, this makes 2 attempts in the space of 2 minutes. The time is too slow for this to be automated, hence it must be conducted manually and since the same destination host is used both times, this lead to me believe this probe was intentional.

Security Recommendation

Incoming traffic with port 25 should be blocked at the firewall. Packets should not be using a source port of 25, unless it is in response to a SMTP request.

In general, traffic should always be initiated internally and not externally and the firewall should block any traffic originating from external sources.

Also, My.Net.253.52 and My.Net.5.73 may need to be taken offline and investigated further to determine why they are of interest.

3.2.2 Top Talkers

This section lists the most occurring IPs and ports in the alert logs. From these list, we can see what external hosts should be paid further attention to, and what internal hosts requires further investigations.

*The data below was obtained from all alert logs, but **excluding** port scans.*

Top ten external source IP:

IP	Occurrences
63.250.213.73	25821
213.255.24.48	8247

63.250.213.124	2331
169.254.161.0	2083
63.250.213.26	1788
212.179.34.114	573
159.226.41.166	525
165.132.31.137	432
199.84.54.32	311
24.159.128.162	306

Top ten internal source IP:

IP	Occurrences
MY.NET.253.24	48
MY.NET.60.38	32
MY.NET.5.29	32
MY.NET.217.154	26
MY.NET.100.230	20
MY.NET.70.97	18
MY.NET.253.41	16
MY.NET.150.133	15
MY.NET.253.52	12
MY.NET.253.51	11
MY.NET.6.34	9

Top ten internal destination IP:

IP	Occurrences
MY.NET.70.97	863
MY.NET.150.133	670
MY.NET.218.234	527
MY.NET.100.37	526
MY.NET.150.225	346
MY.NET.150.143	269
MY.NET.217.6	244
MY.NET.97.165	175
MY.NET.253.43	121
MY.NET.253.42	116

Top ten external destination IP:

IP	Occurrences
233.28.65.227	25821
233.40.70.50	2331
233.28.65.164	1788
130.132.143.42	1067
130.132.143.43	1016
233.28.65.61	173
167.102.7.105	161
162.129.20.10	141

213.243.141.126	83
64.231.73.233	74

3.2.3 Interesting External Hosts

This section lists the top ten external IP that has initiated a connection to a known trojan port over the seven day period.

This list of IP is all associated with the "Possible Trojan Server Activity" alerts, shown in the alert files.

There were 2,367 alerts generated from connections made from internally to external sites, and 13,122 alerts showing connections made from externally to internal sites.

External IP	**Occurrences**
24.159.128.162	306
24.88.85.106	222
24.78.182.153	200
24.157.8.115	198
24.249.206.23	176
63.10.156.249	139
192.117.130.89	134
65.92.117.50	132
24.76.182.61	126
206.74.76.44	123

Host 1

IP: 24.159.128.162
Resolved Name: ip-128-162.charterpa.com
DNS Information:

Domain Name: CHARTERPA.COM
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: NS.HHS.NET
Name Server: NS.WESTOL.COM
Updated Date: 01-feb-2001

Registrant:
Charter Communications (CHARTERPA-DOM)
120 Southmont Blvd
Johnstown, PA 15905
US

Domain Name: CHARTERPA.COM

Administrative Contact, Billing Contact:
Domain Adminstrator (DA24052-OR) admrole@CHARTERPA.COM
Charter Communications
120 Southmont Blvd
Johnstown , PA 15905
US
814-539-8971
Fax- 814-535-7749

Technical Contact:
Technical Role Account (TR911-ORG) dnsadmin@CHARTERPA.NET
Charter Online
302 West Otterman Street
Greensburg, PA 15601
USA
724-219-0400
Fax- 724-853-0361

Record last updated on 01-Feb-2001.
Record expires on 01-Feb-2002.
Record created on 01-Feb-2000.

Database last updated on 5-Aug-2001 23:52:00 EDT.

Domain servers in listed order:
NS.WESTOL.COM63.93.137.4
NS.HHS.NET63.93.136.29

Example of activity:

```
Jul  2 19:50:26 24.159.128.162:3659 -> MY.NET.111.112:27374 SYN **S*****
Jul  2 19:50:26 24.159.128.162:3657 -> MY.NET.111.110:27374 SYN **S*****
Jul  2 19:50:26 24.159.128.162:3653 -> MY.NET.111.106:27374 SYN **S*****
Jul  2 19:50:26 24.159.128.162:3665 -> MY.NET.111.118:27374 SYN **S*****
Jul  2 19:50:26 24.159.128.162:3667 -> MY.NET.111.120:27374 SYN **S*****
Jul  2 19:50:26 24.159.128.162:3669 -> MY.NET.111.122:27374 SYN **S*****
Jul  2 19:50:29 24.159.128.162:3718 -> MY.NET.111.173:27374 SYN **S*****
Jul  2 19:50:28 24.159.128.162:3719 -> MY.NET.111.174:27374 SYN **S*****
Jul  2 19:50:28 24.159.128.162:3722 -> MY.NET.111.177:27374 SYN **S*****
Jul  2 19:50:28 24.159.128.162:3724 -> MY.NET.111.179:27374 SYN **S*****
Jul  2 19:50:29 24.159.128.162:3702 -> MY.NET.111.157:27374 SYN **S*****
Jul  2 19:50:29 24.159.128.162:3706 -> MY.NET.111.161:27374 SYN **S*****
Jul  2 19:50:29 24.159.128.162:3705 -> MY.NET.111.160:27374 SYN **S*****
Jul  2 19:50:29 24.159.128.162:3713 -> MY.NET.111.168:27374 SYN **S*****
Jul  2 19:50:29 24.159.128.162:3711 -> MY.NET.111.166:27374 SYN **S*****
```

Host 2

IP: 24.88.85.106
Resolved Name: cae88-85-106.sc.rr.com
DNS Information:

Domain Name: SC-RR.COM
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: MB1.MICROBYTE.NET
Name Server: NS1.ESPIRE.NET
Updated Date: 24-jun-2001

Registrant:
Time Warner Cable (SC-RR-DOM)
293 Greystone Blvd
Columbia, SC 29210

Domain Name: SC-RR.COM

Administrative Contact, Technical Contact, Billing Contact:
Almassri, Maher (MA1855) webmaster@MICROBYTE.NET
MicroByte Net
1410 Colonial Life Boulevard
Columbia, SC 29210
(803) 750-7500 (FAX) (803) 750 - 0174

Record last updated on 22-Jun-1999.
Record expires on 22-Jun-2001.
Record created on 22-Jun-1999.

Database last updated on 5-Aug-2001 23:52:00 EDT.

Domain servers in listed order:
MB1.MICROBYTE.NET 207.201.197.10
NS1.ESPIRE.NET 206.222.97.82

Example of activity:

```
Jul  2 19:49:51 24.88.85.106:1197 -> MY.NET.158.46:27374 SYN **S*****
Jul  2 19:49:51 24.88.85.106:1184 -> MY.NET.158.33:27374 SYN **S*****
Jul  2 19:49:51 24.88.85.106:1198 -> MY.NET.158.47:27374 SYN **S*****
```

Host 3

IP: 24.78.182.153
Resolved Name: No host name is associated with this IP address or no reverse lookup is configured
DNS Information:

Domain Name: SHAWCABLE.NET
Registrar: NETWORK SOLUTIONS, INC.

Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: NS1SO.CG.SHAWCABLE.NET
Name Server: NS2SO.CG.SHAWCABLE.NET
Updated Date: 21-apr-2000

Registrant:
Shaw Cablesystems G.P. (SHAWCABLE7-DOM)
Suite 900 630-3rd Avenue S.W.
Calgary, AB T2P 4L4
CA

Domain Name: SHAWCABLE.NET

Administrative Contact, Billing Contact:
Thierman, Chris (CT11) Chris.Thierman@SHAW.CA
Shaw Cable Systems
Calgary, Alberta
Calgary
AB
T2P 4L4
CA
(403) 750 6991 (FAX) (403) 750 4504

Technical Contact:
Shaw Cable,Internet Engineering (SC5338-ORG) internet.engineering@SHAW.CA
Shaw Cablesystems G.P.
630 - 3rd Avenue S.W.
Calgary, AB T2P 4L4
CA
(403)750-4500 Fax- (403)750-4504
Fax- internet.abuse@SHAW.CA

Record last updated on 14-Aug-2000.
Record expires on 05-Nov-2001.
Record created on 05-Nov-1999.

Database last updated on 5-Aug-2001 23:52:00 EDT.

Domain servers in listed order:
NS1SO.CG.SHAWCABLE.NET 24.64.63.195
NS2SO.CG.SHAWCABLE.NET 24.64.63.212

Example of activity:

```
Jul  2 19:52:35 24.78.182.153:2335 -> MY.NET.106.238:27374 SYN **S*****
Jul  2 19:52:35 24.78.182.153:2365 -> MY.NET.107.11:27374 SYN **S*****
```

```
Jul  2 19:52:35 24.78.182.153:2367 -> MY.NET.107.13:27374 SYN **S*****
Jul  2 19:52:35 24.78.182.153:2371 -> MY.NET.107.17:27374 SYN **S*****
Jul  2 19:52:35 24.78.182.153:2373 -> MY.NET.107.19:27374 SYN **S*****
Jul  2 19:52:35 24.78.182.153:2375 -> MY.NET.107.21:27374 SYN **S*****
Jul  2 19:52:35 24.78.182.153:2379 -> MY.NET.107.25:27374 SYN **S*****
```

Host 4

IP: 24.157.8.115
Resolved Name: 24.157.8.115.on.wave.home.com
DNS Information:

Domain Name: HOME.COM
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: NS3.HOME.NET
Name Server: NS4.HOME.NET
Name Server: NS5.HOME.NET
Name Server: NS6.HOME.NET
Updated Date: 13-Jun-2001

Registrant:
Home Network (HOME-DOM)
425 Broadway St.
Redwood City, CA 94063
US

Domain Name: HOME.COM

Administrative Contact, Technical Contact:
DNS Administration (DA24627-OR) abuse@HOME.COM
@Home Network
425 Broadway St
Redwood City , CA 94063
US
650-556-5399
Fax- 650-556-6666

Billing Contact:
Du, Trung (TD2157) trung@CORP.HOME.NET
@Home Network
425 Broadway Street
Redwood City, CA 94063-3126
650-569-5437 (FAX) 650-569-5100

Record last updated on 15-Mar-2001.
Record expires on 17-Dec-2002.
Record created on 16-Dec-1993.

Database last updated on 5-Aug-2001 23:52:00 EDT.

Domain servers in listed order:
NS3.HOME.NET24.0.95.250
NS4.HOME.NET24.14.77.13
NS5.HOME.NET24.0.95.252
NS6.HOME.NET24.14.77.14

Example of activity:

```
Jul  2 19:51:25 24.157.8.115:4260 -> MY.NET.27.61:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4254 -> MY.NET.27.55:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4270 -> MY.NET.27.71:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4264 -> MY.NET.27.65:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4258 -> MY.NET.27.59:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4268 -> MY.NET.27.69:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4252 -> MY.NET.27.53:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4262 -> MY.NET.27.63:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4256 -> MY.NET.27.57:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4266 -> MY.NET.27.67:27374 SYN **S*****
Jul  2 19:51:25 24.157.8.115:4250 -> MY.NET.27.51:27374 SYN **S*****
```

Host 5

IP: 24.249.206.23
Resolved Name: c1622621-b.ross1.pa.home.com
DNS Information: *See Host 4*

Example of activity:

```
Jul  2 19:51:32 24.249.206.23:2015 -> MY.NET.146.140:27374 SYN **S*****
Jul  2 19:51:32 24.249.206.23:2021 -> MY.NET.146.146:27374 SYN **S*****
Jul  2 19:51:32 24.249.206.23:2023 -> MY.NET.146.148:27374 SYN **S*****
Jul  2 19:51:35 24.249.206.23:2026 -> MY.NET.146.151:27374 SYN **S*****
Jul  2 19:51:32 24.249.206.23:2028 -> MY.NET.146.153:27374 SYN **S*****
Jul  2 19:51:32 24.249.206.23:2030 -> MY.NET.146.155:27374 SYN **S*****
Jul  2 19:51:32 24.249.206.23:2032 -> MY.NET.146.157:27374 SYN **S*****
Jul  2 19:51:32 24.249.206.23:2034 -> MY.NET.146.159:27374 SYN **S*****
Jul  2 19:51:32 24.249.206.23:2038 -> MY.NET.146.163:27374 SYN **S*****
```

Host 6

IP: 63.10.156.249
Resolved Name: 1cust249.tnt1.lafayette.la.da.uu.net
DNS Information:

Domain Name: UU.NET
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: AUTH00.NS.UU.NET
Name Server: AUTH60.NS.UU.NET
Updated Date: 13-jul-2001

Registrant:
UUNET Technologies, Inc. (UU-DOM)
3060 Williams Drive Ste 601
Fairfax, VA 22031
USA

Domain Name: UU.NET

Administrative Contact, Technical Contact:
UUNET, AlterNet - Technical Support (OA12) help@UU.NET
3060 Williams Drive
Fairfax, VA 22031
+1 (800) 900-0241

Billing Contact:
UUNET Technologies, Inc. (PA10-ORG) help@UU.NET
22001 Loudoun County Parkway
Ashburn, VA 20147
US
+1 (800)900-0241
Fax- .: (703) 206-5601

Record last updated on 13-Jul-2001.
Record expires on 21-May-2002.
Record created on 20-May-1987.

Database last updated on 5-Aug-2001 23:52:00 EDT.

Domain servers in listed order:
AUTH00.NS.UU.NET198.6.1.65
AUTH60.NS.UU.NET198.6.1.181

Example of activity:

```
Jul  2 19:49:55 63.10.156.249:1121 -> MY.NET.253.212:27374 SYN **S*****
Jul  2 19:49:55 63.10.156.249:1123 -> MY.NET.253.214:27374 SYN **S*****
Jul  2 19:49:55 63.10.156.249:1125 -> MY.NET.253.216:27374 SYN **S*****
Jul  2 19:49:55 63.10.156.249:1127 -> MY.NET.253.218:27374 SYN **S*****
```

Host 7

IP: 192.117.130.89
Resolved Name: SHARON
DNS Information:

European Regional Internet Registry/RIPE NCC (NET-IL-ISOC-RIPE)

These addresses have been further assigned to European users. Contact information be found in the RIPE database, whois.ripe.net NL

Netname: IL-ISOC-RIPE
Netblock: 192.114.0.0 - 192.118.255.255
Maintainer: RIPE

Coordinator: Reseaux IP European Network Co-ordination Centre Singel 258 (RIPE-NCC-ARIN) nicdb@RIPE.NET +31 20 535 4444

Domain System inverse mapping provided by:

RELAY.HUJI.AC.IL	128.139.6.1
LOOKUP.IUCC.AC.IL	128.139.34.240
NS.RIPE.NET	193.0.0.193
DNSAUTH1.SYS.GTEI.NET	4.2.49.2
DNSAUTH2.SYS.GTEI.NET	4.2.49.3
DNSAUTH3.SYS.GTEI.NET	4.2.49.4

Record last updated on 07-Jun-2001.
Database last updated on 1-Aug-2001 23:18:04 EDT.

Example of activity:

```
Jul  2 19:49:45 192.117.130.89:2479 -> MY.NET.215.196:27374 SYN **S*****
Jul  2 19:49:45 192.117.130.89:2481 -> MY.NET.215.198:27374 SYN **S*****
Jul  2 19:49:45 192.117.130.89:2485 -> MY.NET.215.202:27374 SYN **S*****
Jul  2 19:49:45 192.117.130.89:2491 -> MY.NET.215.208:27374 SYN **S*****
Jul  2 19:49:45 192.117.130.89:2489 -> MY.NET.215.206:27374 SYN **S*****
Jul  2 19:49:45 192.117.130.89:2498 -> MY.NET.215.215:27374 SYN **S*****
Jul  2 19:49:45 192.117.130.89:2493 -> MY.NET.215.210:27374 SYN **S*****
Jul  2 19:49:45 192.117.130.89:2502 -> MY.NET.215.219:27374 SYN **S*****
```

Host 8

IP: 65.92.117.50
Resolved Name: hse-toronto-ppp3489753.sympatico.ca
DNS Information:

Status: EXIST
Registrar: Webnames.ca (UBC Research Enterprises Inc.)
Registrar-no: 70
Registrant-no: 7336
Domaine-no: 7336
Subdomain: sympatico.ca
Date-Approved: 2000/10/02
Date-Modified: 2001/05/03
Organization: Bell ActiMedia Inc.
Description: Bell Canada provides telecom and internet working solutions.

Admin-Name: Pascale Mercier
Admin-Postal: Montreal QC H3B 5H8 Canada
Admin-Phone: 514-870-6564
Admin-Fax: 514-870-4833
Admin-Mailbox: trademarks@bell.ca
Tech-Name: Sandie Riff
Tech-Title: Operations & Technology Solutions
Tech-Postal: Sympatico (TM), Bell Canada
Sympatico (TM), Bell Canada
160 Elgin 12
Ottawa ON K1G 3J4 Canada
Tech-Phone: +1 (800) 565-0567
Tech-Fax: +1 613-339-1805
Tech-Mailbox: dns-admin@bellglobal.com
NS1-Hostname: dns1.sympatico.ca
NS1-Netaddress: 204.101.251.1
NS2-Hostname: dns2.sympatico.ca
NS2-Netaddress: 204.101.251.2
NS3-Hostname: ns5.bellnexxia.net
NS3-Netaddress: 209.226.175.236
NS4-Hostname: ns6.bellnexxia.net
NS4-Netaddress: 209.226.175.237

Example of activity:

```
Jul  2 19:51:28 65.92.117.50:1578 -> MY.NET.206.176:27374 SYN **S*****
Jul  2 19:51:28 65.92.117.50:1576 -> MY.NET.206.174:27374 SYN **S*****
Jul  2 19:51:28 65.92.117.50:1586 -> MY.NET.206.184:27374 SYN **S*****
Jul  2 19:51:28 65.92.117.50:1584 -> MY.NET.206.182:27374 SYN **S*****
```

Host 9

IP: 24.76.182.61
Resolved Name: No host name is associated with this IP address or no reverse lookup is configured
DNS Information:

Shaw Fiberlink (aka Shaw@HOME) (NETBLK-FIBERLINK-CABLE-2BLK)
630 - 3rd Ave SW
Calgary, AB 4L4
CA

Netname: FIBERLINK-CABLE-2BLK
Netblock: 24.76.0.0 - 24.79.255.255
Maintainer: FBCA

Coordinator:
Sachetti, Ron (RS1472-ARIN) ipadmin@shaw.ca

403-750-7428

Domain System inverse mapping provided by:
NS2SO.CG.SHAWCABLE.NET24.64.63.212
NS1SO.CG.SHAWCABLE.NET24.64.63.195

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

Record last updated on 26-Feb-2001.
Database last updated on 4-Aug-2001 23:01:41 EDT.

Example of activity:

```
Jul  2 19:51:12 24.76.182.61:4689 -> MY.NET.99.251:27374 SYN **S*****
Jul  2 19:51:12 24.76.182.61:4690 -> MY.NET.99.252:27374 SYN **S*****
```

Host 10

IP: 206.74.76.44
Resolved Name: dial-43.r1.scccrk-gwy.infoave.net
DNS Information:

Domain Name: INFOAVE.NET
Registrar: THE REGISTRY AT INFO AVENUE D/B/A IA REGISTRY
Whois Server: whois.iaregistry.com
Referral URL: http://www.iaregistry.com
Name Server: DNS4.INFOAVE.NET
Name Server: DNS2.INFOAVE.NET
Updated Date: 13-jun-2001

Registrant:
Info Avenue Internet Services, LLC
P.O. Box 698
Fort Mill, SC 29716
US
Registrar..: IARegistry.com (http://www.iaregistry.com)

Domain Name: INFOAVE.NET
Created on.....: 17-Aug-1995
Expires on.....: 16-Aug-2006
Record last updated on..: 02-Aug-2001

Administrative Contact:
Host, Master webadmin@infoave.net
Info Avenue Internet Services, LLC
P.O. Box 698
Fort Mill, SC 29716 US

803-802-4600 (FAX) 803-802-4700

Technical Contact, Zone Contact:
Site, Administrator hostmaster@infoave.net
Info Avenue Internet Services, LLC
P.O. Box 698
3555 Centre Circle Drive, Suite A
Fort Mill, SC 29716 US
803-802-4600 (FAX) 803-802-4700

Name servers for this domain:

DNS4.INFOAVE.NET	206.74.254.2
DNS2.INFOAVE.NET	165.166.0.3

Example of activity:

```
Jul  2 19:51:40 206.74.76.44:1415 -> MY.NET.154.25:27374 SYN **S*****
Jul  2 19:51:40 206.74.76.44:1417 -> MY.NET.154.27:27374 SYN **S*****
Jul  2 19:51:40 206.74.76.44:1423 -> MY.NET.154.33:27374 SYN **S*****
Jul  2 19:51:40 206.74.76.44:1421 -> MY.NET.154.31:27374 SYN **S*****
```

Summary

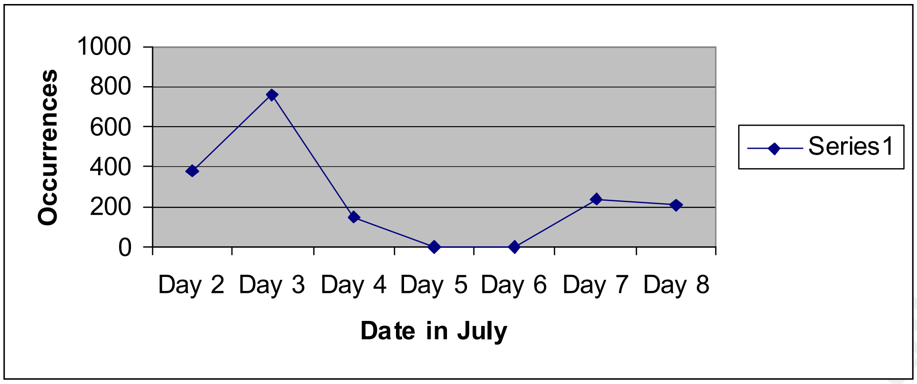
The above were all extracts from the scan logs of 2 July 2001. It seems that the majority of possible trojan activity occurred on 2 July 2001, with all external IP scanning for the SubSeven trojan.

I have not included all the logs here, but from the scan logs, it is clear that these ten hosts are using an automated script to perform the scans, since the scans are very frequent and close together, hence showing evidence that it cannot be performed manually. Also, all these ten hosts performed their scans around 7:50pm and on the same day, which brings me to conclude that they are related.

3.3 OOS Analysis

This section analyses the OOS (Out of Spec) files in further details.

The line graph below shows the level of activity over the week. The peak in the traffic was on the 3 July, but there were no occurrences of any OOS alert on the 5 July and the 6 July.



Internal Source

There was only one internal source host:

Extracted from OOS file of 8 July 2001

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/08-05:16:28.869667 MY.NET.217.134:0 -> 128.121.11.125:1270
TCP TTL:126 TOS:0x0 ID:5404  DF
2*SFRPA* Seq: 0x50000C   Ack: 0x2E935FA1   Win: 0x5010
22 38 CC E4 20 20 20 20 20 00                    "8..     .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

This host makes a single connection to target 128.121.11.125 on port 1270, using a crafted packet. From the above extract, it can be seen that the packet is using flags SYN + FIN + RST + PSH + ACK + a reserved bit. This is not normal, since it is not logical to have both a SYN and FIN in the same packet. Attackers who are aware that intrusion detection systems may be looking for packets with just the SYN and FIN bits set, not additional bits set may use these packets. Also, the host is using a source port of 0, which is reserved and a known port for the linuxportz attack tool.

The host My.Net.217.134 may be being used by a malicious student, or have been compromised by external attackers. In either case, it should be taken offline and investigated further.

External Source

Generally, all hosts are using crafted packets of various forms. Some sending packets with both the SYN and FIN (and any other flags) set, others with the reserve bits set.

Top ten TCP flags:

TCP Flag	Occurrences
21S*****	5208
SF**	3391
**SFRP*U	121
2*SF***U	71

2*SFRP*U	68
*1SF****	50
2*SFRP**	45
**SFRPA*	41
*1SFR***	38
SFA*	37

All the above flags are not normal with the exception of the first flag, and are generally used to by-pass IDS or to perform OS fingerprinting. The flag *21S***** denotes the usage of the ECN option that the IETF proposed. The alerts were triggered probably due to an old rule set, which has not been updated to recognize the new usage of the reserved bits.

The top five external 'top talkers' are:

IP	Occurrences
211.180.236.194	557
199.183.24.194	317
210.77.146.33	234
193.226.113.248	61
24.216.144.34	45

Further Analysis of the Top Five 'Top Talkers'

211.180.236.194

This host seems to be scanning for portmap, using SYN + FIN to by-pass the IDS. It seems to be at first only targeting hosts with an odd host number, then hosts with an even host number. It is also generally using a static sequence number (but changing to a new sequence number after about every 22 attempted connections).

This address is assigned to Asia-Pacific users.

Example of activity:

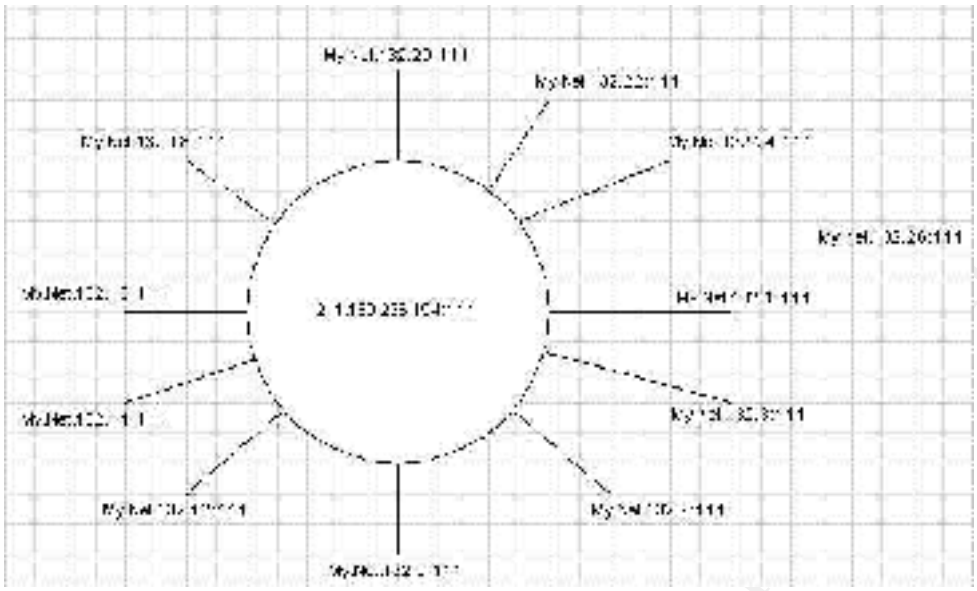
Extract from logs

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/03-13:10:29.120891 211.180.236.194:111 -> MY.NET.6.15:111
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x1F04031C   Ack: 0x2B81D62C   Win: 0x404
00 00 00 00 00 00                                  ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Link diagram

This diagram below shows the activities between the host 211.180.236.194 and various hosts on the My.Net.x.x network, during 13:21:11 (i.e. a 1-second scan) on the 3 July 2001.



199.183.24.194

This host has also generated a high volume of alerts in the alert logs, by triggering the Queso Fingerprinting alert.

This address is assigned to ICG NetAhead, Inc.

Example of activity:

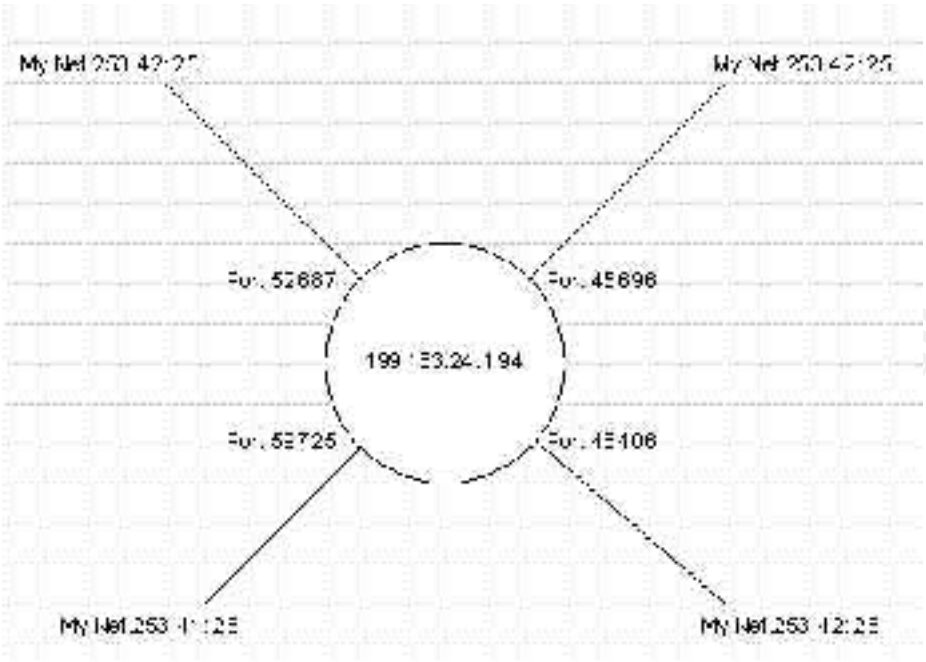
Extract from logs

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/03-13:41:13.184062 199.183.24.194:42913 -> MY.NET.253.41:25
TCP TTL:54 TOS:0x0 ID:28114  DF
21S***** Seq: 0x2068EC0B   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 366576089 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Link diagram

This diagram below shows the activities between the host 199.183.24.194 and hosts My.Net.253.41 and My.Net.253.42, between 14:47:44 to 15:17:39 (i.e. a period of 30 minutes) on the 3 July 2001.



210.77.146.33

This host scans MY.NET.253.114 repeatingly on port 80 (HTTP). Looking at the time of each scan, it is highly likely that this is performed by some automated script. Because the target port is port 80, this leads me to believe that this is the activity of a worm, may be one that is trying to exploit the Apache Web Server or IIS, such as the sadmind/IIS Worm or the Code Red Worm.

This address is assigned to Asia-Pacific users.

Example of activity:

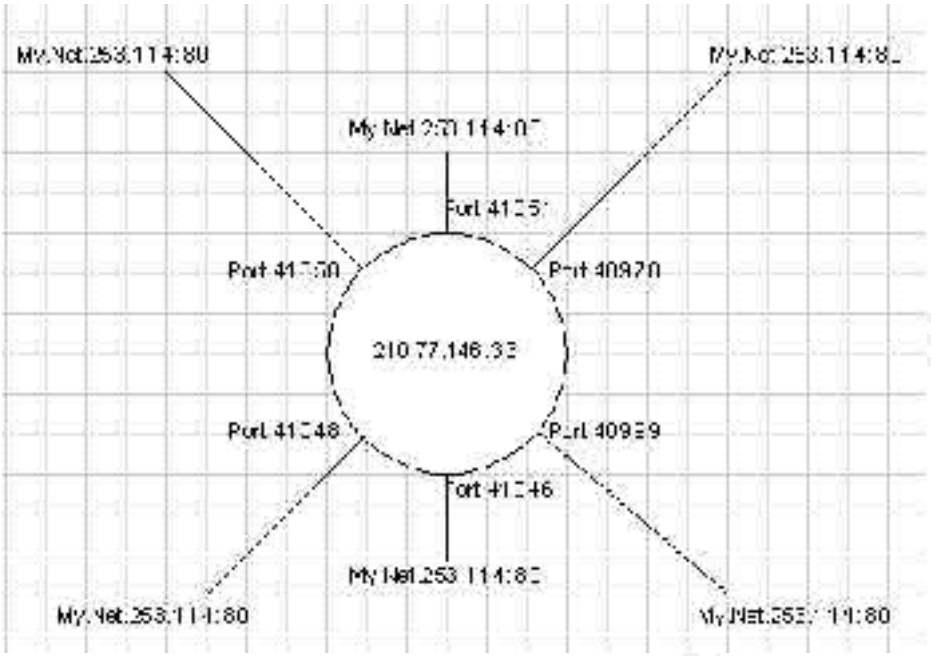
Extract from logs

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/04-23:20:01.972312 210.77.146.33:39636 -> MY.NET.100.165:80
TCP TTL:46 TOS:0x0 ID:18940  DF
21S***** Seq: 0x55F33CB4   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 3175752 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Link diagram

This diagram below shows the activities between the host 210.77.146.33 and the host My.Net.253.114 on port 80, between 03:25:50 to 03:25:53 (i.e. a period of 3-seconds) on the 2 July 2001.



193.226.113.248

This host has also generated a high volume of alerts in the alert logs, by triggering the Queso Fingerprinting alert.

The IP resolves to the hostname 248.valahia.ro, which is registered to State University "Valahia" in Romania.

Example of activity:

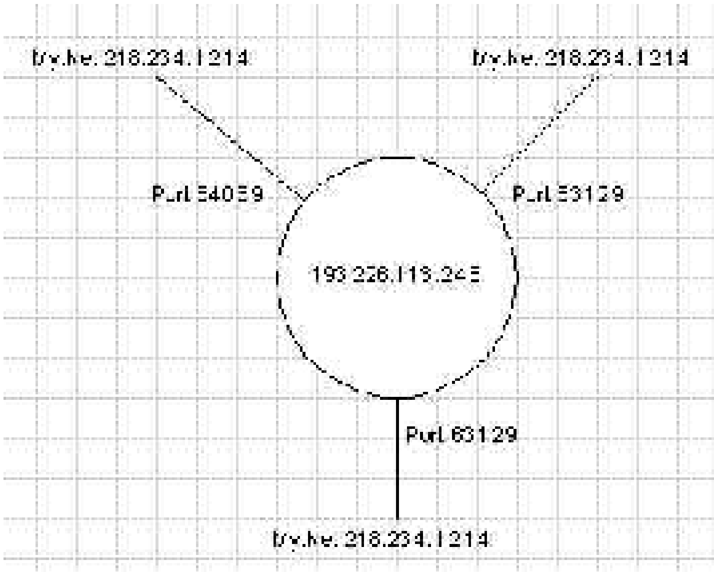
Extract from logs

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/07-11:39:06.560331 193.226.113.248:4944 -> MY.NET.70.97:1214
TCP TTL:46 TOS:0x0 ID:0  DF
21S***** Seq: 0x5CDA9940   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 45873795 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Link diagram

This diagram below shows the activities between the host 193.226.113.248 and the host My.Net.218.234 on port 1214, between 22:18:54 to 23:08:06 (i.e. a period of 50 minutes) on the 4 July 2001.



<u>24.216.144.34</u>

The behavior for this host is similar to that of 210.77.146.33, in that it is continuously scanning MY.NET.70.161 on port 80.

The IP resolves to the hostname 24-216-144-34.hsacorp.net, which is registered to HSA Corp in the US.

Example of activity:

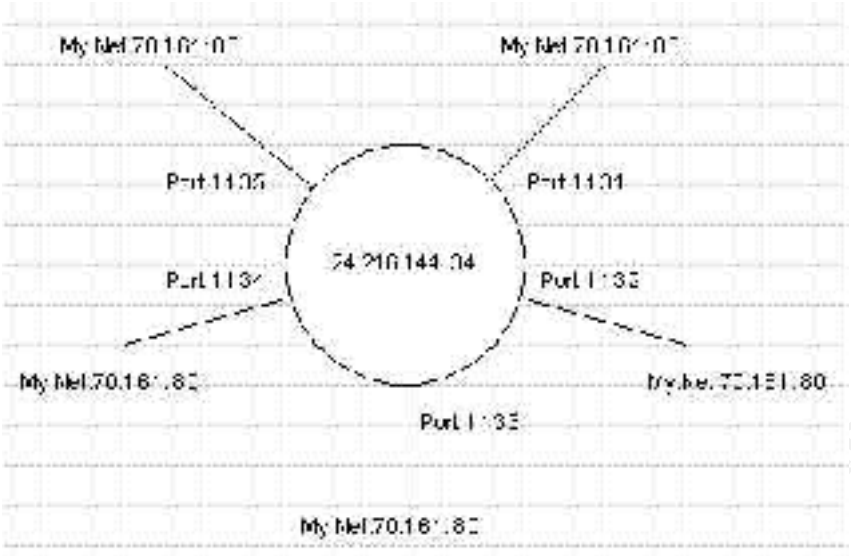
Extract from logs

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/08-05:48:06.522731 24.216.144.34:1131 -> MY.NET.70.161:80
TCP TTL:47 TOS:0x0 ID:8295  DF
21S***** Seq: 0x558FBC35   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 122592338 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Link diagram

This diagram below shows the activities between the host 24.216.144.34 and the host My.Net.70.161 on port 80, between 05:48:06 to 05:58:48 (i.e. a period of 10 minutes) on the 8 July 2001.



3.4 Executive Summary

From the logs for the week 2 Jul – 8 Jul 2001, there is a high volume of trojan activity. Actually scanning through the logs, all trojan activity is related to port 27374, which is the SubSeven trojan. Not surprisingly though since at that time (during June and early July), there were an increase in SubSeven activity. This was noted in various mailing lists, especially in the vuln-dev and incident mailing lists from http://www.securityfocus.com.

From the OOS log files, there are high volumes of what may be traffic utilising the ECN option. Because ECN is still relatively new, this may be why Snort alerts were triggered and OOS logged. But ECN may be used for fingerprinting as well, since not all systems currently support it.

One way to minimize false positive generated by ECN traffic is to update the rules set to recognize this new addition to TCP.

There is a lot of traffic that is initiated internally, which could indicate one (or more) of the following reasons:

- A internal user is using the network to perform scans
- A majority of internal hosts have been compromised
- The internal address space is being used for spoofing

Also, with the increasing popularity of tools such as Napster, there are many connections to these ports to both external and internal destinations. These traffic not only contributes towards the false positive category, but is also dangerous in their own right. File sharing is generally not good security practice and should be discouraged.

3.5 Description of Analysis Process

Tools that were used in the analysis process were:

- Microsoft Excel
- Perl 5.6
- Unix commands such as grep, cat, more, etc

3.5.1 Alerts

In order to analyse the alerts, the types of alerts were first collaborated into a table, by using a Perl script that was written especially for this task. All the alert files were concatenated into one file, which is passed to the script. The script loops through the file and collects each unique alert type into a table (a hash array).

The output is directed to another file, which is then imported into Microsoft Excel. Using the functions available within Excel, this file is manipulated such that the alerts are listed in descending order of occurrences. To aid the analysis process, a pie chart was also produced based on the data.

Once I had the lists of unique alert types, I further broke down the analysis of the alerts by using the Grep command in Unix, and grepping for all occurrences of a specific alert type. The output is directed to a file, which is passed through another Perl script similar to the one used to collect the occurrences of each alert type.

Example Grep command

```
#more alertalert.* | grep 'Back Orifice' >> back-orifice.log
```

An Example of the Perl script used for further analysis of each Alert Type

```
#!/usr/bin/perl

%alerts = ();
$num = 1;

foreach $alert (@ARGV)
{
        open(SFILE, $alert);

        while (<SFILE>)
        {
                $_ =~ /^.*:.*] (.*):.*:.*$/;

                if (!defined $alerts{$1})
                {
                        $alerts{$1} = $num;
                }
```

```
                else {
                        $alerts{$1} = $alerts{$1} + 1;
                }
        }

        close(SFILE);
}

foreach $key (keys (%alerts))
{
        print "$key = $alerts{$key}\n";
}
```

The output from this script is also directed to a file, which is then imported into Excel for further data manipulation.

This is the method I used to obtain the list of 'top talkers' for each alert type (i.e. the top ten external source IP for WinGate) and the most common source and destination ports.

3.5.2 Scans

The scan logs has the following format:

```
Jul  2 00:00:32 MY.NET.97.188:1037 -> 63.251.143.213:6003 SYN **S*****
Jul  2 00:00:34 MY.NET.97.188:1038 -> 216.52.220.15:19613 UDP
Jul  2 00:00:34 MY.NET.97.188:1038 -> 216.52.220.15:19612 UDP
Jul  2 00:00:34 MY.NET.97.188:1038 -> 216.52.220.15:19611 UDP
```

From these logs, all I wanted to analyse were the TCP flags, what protocol was being used and the most common source and destination ports.

With the TCP flags, no Perl scripts were used. Only the simple Grep command.

Example Grep command

```
#more scan* | grep –c 'UDP'
```

The above example went through each scan log and counted the number of times an UDP connection has been made. The same command structure was used to obtain the TCP count, and the count for each TCP flag.

Also, to try and utilise the relationship of the scan logs and the alerts logs more, after the list of the top ten external IP were obtained from the alert logs, the scan logs were also searched for occurrences of those particular IPs.

Example Grep command

```
#more scan* | grep '206.74.76.44' >> 206.74.76.44
```

This command direct output to a file, which can then be examined further. In my case, I looked through them, and took extracts from them to include in my report.

As for the common source and destination IPs, the same Perl script used for the alert files was used. All the scan logs were concatenated into one file and passed through the script.

3.5.3 OOS

The OOS logs have the following format:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/02-00:10:46.528381 209.150.103.213:32769 -> MY.NET.60.11:22
TCP TTL:51 TOS:0x0 ID:38309  DF
21S***** Seq: 0xE87C11D9   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 33916 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Another simple Perl script was used to gather information from the OOS files. The script is similar to the ones used to gather information from the alert files, except with the OOS files, each line is not uniform.

Hence the regular expression used to obtain the source IPs looked something like the following:

$_ =~ /^.*:.* (.*):.*:.*$/;

And the regular expression used to obtain the TCP flags looked something like this:

$_ =~ /^(.*) Seq:.*$/;

As with the alert and scan files, the OOS files are concatenated into one file and passed through the Perl script. The output of the Perl script is directed to another file, which is then imported into Excel.

3.5.4 Analyzing the Data

Once the desired data have been retrieved, these data is imported into Excel to be collaborated into tables, to allow for easy analysis. Whether the subject of the data is IPs, ports or flags, these are sorted in descending order, according to the number of occurrences. Then either the top five, ten or all occurrences are imported into this report.

To aid my investigation of the data, I used search engines such as http://www.google.com and websites such as http://www.securityfocus.com to search for further information concerning the attacks.

3.6 Misc

Scans Logs

The scan logs offers a vast amount of information, which can be used to supplement the analysis of the alert logs.

This section just presents the analysis of the scan logs, which can be used to aid the analysis of the alert and OOS logs.

a. TCP or UDP?

Protocol	Occurrences
UDP	189002
TCP	122744
Total	**311746**

b. TCP Flags

TCP Flag	Occurrences
SYN	113531
SYNFIN	8507
NOACK	179
INVALIDACK	223
FIN	16
UNKNOWN	53
NULL	152
VECNA	52
FULLXMAS	11
XMAS	9
SPAU	5
NMAPID	6
Total	**122744**

c. Top Ten Destination Ports

Port	Protocol	Occurrences	Name
6970	UDP	62963	GateCrasher - Trojan
27005	UDP	54951	FlexLM
53	SYN	38913	DNS
1214	SYN	20156	KAZZA
21	SYN	16138	FTP
7778	UDP	8893	Interwise/UnReal_UT (game)
21	SYNFIN	8224	FTP
6112	UDP	8038	dtspcd/FSGS (game)
27374	SYN	5914	SubSeven – Trojan
6346	SYN	3669	Gnutella-svc

d. Top Ten Source Ports

Port	Occurrences
777	79714
21	8227
6112	6429
7001	5019
53	4846
32805	3464
28800	2874
2000	1522
2002	1510
2007	1480

Appendix

The following main websites were used:

These are general sites that were used for references. Any quotes or direct reference to information obtained from these sites are mentioned in the body of this paper.

- http://network-tools.com/ - Whois server
- http://cve.mitre.org/ - Common Vulnerabilities and Exposures
- http://www.snort.org – Snort Website
- http://www.whitehats.com – Whitehats Network Security Resources
- http://www.google.com – Search engine
- http://www.sans.org – Sans Website
- http://www.securityfocus.com – Security Focus Website
- http://www.eeye.com/ - eEye Digital Security
- http://www.incidents.org – SANS Emergency Incident Handler

The following text were used as reference:

- Northcutt, Stephen. Network Intrusion Detection: An Analysts' Handbook. New Riders Pub, 1999

Last Updated: May 28th, 2017

Upcoming Training

Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Boston SEC503	Boston, MA	Oct 09, 2017 - Oct 14, 2017	Community SANS
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced