Global Information Assurance Certification Paper

GIAC

Intrusion Detection in Depth

Certification Practical Version 2.9

Tamara Bowman

July 2001

Section 1: 5 Detects

Detect 1

```
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.15/47017 flags SYN  on interface outside
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.20/47017 flags SYN  on interface outside
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.50/47017 flags SYN  on interface outside
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.51/47017 flags SYN  on interface outside
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.10/47017 flags SYN  on interface outside
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.14/47017 flags SYN  on interface outside
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.13/47017 flags SYN  on interface outside
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.21/47017 flags SYN  on interface outside
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.24/47017 flags SYN  on interface outside
firewall Jun 08 2001 12:28:39: %PIX-2-106001: Inbound TCP connection denied from
211.13.194.148/47017 to my.net.58.25/47017 flags SYN  on interface outside
```

<cut 7 lines of more of the same>

1. Source of Trace.

My network

2. Detect was generated by:

PIX firewall logging to syslog. The log is partially cleaned up. In a standard PIX log message to syslog there are two date and time stamps. The first data and time stamp is generated by syslog. I've stripped the initial time stamp from the logs and use only the time stamp supplied by the PIX.

The log messages have the following fields:

Hostname – Will contain the hostname or IP address of the firewall reporting the message
Date – Date the PIX logged the message
Time – Time the message was logged in the format HH:MM:SS.
Message identifier – Format %PIX-2-######. This identifies the Cisco message type. The Cisco web site (www.cisco.com) has the definition and suggested action for each message identifier
Message – In this log message "Inbound TCP connection denied from"
Source IP and port – self explanatory
Destination IP and port – self explanatory
Flags – On TCP packets the flags that were set on the packet that was denied. Valid flags are ACK, FIN, PSH, RST, SYN, & URG.
On interface –PIX interface that received the packet.

3. Probability the source address was spoofed:
Highly unlikely the source address was spoofed. These are TCP connections the three-way handshake would fail if the source were spoofed. If this were a SYN flood attempt the source could be spoofed, since the point of the flood is to leave half open connections. There are too few connections for this to be a SYN flood.

4. Description of attack
This isn't an attack but a scan for boxes that have already been compromised. The port 47017 is a sshd trojan that is part of the t0rn rootkit. From the logs the scanning packets are most certainly crafted. The source port is also 47017 and never increments. The scanning is possibly being done by a system that has already been compromised and is searching for other compromised systems.

5. Attack mechanism
The system scanning is looking for port 47017/tcp a port used by the t0rn rootkit. A rootkit is a set of binaries or utilities that are installed on a system that has been successfully cracked. The rootkit generally installs a backdoor for the cracker's return and hides the activity of the cracker. The t0rn rootkit first appeared in mid 2000. It was initially seen on systems compromised via rpc.statd or wu-ftpd vulnerabilities. It is a fairly comprehensive rootkit. It installs a trojan sshd listening on port 47017 so the compromised box can be accessed later. To cover its tracks it also installs the following trojan binaries du, find, ifconfig, login, ls, netstat, ps, sz and top. The trojan can be discovered with lsof which will show the 47017/tcp port listening or with an nmap scan. On systems running tripwire or similar file integrity checkers the trojan binaries should be revealed by the integrity checker.

6. Correlations
This port was identified as part of the t0rn rootkit in "Analysis of the T0rn Rootkit" by Toby Miller http://packetstorm.securify.com/papers/IDS/t0rn.txt. T0rn rootkit is identified in CERT® Incident Note IN-2000-10 http://www.cert.org/incident_notes/IN-2000-10.html The CERT notice identifies the t0rn kit as a rootkit that is being installed on systems compromised via rpc.statd and wu-ftpd vulnerabilities.

7. Evidence of active targeting:
No evidence of active targeting. The scanning server hit all active servers in our address space.

8. Severity:
Formula (criticality + lethality) – (system + network) = severity
(4 + 1) – (5 + 5) = -5
Criticality – (4) There was no real targeting the systems being scanned were production servers
Lethality – (1) The attack is very unlikely to succeed.
System counter measures - (5) Systems have modern OS with all patches.
Network counter measures – (5) Restricted firewall blocks access, no other way in

9. Defensive recommendation:
Defenses are fine attack was blocked by firewall and no servers were listening on port 47017.

10. Multiple choice test question:
A rootkit is:

a) Any vulnerability that allows a root shell
b) A cracker tool kit installed after a root vulnerability has been exploited
c) A tool used to crack the root password

d) Slang for an OS build disk

b

Detect 2

```
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.14/24452 flags SYN  on interface outside
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.10/24452 flags SYN  on interface outside
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.13/24452 flags SYN  on interface outside
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.15/24452 flags SYN  on interface outside
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.21/24452 flags SYN  on interface outside
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.26/24452 flags SYN  on interface outside
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.20/24452 flags SYN  on interface outside
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.24/24452 flags SYN  on interface outside
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.25/24452 flags SYN  on interface outside
firewall Jun 09 2001 08:18:53: %PIX-2-106001: Inbound TCP connection denied from
128.59.49.15/24452 to my.net.58.29/24452 flags SYN  on interface outside
```

<**cut 19 lines of more of the same**>

1. Source of Trace.

My network

2. Detect was generated by:

PIX firewall logging to syslog. The log is partially cleaned up. In a standard PIX log message to syslog there are two date and time stamps. The first data and time stamp is generated by syslog. I've stripped the initial time stamp from the logs and use only the time stamp supplied by the PIX.

The log messages have the following fields:

Hostname – Will contain the hostname or IP address of the firewall reporting the message
Date – Date the PIX logged the message
Time – Time the message was logged in the format HH:MM:SS.
Message identifier – Format %PIX-2-######. This identifies the Cisco message type. The Cisco web site (www.cisco.com) has the definition and suggested action for each message identifier
Message – In this log message "Inbound TCP connection denied from"
Source IP and port – self explanatory
Destination IP and port – self explanatory
Flags – On TCP packets the flags that were set on the packet that was denied. Valid flags are ACK, FIN, PSH, RST, SYN, & URG.
On interface – PIX interface that received the packet.

3. Probability the source address was spoofed:
Highly unlikely the source address was spoofed. These are TCP connections the three-way handshake would fail if the source were spoofed. If this were a SYN flood attempt the source could be spoofed, since the point of the flood is to leave half open connections. There are too few connections for this to be a SYN flood.

4. Description of attack
This isn't an attack but a scan for boxes that have already been compromised. The port 24452 is for a root /bin/sh in a root kit that uses Linux loadable kernel modules. The source port is also 24452 and never increments. The scanning is possibly being done by a system that has already been compromised and is searching for other compromised systems.

5. Attack mechanism
The system scanning is looking for port 24452. This port has been seen on Linux boxes compromised by a root kit using korelkm.o a loadable kernel module. The port 24452 is a root shell installed by the root kit. The loadable kernel module makes itself invisible by hiding its activity to ls, find and other similar utilities. This rootkit may be invisible to tripwire, the MD5 checksum of ls appears the same as a legitimate ls. An nmap scan should still show port 24452 listening.

6. Correlations
Monta Elkins reported this in the RVGLUG mailing list Feb 23, 2001
http://www.rvglug.org/pipermail/rvglug/2001-February/000436.html

7. Evidence of active targeting:
No evidence of active targeting. The scanning server hit all active servers in our address space.

8. Severity:
Formula (criticality + lethality) – (system + network) = severity
(4 + 1) – (5 + 5) = -5
Criticality – (4) There was no real targeting the systems being scanned were production servers
Lethality – (1) The attack is very unlikely to succeed.
System counter measures - (5) Systems have modern OS with all patches.
Network counter measures – (5) Restricted firewall blocks access, no other way in

9. Defensive recommendation:
Defenses are fine attack was blocked by firewall and no servers were listening on port 24452.

10. Multiple choice test question:
What would identify these packets as crafted?

a. The network is being scanned
b. The source port never changes
c. The SYN flag is set
d. The connections were denied

b

Detect 3

June 20 ,2001

12:17:32.108944 63.251.179.201.13570 > my.net.58.156.37852: udp 10 (ttl 54, id 40804)
12:17:32.110745 my.net.58.156 > 63.251.179.201: icmp: my.net.58.156 udp port 37852 unreachable (DF) (ttl 255, id 61043)

12:17:32.114568 63.251.179.201 > my.net.58.156: icmp: echo request (ttl 54, id 40806)
12:17:32.114633 my.net.58.156 > 63.251.179.201: icmp: echo reply (DF) (ttl 255, id 61044)
12:17:32.119810 63.251.179.201.80 > my.net.58.156.31616: . ack 0 win 1024 (ttl 54, id 40808)
12:17:32.119859 my.net.58.156.31616 > 63.251.179.201.80: R 0:0(0) win 0 (DF) (ttl 64, id 61045)
12:17:32.123792 63.251.179.201.13568 > my.net.58.156.31616: S 2652374739:2652374739(0) win 1024 (ttl 54, id 40810)
12:17:32.123874 my.net.58.156.31616 > 63.251.179.201.13568: R 0:0(0) ack 2652374740 win 0 (DF) (ttl 64, id 61046)
12:17:32.226620 63.251.179.201.13568 > my.net.58.156.31616: R 2652374740:2652374740(0) win 1024 (ttl 54, id 40812)

19:57:38.005817 63.251.179.201.13570 > my.net.58.156.37852: udp 10 (ttl 54, id 29492)
19:57:38.007427 my.net.58.156 > 63.251.179.201: icmp: my.net.58.156 udp port 37852 unreachable (DF) (ttl 255, id 9671)
19:57:38.012083 63.251.179.201 > my.net.58.156: icmp: echo request (ttl 54, id 29494)
19:57:38.012175 my.net.58.156 > 63.251.179.201: icmp: echo reply (DF) (ttl 255, id 9672)
19:57:38.017511 63.251.179.201.80 > my.net.58.156.31616: . ack 0 win 1024 (ttl 54, id 29496)
19:57:38.017573 my.net.58.156.31616 > 63.251.179.201.80: R 0:0(0) win 0 (DF) (ttl 64, id 9673)
19:57:38.020507 63.251.179.201.13568 > my.net.58.156.31616: S 963840147:963840147(0) win 1024 (ttl 54, id 29498)
19:57:38.020598 my.net.58.156.31616 > 63.251.179.201.13568: R 0:0(0) ack 963840148 win 0 (DF) (ttl 64, id 9674)
19:57:38.120830 63.251.179.201.13568 > my.net.58.156.31616: R 963840148:963840148(0) win 1024 (ttl 54, id 29500

June 22, 2001

10:43:00.347016 63.251.179.201.13570 > my.net.58.156.37852: udp 10 (ttl 54, id 21572)
10:43:00.348672 my.net.58.156 > 63.251.179.201: icmp: my.net.58.156 udp port 37852 unreachable (DF) (ttl 255, id 427)
10:43:00.354303 63.251.179.201 > my.net.58.156: icmp: echo request (ttl 54, id 21574)
10:43:00.354384 my.net.58.156 > 63.251.179.201: icmp: echo reply (DF) (ttl 255, id 428)
10:43:00.362522 63.251.179.201.80 > my.net.58.156.31616: . ack 0 win 1024 (ttl 54, id 21576)
10:43:00.362616 my.net.58.156.31616 > 63.251.179.201.80: R 0:0(0) win 0 (DF) (ttl 64, id 429)
10:43:00.375657 63.251.179.201.13568 > my.net.58.156.31616: S 1484314279:1484314279(0) win 1024 (ttl 54, id 21578)
10:43:00.375905 my.net.58.156.31616 > 63.251.179.201.13568: R 0:0(0) ack 1484314280 win 0 (DF) (ttl 64, id 430)
10:43:00.475841 63.251.179.201.13568 > my.net.58.156.31616: R 1484314280:1484314280(0) win 1024 (ttl 54, id 21580)

19:57:30.651899 63.251.179.201.13570 > my.net.58.156.37852: udp 10
19:57:30.653425 my.net.58.156 > 63.251.179.201: icmp: my.net.58.156 udp port 37852 unreachable (DF)
19:57:30.658179 63.251.179.201 > my.net.58.156: icmp: echo request
19:57:30.658229 my.net.58.156 > 63.251.179.201: icmp: echo reply (DF)
19:57:30.663489 63.251.179.201.80 > my.net.58.156.31616: . ack 0 win 1024
19:57:30.663580 my.net.58.156.31616 > 63.251.179.201.80: R 0:0(0) win 0 (DF)
19:57:30.666957 63.251.179.201.13568 > my.net.58.156.31616: S 1211842187:1211842187(0) win 1024
19:57:30.667049 my.net.58.156.31616 > 63.251.179.201.13568: R 0:0(0) ack 1211842188 win 0 (DF)
19:57:30.773548 63.251.179.201.13568 > my.net.58.156.31616: R 1211842188:1211842188(0) win 1024
< more of the same pattern of traffic over several days>

1. Source of Trace:
My network

2. Detect was generated by:
Snort logging in binary tcpdump format.
The above logs were generated with tcpdump with just the –n flag to avoid resolving domain names and port numbers. The fields are:

Timestamp – No date just the time of day in 24 hour format HH.MM.SS.milliseconds
Source IP and port – Self explanatory, without the –n flag these may be resolved to names
> - An arrow to remind you which way the communication is flowing
Destination IP and port – Self explanatory, without the –n flag these may be resolved to names
Protocol Identifier – For udp and icmp the type is listed. For tcp the flag(s) set in the packet are listed. Possible flags for tcp are S (syn), R (reset), P (push), F (finish) and . for no flags.
Additional info – This will vary by protocol type. For udp it will include number of bytes of user data and time to live, for icmp it will include the icmp type and for tcp message IDs, window size and time to live. There is additional information when using the –v (verbose) flags.

3. Probability the source address was spoofed:
Highly unlikely the source address was spoofed. There is an ongoing dialog with the system my.net.58.156. If these were spoofed addresses you would expect the communication to stop after the port unreachable message on the attempt to port 37852/udp. Instead connect attempts continue with ICMP requests and TCP connection attempts.

4. Description of attack
The connection attempts always occur in the same pattern.
1) Initial connect attempt to my.net.58.156 port 37852/udp from 63.251.179.201 port 13570
2) ICMP port unreachable message from my.net.58.156
3) Single ICMP echo request from 63.251.179.201
4) Ack from 63.251.179.201 port 80/tcp to my.net.58.156 port 31616
5) Reset from my.net.58.156 port 31616 to 63.251.179.201 port 80
6) Syn from 63.251.179.201 port 13568/tcp to my.net.58.156 port 31616
7) Reset from my.net.58.156 port 31616 to 63.251.179.201 port 13568
8) Reset from 63.251.179.201 port 13568 to my.net.58.156 port 31616

The ports being probed are not assigned by IANA, known trojans or applications. I could find no reference to what this host is trying to connect to. But the host is persistent.

5. Attack mechanism
In an attempt to gain some insight into what this communication attempt might be trying to accomplish I looked at the hexadecimal and ASCII output of one of the connections. I examined the packets targeted at my host. This is still tcpdump. The first line shows the timestamp, source IP address and port, destination IP address and port followed by the protocol. The following lines are hex followed by ASCII interpretation.

For this udp packet to port 37852 there are 10 bytes of user data. There doesn't seem to be anything in the datagram (the user data is highlighted in yellow) to identify what the user is attempting to accomplish with this packet.

```
10:43:00.347016 63.251.179.201.13570 > my.net.58.156.37852:  udp 10
0x0000   4500 0026 5444 0000 3611 c2a9 3ffb b3c9        E..&TD..6...?...
0x0010   3f79 3a9c 3502 93dc 0012 c911 0000 0000        ?y:.5...........
0x0020   0000 0000 0000 0001 0543 4f41 4952             .........COAIR
```

The ICMP message is highlighted in yellow. The first two bytes give the ICMP type (8) and ICMP code (0). This is an echo request, which we can already see in the first line. The rest of the ICMP packet does not appear to contain anything unusual. If this is a stealth method of sending data imbedded in ICMP packets its very stealthy and very inefficient. At one or two ICMP packets a day it could take years to communicate a message. Chances that these packets are being used for stealth communication are very slim.

```
10:43:00.354303 63.251.179.201 > my.net.58.156: icmp: echo request
0x0000   4500 0026 5446 0000 3601 c2b7 3ffb b3c9        E..&TF..6...?...
0x0010   3f79 3a9c 0800 5076 936f 0001 0001 0203        ?y:...Pv.o......
0x0020   0405 0607 0809 0000 3235 302d 696d             ........250-im
```

This is a TCP packet sent from the attack host's port 80 to my host's port 31616. The TCP header is highlighted in yellow. The only unusual thing about this packet is that it is an ACK packet. There was no corresponding SYN packet sent from my host so this packet is out of sequence. There are no flags set that would indicate this might be a fingerprinting technique.

```
10:43:00.362522 63.251.179.201.80 > my.net.58.156.31616: . ack 0 win 1024
0x0000   4500 0028 5448 0000 3606 c2ae 3ffb b3c9        E..(TH..6...?...
0x0010   3f79 3a9c 0050 7b80 0000 0393 0000 0000        ?y:..P{.........
0x0020   5010 0400 be97 0000 3235 3020 3c6d             P.......250.<m
```

This is a TCP packet sent from the attack host's port 13568 to my host's port 31616. It is a SYN packet, which is expected at the start of a TCP communication. The TCP header is highlighted in yellow. There is nothing unusual about this packet, no flags are set, nothing to indicate what the intent might be.

```
10:43:00.375657 63.251.179.201.13568 > my.net.58.156.31616: S 1484314279:1484314279(0) win 1024
0x0000   4500 0028 544a 0000 3606 c2ac 3ffb b3c9        E..(TJ..6...?...
0x0010   3f79 3a9c 3500 7b80 5878 d6a7 0000 0000        ?y:.5.{.Xx......
0x0020   5002 0400 5e68 0000 3235 302d 686d             P...^h..250-hm
```

This is a TCP packet sent from the attack host. It is a RESET packet, evidently in response to its last connection attempt. The sequence numbers are valid and there is nothing else unusual about this packet other than why it sent the RESET, my host also sent a RESET in response to the original packet.

```
10:43:00.475841 63.251.179.201.13568 > my.net.58.156.31616: R 1484314280:1484314280(0) win 1024
0x0000   4500 0028 544c 0000 3606 c2aa 3ffb b3c9        E..(TL..6...?...
0x0010   3f79 3a9c 3500 7b80 5878 d6a8 0000 0000        ?y:.5.{.Xx......
0x0020   5004 0400 5e65 0000 3235 3020 3c45             P...^e..250.<E
```

I have not been able to determine the purpose behind these communication attempts. There are small anomalies within the series of packets but not enough for this to be a fingerprinting

attempt. This is not normal IP traffic since the source ports that attempt the connections never vary. The packets are definitely crafted but what they are trying to accomplish is a mystery.

6. Correlations
None were found for this particular pattern. This appears to be an original.

7. Evidence of active targeting:
Evidence is strong that my DNS server is being actively targeted. The DNS server is the only one on this subnet being targeted in this fashion. This does not appear to be a wrong number because the connection attempts persist over several weeks. If it were someone's automated script gone awry I would expect to see connection attempts at standard times, but the connections occur at random times.

8. Severity:
Formula (criticality + lethality) – (system + network) = severity
(5 + 1) – (5 + 0) = 1
Criticality – (4) The server targeted is the DNS server
Lethality – (1) The attack is very unlikely to succeed.
System counter measures - (5) Systems have modern OS with all patches and server is hardened.
Network counter measures – (0) Server is in the DMZ

9. Defensive recommendation:
Defenses are fine the server doesn't listen on any ports being probed.

10. Multiple choice test question:
The following two lines are the complete log entries for a connection between two hosts. What is unusual about this connection?

19:57:30.663489 63.251.179.201.80 > my.net.58.156.31616: . ack 0 win 1024
19:57:30.663580 my.net.58.156.31616 > 63.251.179.201.80: R 0:0(0) win 0 (DF)

a) The window size is 1024, a standard window is 512
b) The responding host has set the do not fragment (DF) flag
c) The connection begins with an ACK
d) There is nothing unusual about this connection

c

Detect 4

firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22664 to my.net.58.13/98 flags SYN on interface outside
firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22671 to my.net.58.20/98 flags SYN on interface outside
firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22710 to my.net.58.24/98 flags SYN on interface outside
firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22661 to my.net.58.10/98 flags SYN on interface outside
firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22768 to my.net.58.26/98 flags SYN on interface outside
firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22665 to my.net.58.14/98 flags SYN on interface outside
firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22666 to my.net.58.15/98 flags SYN on interface outside
firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22774 to my.net.58.28/98 flags SYN on interface outside

firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22778 to my.net.58.30/98 flags SYN on interface outside
firewall Jul 10 2001 21:54:48: %PIX-2-106001: Inbound TCP connection denied from 129.24.184.102/22705 to my.net.58.21/98 flags SYN on interface outside
<**cut 48 lines of more of the same**>

1. Source of Trace.

My network

2. Detect was generated by:

PIX firewall logging to syslog. The log is partially cleaned up. In a standard PIX log message to syslog there are two date and time stamps. The first data and time stamp is generated by syslog. I've stripped the initial time stamp from the logs and use only the time stamp supplied by the PIX.

The log messages have the following fields:

Hostname – Will contain the hostname or IP address of the firewall reporting the message
Date – Date the PIX logged the message
Time – Time the message was logged in the format HH:MM:SS.
Message identifier – Format %PIX-2-######. This identifies the Cisco message type. The Cisco web site (www.cisco.com) has the definition and suggested action for each message identifier
Message – In this log message "Inbound TCP connection denied from"
Source IP and port – self explanatory
Destination IP and port – self explanatory
Flags – On TCP packets the flags that were set on the packet that was denied. Valid flags are ACK, FIN, PSH, RST, SYN, & URG.
On interface – PIX interface that received the packet.

3. Probability the source address was spoofed:

Highly unlikely the source address was spoofed. These are TCP connections the three-way handshake would fail if the source were spoofed. If this were a SYN flood attempt the source could be spoofed, since the point of the flood is to leave half open connections. There are too few connections for this to be a SYN flood.

4. Description of attack

This is a scan for systems listening on port 98/tcp. The packets being sent do not appear to be crafted, the source ports increment and only the SYN flag is set on the initial connect. This is normal behavior for a TCP connection attempt. There may be a buffer overflow attack possible against linuxconf on port 98/tcp, which allows creation of a root shell. The attack is currently a CVE candidate CAN-2000-0017 under review http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0017

5. Attack mechanism

This is a scan to find port 98/tcp open. The port 98/tcp is used by some Linux distributions for linuxconf. There may be a buffer overflow vulnerability in linuxconf particularly in how it handles HTTP headers. The exploit code given in the BUGTRAQ posting uses an http POST statement to port 98 with the data intended to overflow the buffer. So far no one in the security community has been able to use the exploit successfully. It is possible that the increased

scanning activity on port 98/tcp is just a means for crackers to identify Linux systems. It is equally possible that there is a vulnerability in linuxconf and it hasn't been identified by the security community yet.

6. Correlations

The increased scanning for port 98/tcp was noted in the CERT/CC Current Activity Tue Jun 26 09:52:57 EDT 2001 http://www.cert.org/current/current_activity.html

Possible buffer overflow vulnerability in linuxconf was reported in BUGTRAQ url: http://www.securityfocus.com/archive/1/39279

7. Evidence of active targeting:

No evidence of active targeting. The scanning server hit all active servers in our address space.

8. Severity:

Formula (criticality + lethality) – (system + network) = severity
(4 + 1) – (5 + 5) = -5
Criticality – (4) There was no real targeting the systems being scanned were production servers
Lethality – (1) The attack is very unlikely to succeed (we're not running Linux).
System counter measures - (5) Systems have modern OS with all patches.
Network counter measures – (5) Restricted firewall blocks access, no other way in

9. Defensive recommendation:

Defenses are fine attack was blocked by firewall and no servers were listening on port 98.

10. Multiple choice test question:

Buffer overflows are caused by:

a) Programs not checking input before processing
b) Insufficient system memory
c) Users keying data incorrectly
d) Network traffic

a

Detect 5

June 25, 2001

```
00:12:32.692077 208.10.129.66.4547 > my.net.58.156.53: S 3598405290:3598405290(0) win 32120
<mss 1460,sackOK,timestamp 97436697 0,nop,wscale 0> (DF)
00:12:32.692148 my.net.58.156.53 > 208.10.129.66.4547: S 2026221812:2026221812(0) ack
3598405291 win 24616 <nop,nop,timestamp 414122359 97436697,nop,wscale 0,nop,nop,sackOK,mss
1460> (DF)
00:12:32.733598 208.10.129.66.4547 > my.net.58.156.53: . ack 1 win 32120 <nop,nop,timestamp
97436702 414122359> (DF)
00:12:32.779780 208.10.129.66.4019 > my.net.58.156.53:  14049 TXT CHAOS)? version.bind. (30)
00:12:32.783407 my.net.58.156.53 > 208.10.129.66.4019:  14049*- 1/0/0 CHAOS) TXT 9.1.1 (48) (DF)
00:12:32.820025 208.10.129.66.4019 > my.net.58.156.53:  14049 inv_q+ [b2&3=0x980] A?
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.BBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB.CCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC.^@^A^B^C^D^E^F^G^H
^I^J^K^L^M^N^O^P^Q^R^S^T^U^V^W^X^Y^Z^[^\^]^^^_ !"#$%&'()*+,-
./0123456789:;<=.EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEE.FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.GGGGGG
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG. (465)
00:12:32.829301 my.net.58.156.53 > 208.10.129.66.4019:  14049 inv_q ServFail- [0q] 0/0/0 (12) (DF)
```

1. Source of Trace.
My network

2. Detect was generated by:
Snort logging in binary tcpdump format.
The above logs were generated with tcpdump with just the –n flag to avoid resolving domain names and port numbers. The fields are:

Timestamp – No date just the time of day in 24 hour format HH.MM.SS.milliseconds
Source IP and port – Self explanatory, without the –n flag these may be resolved to names
> - An arrow to remind you which way the communication is flowing
Destination IP and port – Self explanatory, without the –n flag these may be resolved to names
Protocol Identifier – For udp and icmp the type is listed. For tcp the flag(s) set in the packet are listed. Possible flags for tcp are S (syn), ack (ack), R (reset), P (push), F (finish)
Additional info – This will vary by protocol type. For udp it will include number of bytes of user data and time to live, for icmp it will include the icmp type and for tcp message IDs, window size and time to live. There is additional information when using the –v (verbose) flags.

NOTE: These are mostly dns related entries and the protocol identifier and additional information provided are specific to DNS. After the destination IP and port a request to a DNS server has the numerical ID of the request, the operation requested, any flags, any arguments to the operation and finally the size in bytes of the request minus IP and UDP headers.

The response back to a request has a slightly different format. After the destination IP and port the response has the numerical ID of the original request, followed by numbers in the format 1/0/1, the type of response, the data for the response and the size in bytes of the request minus IP and UDP headers. The numbers 1/0/1 mean the server responded with 1 answer record, 0 name server records and 1 authority record.

3. Probability the source address was spoofed:
Highly unlikely the source address was spoofed. These are TCP connections the three-way handshake would fail if the source were spoofed. If this were a SYN flood attempt the source could be spoofed, since the point of the flood is to leave half open connections. There are too few connections for this to be a SYN flood.

4. Description of attack
This is one of many buffer overflow attacks against BIND. This particular attack uses a buffer overflow when handling inverse queries. This attack is registered as CVE-1999-0009
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0009

5. Attack mechanism
The attack works by contacting the DNS port with an inverse query. The argument to the inverse query is a large number of characters followed by an attempt to open a root shell. This attack can either crash the DNS server or successfully open a root shell. This vulnerability is only present in BIND versions up to 4.9.7 and 8.1.2 if inverse queries are enabled.

1 00:12:32.779780 208.10.129.66.4019 > my.net.58.156.53: 14049 TXT CHAOS)? version.bind. (30)
2 00:12:32.783407 my.net.58.156.53 > 208.10.129.66.4019: 14049*- 1/0/0 CHAOS) TXT 9.1.1 (48) (DF)

3 00:12:32.820025 208.10.129.66.4019 > my.net.58.156.53: 14049 inv_q+ [b2&3=0x980] A?
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.BBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB.CCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC.^@^A^B^C^D^E^F^G^H
^I^J^K^L^M^N^O^P^Q^R^S^T^U^V^W^X^Y^Z^[^\^]^^^_ !"#$%&'()*+,-
./0123456789:;<=.EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEE.FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.GGGGGG
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG. (465)
4 00:12:32.829301 my.net.58.156.53 > 208.10.129.66.4019: 14049 inv_q ServFail- [0q] 0/0/0 (12) (DF)

In the lines above line numbers were added for easy reference. Line 1 shows the attacker requesting the BIND version. Line 2 is the response with the BIND version of 9.1.1. Even though the BIND version is not one that is vulnerable to this buffer overflow the attacker tries anyway. Line 3 shows the inverse query. The inv_q+ in field 5 denotes an inverse query. The argument is the padding characters for the buffer overflow. The snaplen wasn't long enough to capture the entire query argument. Line 4 shows the inverse query attempt failing. This buffer overflow is fixed in this version of BIND.

6. Correlations
CERT* Advisory CA-98.05 Topic: Multiple Vulnerabilities in BIND
http://www.cert.org/advisories/CA-98.05.bind_problems.html

The vulnerability was also reported in Security Focus. I-044A: BIND Vulnerabilities Published: Mon Dec 28 1998 Updated: Mon Dec 28 1998 http://www.securityfocus.com/advisories/1093

7. Evidence of active targeting:
This attacker was actively targeting DNS servers. My entire network was scanned for open DNS ports. Once the actual DNS server was identified the attacker attempted the buffer overflow exploit.

8. Severity:
Formula (criticality + lethality) – (system + network) = severity
(5 + 1) – (5 + 0) = 1
Criticality – (5) The server being targeted is the primary DNS server
Lethality – (1) The attack is very unlikely to succeed.
System counter measures - (5) Systems have modern OS with all patches and up to date BIND.
Network counter measures – (0) This server is in the DMZ, no firewall

9. Defensive recommendation:
Defenses are fine. BIND version is not vulnerable to this attack. Being behind our firewall would not have improved our defenses in this instance since the external firewall is a stateful firewall. A proxy firewall may have caught the inverse query overflow attempt.

10. Multiple choice test question
00:12:32.692077 208.10.129.66.4547 > my.net.58.156.53: S 3598405290:3598405290(0) win 32120 <mss 1460,sackOK,timestamp 97436697 0,nop,wscale 0> (DF)
00:12:32.692148 my.net.58.156.53 > 208.10.129.66.4547: S 2026221812:2026221812(0) ack 3598405291 win 24616 <nop,nop,timestamp 414122359 97436697,nop,wscale 0,nop,nop,sackOK,mss 1460> (DF)
00:12:32.733598 208.10.129.66.4547 > my.net.58.156.53: . ack 1 win 32120 <nop,nop,timestamp 97436702 414122359> (DF)
00:12:32.779780 208.10.129.66.4019 > my.net.58.156.53: 14049 TXT CHAOS)? version.bind. (30)
00:12:32.783407 my.net.58.156.53 > 208.10.129.66.4019: 14049*- 1/0/0 CHAOS) TXT 9.1.1 (48) (DF)

00:12:32.820025 208.10.129.66.4019 > my.net.58.156.53: 14049 inv_q+ [b2&3=0x980] A?
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.BBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB.CCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC.^@^A^B^C^D^E^F^G^H
^I^J^K^L^M^N^O^P^Q^R^S^T^U^V^W^X^Y^Z^[^\^]^^^_ !"#$%&'()*+,-
./0123456789:;<=.EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEE.FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.GGGGGG
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG. (465)
00:12:32.829301 my.net.58.156.53 > 208.10.129.66.4019: 14049 inv_q ServFail- [0q] 0/0/0 (12) (DF)

The preceding tcpdump trace does NOT show:

a) DNS version query
b) DNS inverse query
c) DNS zone transfer
d) Scan for DNS port

c

Section 2: Managing Snort Intrusion Detection Logs

Snort (www.snort.org) is a flexible lightweight intrusion detection system and its free. This comes in handy when you need to implement IDS but can't get budget allocated. Snort can be used to generate alerts in response to attack signatures. This could be considered its main purpose. But, snort can also be used to collect network traffic other than alerts. This additional data can be analyzed to identify out of spec traffic and along with the alerts create trends of network activity.

Why bother collecting more data if the alerts are working? The main weakness of any IDS is that it can only generate alerts on known attack signatures. At any given time there are numerous attacks in the wild that have not been identified and do not have signatures. Attacks can be occurring for weeks or months before being captured and analyzed by the security community. Collecting additional data outside of the alerts is a way of mitigating this weakness. If the out of spec traffic on your network is analyzed you may discover a new risk to your network and be able to act before the attacker is successful. Looking at the data over the long term can also reveal attack patterns or activity that is not apparent looking at a day or a week's worth of data.

Working Environment

Following is a methodology for collecting and managing intrusion detection data using snort as the IDS. This method was designed for a switched network with multiple VLANs and multiple levels of firewall protection. There are 6 snort collectors. The snort collectors are attached to the networks they monitor on an interface that does not have an assigned IP address. The collectors are also attached to a second network with an assigned IP address. The second network is used to communicate with the central log server. The central log server is located on an internal administrative network and only has a private IP address. Communication between the log server and all other servers is via the ssh utilities, using ssh v2.4. All the servers are running Solaris 2.8. The versions of software being used are snort v1.7 and tcpdump v3.5. Simple shell and perl scripts are used to collect and manage the data at the log host.

The logs

Two copies of snort are running in daemon mode on all snort collectors. One copy is running in alert mode logging to syslog only. The syslog data is automatically logged to the remote loghost. The second copy of snort is collecting network traffic and logging it in binary (tcpdump) mode. Since the logs are coming from multiple hosts the default log name was changed to include the hostname and a date stamp. Setting the –L flag using `hostname`.`date '+%m%d.%H%M'`.log gives a log file of the form sunhost.0713.1615.log.

There are three types of logs for analyzing network traffic that doesn't generate alerts. Each is handled differently. The logs are all traffic, out of spec and summary. A description of each log, its purpose and how it is handled follow.

All traffic logs

These are the binary log files generated by each snort collector. They contain all the network traffic that was received on the outside interface of the collector. These are kept on disk for 3 months. These are the largest logs.

The all traffic logs are collected from the snort sensors on a daily basis. The logs are stored on the log host in a directory structure that stores them by month. The all traffic logs are used to research known alerts and to check for attacks as new signatures are received.

As an alert is received it goes through the following basic analysis.

- The alert "SMTP sendmail 8.6.9 exploit" from host BAD.NET.3.2 is received. First, identify what sensor or sensors generated the alert. The alert was generated at sensor 1, 3 and 4.
- Use the all traffic logs to look at the detail of the communication that generated the alert and determine if the alert is a true positive and whether it was successful. The alert was a true positive but we're running a later version of sendmail so the attack was unsuccessful.
- Check the all traffic logs for sensors 2, 5 and 6. These sensors did not generate the alert for the same traffic pattern. Sensors 2 and 5 did not have any traffic from BAD.NET.3.2 but sensor 6 did.
- Sensor 6 did not generate an alert but should have. Check the rule set for sensor 6. For some reason sensor 6 does not include the smtp.rules file. Replace the smtp.rules file.

The all traffic logs can also be used when new signatures are identified. The logs can be processed through snort with a configuration file that just contains the new signature. This can determine if a newly identified attack was tried on your network. The full logs would also allow you to determine if the attack had been successful.

This traffic history can be particularly useful for applications that are frequent targets such as, BIND, SMTP and HTTP. For servers running these protocols it is worthwhile to keep all traffic logs for 6 months. If disk space is too tight to save the full all traffic logs for 6 months, strip out the traffic to the target servers using tcpdump. For example, to save only the entries for the mail server 192.168.5.15 from the binary dump file sunhost.0712.1458.log to a file mailhost.0712.1548.log use the tcpdump command.

```
tcpdump –n –r sunhost.0712.1458.log –w mailhost.0712.1548.log host 192.168.5.15
```

What does this command do?
The –n flag in this command preserves the IP addresses and port numbers. Without this flag tcpdump will list domain name entries and port names from the /etc/services file for IP addresses and port numbers. This information is only useful if the domain and port names were guaranteed to be consistent across environments and over time.

The –r flag specifies data is to be read from a file. The argument immediately after the flag is the file to read from. By default tcpdump will collect data from the primary interface.

The –w flag specifies output is to be written to a file. The argument immediately after the flag is the file to be written to. By default tcpdump will write output to standard out.

The host keyword identifies that only network traffic for the host 192.168.5.15 is to be saved. This command can easily be changed to include multiple hosts. To collect data on multiple hosts from the same binary file specify multiple hosts. For example start with the initial command and change the host arguments.

….. host 192.168.5.15 and host 192.168.5.20 and host 192.168.5.21

This will save data for all three hosts to the specified file.

The benefits of keeping the all traffic logs are we are able to positively determine whether a known alert is valid and whether the attack was successful. We can also double check the configuration and confirm that all components of our IDS are operating as intended. Keeping the history allows us to check as new signatures are identified and determine if our network was attacked and if the attack was successful.

Out of spec logs

Out of spec traffic is network traffic that isn't expected. Some but not all of out of spec traffic will generate alerts. These logs are generated using the all traffic logs as input and eliminating all the expected traffic patterns. For example, if you have web servers on the network it is expected for them to receive connections on port 80 and port 443. Those records would be eliminated from the log file. The out of spec logs are much smaller than the logs with all traffic and could be kept on disk for 6 months for quick reference.

The out of spec logs are generated daily when the all traffic logs are collected. The all traffic logs are processed through tcpdump to eliminate normal traffic. If there is out of spec traffic (sometimes there isn't) the output is mailed to the security and network staff for review. The file output is also saved in binary format for reference later.

To create the appropriate tcpdump filter, identify normal traffic. If you haven't done this before prepare to be surprised by what is normal on your network. A simple example would be a segment with a DNS server. The only normal traffic would be port 53 to the DNS server. Anything that is not directed to port 53 for this server would be out of spec. The following tcpdump command would remove the normal traffic from the log file snort.0714.log and put the output in log file OOS.snort.0714.log.

```
tcpdump –n –r snort.0714.log port !53 and host MY.NET 58.156 –w OOS.snort.0714.log
```

This is a very simple example, most network segments are not going to be this straightforward. For more complex filter rules use a filter file. Do NOT put comments in a tcpdump filter file. Tcpdump will choke on the input. You can include comments in a separate file. For example create a filter oos.filter and have the accompanying comments in oos.filter.README.

Lets look at a more complex filter rule. The expected servers and services on this segment are:

- MY.NET.58.156 , DNS server and time keeper, DNS (53) and NTP (123)
- MY.NET.58.216, Web server, HTTP (80)
- SSH is used to administer both servers so any traffic from port 22 is expected.

The following filter rules filter out the normal traffic to these servers: communication to port 22 (ssh) and all arp packets, communication for host MY.NET.58.156 to port 53 (DNS) and 123 (NTP) and communication to host MY.NET.58.216 to port 80 (HTTP).

```
(port !22 and !arp)
and
(( host MY.NET.58.156 and (port !53 and !123))
or
( host MY.NET.58.216 and port !80))
```

This filter file can be easily modified as new servers and/or services are added. The filters for segments supporting corporate traffic can be complex but are manageable with filter rules files.

The out of spec logs are reviewed daily by network and security staff. The first thing to look for in the logs is external connections that have succeeded. If an external connection succeeds it could be a sign of a problem or that all of the normal traffic hasn't been identified. If the connection is part of normal traffic the new pattern needs to be added to the tcpdump filter.

If the connection isn't normal traffic it needs to be treated as an incident. The impacted server needs to be audited to determine if the server has been cracked and to identify what non-standard services are running. Follow standard incident handling procedures to preserve the evidence and rebuild the server. If the server is inside a firewall the firewall rules need to be audited to determine why the connection was allowed through. Steps need to be taken to block access at the firewall and to control changes to the firewall.

Next check the out of spec external connections that were rejected. These could be accidental connections or information gathering attempts. Accidental connections will generally show up as one-time connections or possibly a cluster of connections if they can't believe they typed it wrong. These generally are not a problem. Reexamine the logs if you notice the same accidental connection recurring. That might represent a slow scan. On information gathering attempts like port scans contact the network administrator for the host and let them know they have a potential problem host.

Finally, check the outbound traffic. The outbound traffic can give you some warning if you have a compromised workstation. In today's environment with laptops proliferating its incredibly easy for a user to connect to an outside network and acquire a virus or worm. Their connections at home or while traveling are rarely protected by a firewall and users have an uncanny tendency to disable features that slow them down, like virus protection. If the outbound out of spec traffic shows connection attempts to Back Orifice or similar activities the workstation generating the activity needs to be located and cleaned up. Keep track of the time and effort involved in cleaning up the workstations and how often they are reinfected. If enough time is spent cleaning up infected laptops there might be justification for implementing personal firewalls on the laptops.

The out of spec logs are a useful means of handling suspect traffic on your network. This is particularly true of traffic that has not yet been identified as an attack. Reviewing these logs lets you act proactively rather than reactively when dealing with potential security threats.

Summary logs

Summary logs are spreadsheets that just show the bare bones of the out of spec network traffic. The summary includes the date, time, source IP, source port, destination IP, destination port and

protocol information. The summary logs are kept for a year and are used to graph network activity and trends.

How is this useful? Once the summary data is in a spreadsheet its easy to use the data manipulation features to examine the data. Spreadsheets also have graphic capabilities that are helpful if you need to present your data to non-technical people.

Use the summary data to create your own top ten lists such as, the top ten addresses probing your site and top ten ports probed. On the top ten addresses probing your site is the same network showing up month after month? It may be time for some serious discussion with the network administrators from that site. If phone calls and escalation aren't effective it might be worthwhile to block that network at the perimeter router.

Publish your top ten ports probed monthly to management along with how your security policy is effectively mitigating the risk or where action needs to be taken. The intent in publishing the list is to keep upper management aware of the ongoing security threats and your security policy implementation. When sending the top ten ports probed I order the list by level of potential risk rather than number of probes received.

For example some of the top ten ports probed for your site might be DNS, Web, FTP and RPC. You can publish the full list of ports and then supply details for each probe such as:

> Port 53 DNS – Our DNS server has been receiving buffer overflow attempts that do not fit old vulnerabilities. The increased activity may be related to a new vulnerability. Currently the BIND code on the DNS server is 2 revisions behind. Considering the increased activity and the likelihood of a new BIND vulnerability, testing and deployment of the new BIND release needs to take priority.
>
> Port 80 Web – There have been a number of new vulnerabilities identified in the IIS server. We are currently behind in patches for this code. The patch required to fix one of the vulnerabilities breaks a feature in our main web page. We are working with the vendor to resolve this issue. If our web site were successfully cracked it would likely be defaced and would require several hours to recover.
>
> Port 21 FTP - Probes of the FTP port have shown increased activity since the announcement of a vulnerability in a popular anonymous ftp server. Per our security policy we do not run anonymous ftp servers and the ftp access is blocked at the firewall.
>
> Port 111 RPC – Scans for open RPC ports continue to be in the top ten. Per our security policy access to the RPC port 111 is blocked at the firewall and no Internet accessible servers are running any RPC services.

These descriptions are generic. For a real report you would want to include real patch numbers, cost estimates and level of effort if a cleanup is required and estimated times to resolve problems. The top ten ports probed is also a good spot to highlight current Internet threats. If there is an active worm or vulnerability that could have impacted your site discuss how counter measures you've taken worked or didn't and suggest improvements.

In addition to the top ten lists look for anomalous patterns. Does a particular network or host try a connection at the same time every week or day? Is there evidence of a super slow scan? Use this information to contact other network administrators and identify possible compromised hosts.

Summary data is useful for identifying slow attacks and for trending network activity. The data can be used to keep management informed and as input for security policy.

Program for extracting summary data

Following is a simple PERL script for creating summary files from tcpdump data. This script was designed to handle IP numbers not qualified domain names, use the –n flag to tcpdump. This script does not process multicast or arp packets. It handles icmp, udp and tcp output. The data is extracted into the following comma separated fields:

Date – Supplied at the command line with the –d flag. My logs are rolled at midnight so all records in a given log are the same date.

Time

Source IP Address

Source Port (for ICMP packets this field is blank)

Destination IP Address

Destination Port (For ICMP packets this field is blank)

Protocol – 5 fields to identify protocol

```
#!/usr/bin/perl
# summary - script to take tcpdump –n as input and convert to comma delimited format
# for spreadsheets
#
use Getopt::Std;

getopts ("d:", \%args);
$date=$args{d};

while (<>){
if (/^\d\d:\d\d:\d\d/){
        if (/icmp/)     {
        @icmp = split (" ", );
        chop @icmp[3];
        chop @icmp[4];
        print ($date, ",",@icmp[0], ",", @icmp[1], ",,",@icmp[3], ",,", @icmp[4], " ", @icmp[5]," ",@icmp[6],"
",@icmp[7]," ",@icmp[
8]," ",@icmp[9]," ",@icmp[10], "\n");
        }       else{
        ($time, $src, $null, $dest, $proto, $proto1, $proto2, $proto3, $proto4)  = split(" ", );
        @src = split (/\./, $src);
        $src_port = pop (@src);
        @dest = split (/\./, $dest);
        $dest_port = pop (@dest);
        chop $dest_port;
        print ($date,",",$time, ",",@src[0],".",@src[1],".",@src[2],".",@src[3],",",$src_port,",",@dest[0],"."
,@dest[1],".",@dest[2]
,".",@dest[3],",",$dest_port,",",$proto," ",$proto1," ",$proto2," ",$proto3," ",$proto4, "\n");       }
}}
```

Section 3: Analyze this

Files that were analyzed

Snort Alert Logs	Out of Spec Logs	Snort Scan Reports
Alert-23-Mar	OOS-Mar.23.2001.packets.de0	SnortScan-23-Mar
		SnortScan-24-Mar
	OOS-Mar.25.2001.packets.de0	
Alert-26-Mar	OOS-Mar.26.2001.packets.de0	SnortScan-26-Mar
Alert-27-Mar	OOS-Mar.27.2001.packets.de0	SnortScan-27-Mar

The following files were available for analysis for Mar 23 through Mar 27. There are alerts files missing for traffic on Mar 24 and Mar 25. The OOS file for Mar 24 is missing and the snort scan file is missing for Mar 25.

Executive Summary

The network security analysis for MY.NET was performed using a five-day snapshot of available data from March 23 through March 27. Even with this limited view of the network traffic there are some clear issues. The network perimeter requires attention. Three of the top ten alerts are generated by packets that do not have a source or destination on MY.NET. Making configuration changes on the perimeter could eliminate these alerts.

The network is under constant scrutiny from outside. Scanning activity is persistent and is sometimes mirrored by similar scans from internal hosts. Activities on hosts within MY.NET indicate systems that have been compromised. There are a number of hosts that are actively scanning for Ramen. Other MY.NET hosts are communicating with external networks. Since, the traffic appears in the out of spec logs it is probable that the communication is suspect. Measures need to be taken to identify legitimate business traffic and to tighten access to only permit the legitimate traffic.

From the snapshot that was analyzed the network security appears to be barely adequate. The perimeter network does not follow best practices and there is evidence of compromised hosts within the network.

Detects

Detects in order of number of occurrences. The description includes CVE numbers of exploits identified for the detect and estimated level of risk based on the severity formula.

Watchlist 000220 IL-ISDNNET-990517	9676 occurrences	Risk - Low

This alert triggers on access from network space assigned to Israel and China. It was watch listed in 1999 due to increased scanning activity from that net space. There isn't a record of increased cracks from this particular network. The alert has been removed from the current snort rulebase. This detect is noted in David Singer's practical available at http://www.sans.org/y2k/practical/David_Singer_GCIA.doc. There are no CVEs related to this detect.

Basing the risk on the severity formula this detect scores a low. The attempts are to multiple ports and overall do not appear to be successful. One exception is possible gnutella traffic to MY.NET.219.14

Attempted Sun RPC high port access	8926 occurrences	Risk - Low

This detect was triggered by attempted access to port 32771. This port is frequently used to support rpcbind or portmapper. Rpcbind maps port numbers to rpc services. It must be running on a server supporting RPC services such as NFS. This detect is also in Guy Bruneau's practical available at http://www.sans.org/y2k/practical/Guy_Bruneau.doc

CVE-1999-0168 - The portmapper may act as a proxy and redirect service requests from an attacker, making the request appear to come from the local host, possibly bypassing authentication that would otherwise have taken place. For example, NFS file systems could be mounted through the portmapper despite export restrictions.

CAN-1999-0195 Denial of service in RPC portmapper allows attackers to register or unregister RPC services or spoof RPC services using a spoofed source IP address such as 127.0.0.1.

Basing the risk on the severity formula this detect scores low. The attempts while persistent are not successful so it appears the defenses are sufficient to the threat.

UDP SRC and DST outside network	3077 occurrences	Risk - Low

This alert is generated by UDP packets that do not belong on this network. The alert is a sign of poor perimeter network management. If the packets are not sourced or destined for MY.NET network they are most likely forged. Previously compromised servers may be generating the packets. There are no CVEs for this alert.

This detect scores a low risk. These packets have no potential impact on our network. It would be potentially worthwhile to collect MAC data and try to identify what hosts are generating these packets.

Possible RAMEN server activity	232 occurrences	Risk - High

This alert was triggered by activity on source port 27374 the default port for the Ramen worm and for the SubSeven trojan. The Ramen worm takes advantage of weaknesses in lpd, wu-ftpd or LPRng to spread to Redhat 6.2 and Redhat 7 systems. There are 4 MY.NET hosts that are generating Ramen traffic. There are 30 MY.NET hosts generating the alerts but some of the traffic appears to be normal high port usage.

The Ramen worm is discussed at ArachNIDS at http://www.whitehats.com/info/IDS460 and http://www.whitehats.com/info/IDS461 The SubSeven trojan is a point and click trojan. Once a Windows host is infected the SubSeven port can be used to manipulate files, monitor keystrokes and control network services.

There are no CVE's specific to Ramen or SubSeven.

This detect scores a high risk. The Ramen activity has sources inside and outside the network indicating there are compromised hosts.

SMB Name Wildcard	205 occurrences	Risk - Low

This alert shows attempts to gather NetBIOS information using wildcards. This is a reconnaissance attempt. They are likely attempts to locate open file shares. No traffic was generated from inside the network. The alert is also discussed in Teri Bidwell's practical available at http://www.sans.org/y2k/practical/Teri_Bidwell_GCIA.doc

There are no CVE's specific to this alert.

This detect scores a low risk. This is a reconnaissance attempt. It would be worthwhile to create a temporary watch list of the external networks attempting to gather data to see if there is further activity beyond the recon.

connect to 515 from outside	188 occurrences	Risk - High

This alert represents a successful connection on port 515 (lpd) from an external host. The lpd service is susceptible to numerous vulnerabilities. Weaknesses in lpd have been used to spread worms like Ramen.

There are numerous CVE and CVE candidates for lpd vulnerabilities dependent on what operating system or printers are in use.

CVE-1999-0299 Buffer overflow in FreeBSD lpd through long DNS hostnames.

CAN-1999-0061 File creation and deletion, and remote execution, in the BSD line printer daemon (lpd).

CAN-2000-0839 WinCOM LPD 1.00.90 allows remote attackers to cause a denial of service by sending a large number of LPD options to the LPD port (515).

CAN-2000-1064 Buffer overflow in the LPD service in HP JetDirect printer card Firmware x.08.20 and earlier allows remote attackers to cause a denial of service.

CAN-2001-0353 Buffer overflow in the line printer daemon (in.lpd) for Solaris 8 and earlier allows local and remote attackers to gain root privileges via a "transfer job" routine.

This detect scores a high risk. This alert represents a successful connection. This would indicate that perimeter defenses have been successfully breached.

Queso fingerprint	128 occurrences	Risk - Low

Queso fingerprint is a scan that attempts to guess the operating system and version on the target hosts. Queso collects the responses to various non-standard ip flags to determine what OS is responding.

CAN-1999-0454 A remote attacker can sometimes identify the operating system of a host based on how it reacts to some IP or ICMP packets, using a tool such as nmap or queso.

This detect scores a low risk. This is a reconnaissance attempt. It would be worthwhile to create a temporary watch list of the external networks attempting to gather data to see if there is further activity beyond the recon.

External RPC call	125 occurrences	Risk - Low

This alert is generated by calls to port 111 from external addresses. This was a scan for servers supporting RPC. There are 22 separate CVE listings for RPC vulnerabilities and 13 proposed candidates. It isn't possible to tell which particular CVE this RPC call might be related to.

This detect scores a low risk. These are attempts not successes so defenses were adequate.

Watchlist 000222 NET-NCFC	387 occurrences	Risk

This is the Computer Network Center Chinese Academy of Sciences network. This network was watch listed due to increased suspicious activity generated from the network in early 2000. This network is referenced in Guy Bruneau's practical available at http://www.sans.org/y2k/practical/Guy_Bruneau.doc

There is not a CVE associated with this watch list.

This detect scores a low risk. There is no clear threat and all attempted connections appear to have been blocked by current counter measures.

Back Orifice	109 occurrences	Risk - Low

This alert was generated by a scan for port 31337, the default Back Orifice port. Back orifice is a remote administration tool for Windows systems. It gives full remote access to a server and is frequently installed by crackers once a system is compromised. There is no evidence of successful Back Orifice connections on the network. Back Orifice is discussed in Teri Bidwell's practical available at http://www.sans.org/y2k/practical/Teri_Bidwell_GCIA.doc

There are no CVEs associated with this alert.

This alert scores a low risk. There are no successful connections indicating the defenses are adequate.

WinGate 1080 Attempt	80 occurrences	Risk - Low

This alert is triggered by attempted access to port 1080. This is a scan for WinGate proxy servers. WinGate is a popular proxy server for Windows environments. It has known weaknesses, many which are related to poor configuration. WinGate is also discussed in Teri Bidwell's practical available at http://www.sans.org/y2k/practical/Teri_Bidwell_GCIA.doc

The CVE candidate associated with this alert is CAN-1999-0657 WinGate is being used. There are other candidates related to vulnerabilities with WinGate.

This alert scores a low risk. There do not appear to be successful connections so defenses were adequate.

TCP SRC and DST outside network	76 occurrences	Risk - Low

This alert is generated by TCP packets that do not belong on this network. The alert is a sign of poor perimeter network management. If the packets are not sourced or destined for MY.NET network they are most likely forged. Previously compromised servers may be generating the packets. There are no CVEs for this alert.

This detect scores a low risk. These packets have no potential impact on our network. It would be potentially worthwhile to collect MAC data and try to identify what hosts are generating these packets.

Russia Dynamo - SANS Flash 28-jul-00	44 occurrences	Risk - Low

This alert is triggered on access from the Russian Network 194.87.0.0/16. The SANS site was not searchable so I could not locate this particular flash to determine why this network in particular should be watched. The activity from this network was attempted connections to port 317/tcp. A remote site maintenance product called ZanNet uses this port. ZanNet is a tool designed to work on a Unix server with a Windows client. The client can access the server's files as if they were a Windows network drive. A description of the product is available at http://www.zannet.com.

There is no CVE listing for Russia Dynamo or the ZanNet tool.

This alert scores a low risk. The attempts were not successful indicating the defenses were adequate. It would be worthwhile to create an alert to watch for attempted access to port 317/tcp to see if other networks are making similar attempts.

Port 55850 tcp - Possible myserver activity - ref. 010313-1	26 occurrences	Risk - Medium

This alert triggers on access to port 55850. Some alerts are normal high port usage. The only legitimate alerts are from communication between MY.NET.218.86 and 172.154.1.109. The host 172.154.1.109 is an America Online Host. The MY.NET host was connecting to port 55850 of the AOL host. Since the source port is unchanging it is possible this represents a successful connection.

This alert is not included in current versions of the Snort rule base. I have not been able to identify any product, application or trojan called myserver to determine why the connection would be alerted. There are no CVEs associated with myserver.

This alert scores medium risk, mostly because of the unknowns. There does appear to have been a successful connection from inside the network. Further investigation needs to be done to determine if the connection was supporting a valid business need.

Null scan!	26 occurrences	Risk - Low

This alert is triggered by FIN packets with all flags turned off. The null scan is a stealth scan designed to sneak past firewalls and IDS devices.

There are no CVEs listed for this scan.

This detect scores a low risk. This is a reconnaissance attempt. It would be worthwhile to create a temporary watch list of the external networks attempting to gather data to see if there is further activity beyond the recon.

NMAP TCP ping!	16 occurrences	Risk - Low

This alert is triggered by NMAP TCP ping. NMAP can send TCP ack packets instead of ICMP packets to map a network. The hosts that are up reply with resets.

There is no CVE listed for this alert.

This detect scores a low risk. This is a reconnaissance attempt. It would be worthwhile to create a temporary watch list of the external networks attempting to gather data to see if there is further activity beyond the recon.

SUNRPC highport access!	10 occurrences	Risk - High

This alert is triggered by successful access to the RPC ports 32771 and higher. The rpcbind and portmapper facilities have vulnerabilities that can allow root access. This access is from one host 216.136.171.195 to MY.NET .100.225.

CVE-1999-0168 - The portmapper may act as a proxy and redirect service requests from an attacker, making the request appear to come from the local host, possibly bypassing authentication that would otherwise have taken place. For example, NFS file systems could be mounted through the portmapper despite export restrictions.

CAN-1999-0195 Denial of service in RPC portmapper allows attackers to register or unregister RPC services or spoof RPC services using a spoofed source IP address such as 127.0.0.1.

This detect scores a high risk. There is possible root compromise and defenses were insufficient.

Tiny Fragments - Possible Hostile Activity	8 occurrences	Risk - Med

This alert is generated by TCP packets that are fragmented smaller than normal. The fragments can be a source of stealth communication or a means of slipping packets past firewall defenses.

There is not a CVE listing for this detect.

This detect scores a medium risk. It is unclear what the communication is trying to accomplish but our defenses did not prevent the tiny fragments from entering the network.

ICMP SRC and DST outside network	5 occurrences	Risk - Low

This alert is generated by ICMP packets that do not belong on this network. The alert is a sign of poor perimeter network management. If the packets are not sourced or destined for MY.NET network they are most likely forged. Previously compromised servers may be generating the packets. There are no CVEs for this alert.

This detect scores a low risk. These packets have no potential impact on our network. It would be potentially worthwhile to collect MAC data and try to identify what hosts are generating these packets.

connect to 515 from inside	2 occurrences	Risk - High

This alert is triggered by connections from MY.NET to external addresses. Port 515 is the lpd port a vector for many root exploits. A connection to an external address on this port may indicate a compromised server. The server making the connection is MY.NET.179.78 it is connecting to server 24.13.123.8.

There are numerous CVEs and candidates for lpd vulnerabilities. The vulnerabilities depend on type of operating systems being used.

This detect scores a high risk. The outbound connection does not appear to be blocked.

Top Talkers

The following ten IP addresses generated the most alerts on the network. They are listed with number and type of alerts. Notice that four of the top talkers do not have a source or destination on this network. Two of these four addresses are private networks that should not be routed externally.

IP Address	Number of Detects	Type of Detect
63.121.232.185	8926	Attempted Sun RPC high port access
212.179.4.50	6473	Watchlist 000220 IL-ISDNNET-990517
212.179.127.41	2160	Watchlist 000220 IL-ISDNNET-990517
10.0.0.1	1502	UDP SRC and DST outside of network
212.179.28.66	831	Watchlist 000220 IL-ISDNNET-990517
129.2.225.92	502	UDP SRC and DST outside of network
192.168.0.2	384	UDP SRC and DST outside of network
169.254.67.123	190	UDP SRC and DST outside of network
216.162.44.140	188	Connect to 515 from outside
24.162.245.198	109	Back Orifice

The top ten IP addresses that are scanning the network are included for completeness. The most disturbing information is that six of the top ten addresses scanning are in your local network.

IP Address	Number of scans	Type of scan
193.251.27.118	22269	FTP scan of 12003 targets
212.144.16.169	19589	FTP scan of 13030 targets
MY.NET.220.42	16860	Scan for ports 9xxx of 5492 targets

203.149.183.154	14897	DNS scan of 10969 targets
200.51.8.209	14683	Scan of port 555 (Phase 0 trojan) of 11129 targets
MY.NET.221.198	13103	Port 32768/UDP (filenet TMS) of 220 targets
MY.NET.227.206	9608	All over the map
MY.NET.218.102	9040	Scan for ports 9xxx of 4906 targets
MY.NET.217.222	7640	Scan for ports 9xxx of 2856 targets
MY.NET.218.86	6251	Scan for ports 6346 & 6347 (gnutella) on 140 targets

External Source Addresses to Watch

The host at 63.121.232.185 was the top talker on alerts. They generated 8926 alerts for RPC attempts. This is a UUNET customer with phone and email contact info. It would be worthwhile to contact them by phone so they can investigate this server.

Sigecom (NETBLK-UU-63-121-232)
6045 Wedeking Avenue
Evansville, IN 47715
US

Netname: UU-63-121-232
Netblock: 63.121.232.0 - 63.121.239.255
Maintainer: SIGE

Coordinator:
Wilkison, Chris (CW471-ARIN) cwilkison@sigecom.net 812-437-0530

The host at 193.251.27.118 was the top scanner, scanning for open FTP ports on 12003 target hosts. This is a RIPE network for France Telecom. They do have an abuse contact so it would be worthwhile to contact them.

inetnum: 193.251.0.0 - 193.251.95.255
netname: IP2000-ADSL-BAS
descr: France Telecom IP2000 ADSL BAS
descr: BAS for services FTI-1 and FTI-2
country: FR
admin-c: WITR1-RIPE
tech-c: WITR1-RIPE
status: ASSIGNED PA
remarks: for hacking, spamming or security problems send mail to
remarks: postmaster@wanadoo.fr AND abuse@wanadoo.fr
remarks: for ANY problem send mail to gestionip.ft@francetelecom.com
notify: gestionip.ft@francetelecom.com

The host at 212.144.16.169 was the second most active scanner, scanning for open FTP ports on 13030 hosts. This is a RIPE network assigned to a German company. There is no abuse contact or phone number contacting the hostmaster would probably be a dead end.

inetnum: 212.144.16.0 - 212.144.17.255
netname: O-TEL-O-IPBB
descr: o.tel.o GmbH
descr: Essen
country: DE
admin-c: RH10371-RIPE
tech-c: TW39-RIPE
status: ASSIGNED PA
notify: hostmaster@o-tel-o.de
mnt-by: OTELO-MNT

The host at 212.179.28.66 shows gnutella activity with MY.NET. 219.14 on Mar 22. The activity continues for 6 minutes. It is not possible to correlate this data with the OOS logs. The OOS log for Mar 23 shows data for Mar 23 the Alert log for Mar 23 has the previous day's data from Mar 22. From the pattern of alerts with the source port from host 212.179.28.66 remaining unchanged it is likely the communication was two sided.

```
03/22-04:17:07.372757  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.28.66:37074 ->
MY.NET.219.14:6346
03/22-04:17:07.383289  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.28.66:37074 ->
MY.NET.219.14:6346
>>> Additional log entries omitted
03/22-04:23:59.726696  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.28.66:37074 ->
MY.NET.219.14:6346
03/22-04:24:01.333738  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.28.66:37074 ->
MY.NET.219.14:6346
03/22-04:24:04.540949  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.28.66:37074 ->
MY.NET.219.14:6346
```

This host is in Israel and there is no abuse contact email or phone number.

inetnum: 212.179.28.64 - 212.179.28.127
netname: VSOFT
descr: VSOFT-LAN
country: IL
admin-c: NP469-RIPE
tech-c: NP469-RIPE
status: ASSIGNED PA
notify: hostmaster@isdn.net.il
mnt-by: RIPE-NCC-NONE-MNT

The host at 63.10.42.245 shows Ramen activity on Mar 26 in the alert log. There appears to be a continuing conversation for approximately 5 minutes. There are no corresponding records in the OOS logs.

```
03/26-01:48:47.330614  [**] Possible RAMEN server activity [**] 63.10.42.245:1375 ->
MY.NET.210.2:27374
03/26-01:52:32.232455  [**] Possible RAMEN server activity [**] 63.10.42.245:1375 ->
MY.NET.210.2:27374
>>> Additional log entries removed
03/26-01:53:13.192754  [**] Possible RAMEN server activity [**] 63.10.42.245:1375 ->
MY.NET.210.2:27374
03/26-01:53:13.731993  [**] Possible RAMEN server activity [**] 63.10.42.245:1375 ->
MY.NET.210.2:27374
```

UUNET Technologies, Inc. (NETBLK-NETBLK-UUNET97DU)
3060 Williams Drive, Suite 601
Fairfax, va 22031
US

Netname: NETBLK-UUNET97DU
Netblock: 63.0.0.0 - 63.63.255.255
Maintainer: UUDA

Coordinator:
UUNET, Technical Support (OA12-ARIN) help@uu.net
(800) 900-0241

The host at 216.162.44.140 is among the top ten scanners. They scanned one MY.NET subnet MY.NET.134.0/24. The scan was for port 515, which may mean this host has been compromised and is attempting to spread its infection. The network has both email and phone contact, it would be worthwhile to call the coordinator and let him know of the problem.

iNET Systems Corp (NETBLK-INET-SYS)
430 Tenth Street NW South 102
Atlanta, GA 30318
US

Netname: INET-SYS
Netblock: 216.162.32.0 - 216.162.47.255
Maintainer: INET

Coordinator:
Wall, Gary (GW429-ARIN) gary.wall@INET-SYSTEMS.COM
+1-404-522-1088 (FAX) (404)874-5419

The host at 200.51.8.209 scanned the network extensively looking for the Phase 0 trojan. This may be another infected host trying to spread the joy. The contact information does include a noc contact. It would be worthwhile to attempt email communication.

People's Network Argentina S.A. (NETBLK-PEOPLES-AR)
Beruti 2770

Buenos Aires, Buenos Aires 1425
AR

Netname: PEOPLES-AR
Netblock: 200.51.8.0 - 200.51.9.255

Coordinator:
Tld, Poc (PT92-ARIN) noc@telintar.net.ar
54-11-4370-1555 (FAX) 54-11-4373-9341

The host at 158.75.57.4 was particularly interested in our network. This host stealth scanned the network and ran Queso using the gnutella port. It would be interesting to contact the network contact and find out why there is so much interest in our network. The network is in Poland so the likelihood of a meaningful dialogue is slim.

POLIP (NET-TORUNPOLIP2)
Computer Centre, Nicolaus Copernicus University
ul. Chopina 12/18, 87-100 Torun, Poland
PL

Netname: TORUNPOLIP2
Netblock: 158.75.0.0 - 158.75.255.255

Coordinator:
Szewczak, Zbigniew S. (ZSS-ARIN) zssz@TORUN.PL
(56) 260-17 ext. 70

The host at 129.206.170.20 also showed particular interest in our network. Starting with a stealth scan and moving to a Queso fingerprint attempt using the gnutella port. This host concentrated on the host MY.NET.202.54. This network is in Germany so we may have limited success contacting the network administrator.

University of Heidelberg (NET-HD-NET)
Im Neuenheimer Feld 293
D-69120 Heidelberg,
DE

Netname: HD-NET
Netblock: 129.206.0.0 – 129.206.255.255

Coordinator:
Hebgen, Michael (MH255-ARIN) michael.hebgen@URZ.UNI-HEIDELBERG.DE
+49 6221 54-4501 (FAX) +49 6221 54-5581

The host at 62.31.68.89 scanned 466 hosts looking for open RPC ports. The network is registered to the Cable Internet in Great Britain. There is an email contact.

```
inetnum:     62.31.40.0 - 62.31.78.255
netname:     HSD-UDD
descr:       Uddingston HSD platform
country:     GB
admin-c:     MG645-RIPE
tech-c:      SB264-RIPE
status:      ASSIGNED PA
mnt-by:      RIPE-NCC-NONE-MNT
changed:     mike@cableinet.net 20000328
```

Out of Spec Traffic

Following is the analysis of the out of spec traffic that does not appear in the alerts or port scans. A link diagram accompanies traffic that involves multiple ports. Much of the out of spec traffic appears to involve various scans and fingerprinting attempts against MY.NET hosts.

One interesting feature of many of the scans and out of spec traffic is the use of port 6346 as a destination. This is the well-known port for gnutella. Much of the traffic does not appear to be gnutella but possibly crackers using the port to hide their activities. They may be counting on the network administrator to ignore the activity to gnutella as a known nuisance. It could be advantageous to keep a close eye on traffic for 6346.

The host 62.31.68.89 scanned the network for port 111 (RPC). The scan was a SYN/FIN scan and for some reason did not appear in the Alerts or Snort Scan logs. These packets from the out of spec logs show the SF flags set. There were 466 separate probes.

>>> Additional packets removed

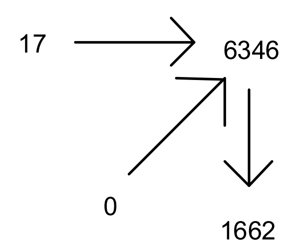
```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/23-10:48:41.502370 62.31.68.89:111 -> MY.NET.132.217:111
TCP TTL:26 TOS:0x0 ID:39426
**SF**** Seq: 0x2B7CD4B   Ack: 0x2FD996F7   Win: 0x404
02 04 05 B4 01 03                             ......
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/23-10:48:41.541021 62.31.68.89:111 -> MY.NET.132.219:111
TCP TTL:26 TOS:0x0 ID:39426
**SF**** Seq: 0x2B7CD4B   Ack: 0x2FD996F7   Win: 0x404
48 54 54 50 2F 31                             HTTP/1
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/23-10:48:41.640212 62.31.68.89:111 -> MY.NET.132.222:111
TCP TTL:26 TOS:0x0 ID:39426
**SF**** Seq: 0x2B7CD4B   Ack: 0x2FD996F7   Win: 0x404
00 00 00 00 00 00
```

>>> Additional packets removed

The host 63.100.208.92 from the New Skies Satellite network scanned port 80 of a single host MY.NET.253.125 for 6 minutes. The packets all had the reserved bits and SYN flag set. The other characteristics of the packet appeared to be normal. The sequence numbers incremented normally and there were no unusual options. This could be a crack attempt or just a misguided connection attempt. It would be worthwhile to contact the network administrator and investigate further.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/23-17:11:51.558394 63.100.208.92:2306 -> MY.NET.253.125:80
TCP TTL:47 TOS:0x0 ID:0  DF
21S***** Seq: 0x9225266B   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 1959587 0 EOL EOL EOL EOL
>>> Additional Packets removed
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/23-17:17:38.094552 63.100.208.92:2565 -> MY.NET.253.125:80
TCP TTL:47 TOS:0x0 ID:0  DF
21S***** Seq: 0xA88ED840   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 1994238 0 EOL EOL EOL EOL
```

Following is some activity between 209.53.61.36 and our host MY.NET.203.222. The conversation is one way from 209.53.61.36, a host in the Telus Advanced Communications network.

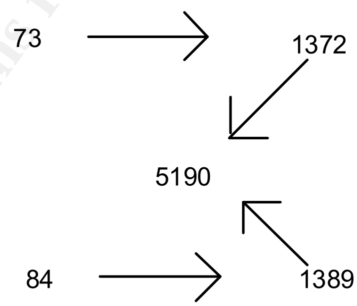


The first packet is from port 17 (quote of the day) to port 6346 (gnutella). The packet appears to contain the beginning of an email address Oz@. The packet also has anomalous flags SYN, FIN, RESET and URGENT are all set. Several minutes later 209.53.61.36 sends a packet from port 6346 to port 1662 on MY.NET. The flags SYN, FIN, RESET, PUSH, ACK and URGENT are all set on this packet. The Telus host then sends a second packet to port 1662 this time with the flags SYN, FIN, ACK and URGENT set along with TCP opt 253. The last packet from the Telus host is from port 0 to 6346 with SYN and FIN set. The packets are most probably crafted, port 0 is not an assigned port. From the use of unusual flags this appears to be a fingerprinting attempt.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/27-09:08:03.494864 209.53.61.36:17 -> MY.NET.203.222:6346
TCP TTL:109 TOS:0x0 ID:15910  DF
**SFR**U Seq: 0x67E015E   Ack: 0xC78C2A0   Win: 0x5018
00 11 18 CA 06 7E 01 5E 0C 78 C2 A0 08 27 50 18  .....~.^.x...'P.
0C C8 52 9D 00 00 75 FE 4F 7A D4 40 B4 F9 D1 B2  ..R...u.Oz.@....
2A 2B                                            *+
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/27-09:16:57.153339 209.53.61.36:6346 -> MY.NET.203.222:1662
TCP TTL:109 TOS:0x0 ID:46008  DF
**SFRPAU Seq: 0x161FBBB   Ack: 0xC8C2A5   Win: 0x5018
30 3F 50 18 0E AE 23 1B 00 00 63 9A 2D 0F CE 2A  0?P...#...c.-..*
8B 4B AB 97 F0 86                                .K....
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/27-09:17:47.711385 209.53.61.36:6346 -> MY.NET.203.222:1662
TCP TTL:109 TOS:0x0 ID:63687  DF
2*SF**AU Seq: 0x1625DD1   Ack: 0xDCC2A5   Win: 0x5018
TCP Options => EOL EOL Opt 253 (9): E67E D463 D5F9 FF62
99 E5                                   ..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/27-09:22:04.707573 209.53.61.36:0 -> MY.NET.203.222:6346
TCP TTL:109 TOS:0x0 ID:16  DF
21SF**** Seq: 0x67E0164   Ack: 0x851CC2A7   Win: 0x5010
6C 6F
```

Following is some unusual activity between MY.NET.208.226 and host 129.2.249.90 in the University of Maryland network. The communication is seemingly random, but when diagrammed it appears that there may be a common communication point at port 5190. Port 5190 is officially the AOL instant message port.



It is possible that this random collection of ports is being used for stealth communication. As seen in the traffic from the out of spec logs. MY.NET.208.226 sends a packet to port 1372 with flags SYN, PUSH and ACK set. Seconds later a packet is sent from port 1372 to port 5190 with the same anomalous flags set. This packet also has TCP options set of opt 30. A series of packets are sent from MY.NET.208.226 port 1389 to 129.2.249.90 port 5190. These packets have all flags set except URGENT. The last packet of this type also has TCP options set of opt 30. The next packet in the sequence is from MY.NET.208.226 port 84 to 129.2.249.90 port 1389. A few seconds later the last packet is sent from MY.NET.208.226 port 1389 to 129.2.249.90 port 5190. For these last two packets all flags except URGENT are set.

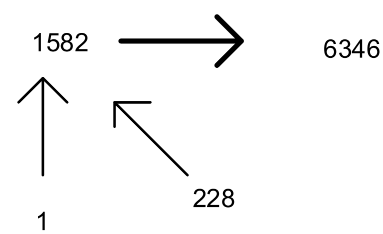
This may be a type of stealth communication. It is too much of a coincidence for our MY.NET host to send information to port 1372 on 129.2.249.90 and then randomly choose 1372 to connect to the same host on port 5190. I cannot tell from the data available in the log what, if anything, is being communicated.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/26-21:03:37.098356 MY.NET.208.226:73 -> 129.2.249.90:1372
TCP TTL:126 TOS:0x0 ID:19498  DF
21S**PA* Seq: 0x14460018   Ack: 0x3D6F01B5   Win: 0x5010
3B DA 50 10 22 38 2B 4F 20 20 20 20 20 00        ;.P."8+O     .
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/26-21:03:40.958417 MY.NET.208.226:1372 -> 129.2.249.90:5190
TCP TTL:126 TOS:0x0 ID:35886  DF
21S**PA* Seq: 0x540018   Ack: 0x3D6F01D7   Win: 0x5010
TCP Options => Opt 32 (32): 2020 2000 0300 0101 080A 007C 0000 0000 0000 0000 0000 0000 0000
0000 0000 EOL EOL EOL EOL EOL EOL EOL EOL
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/26-21:12:12.355920 MY.NET.208.226:1389 -> 129.2.249.90:5190
TCP TTL:126 TOS:0x0 ID:18738  DF
21SFRPA* Seq: 0x201E27   Ack: 0x8701BB   Win: 0x5010
20 20 20 20 20 00                             .
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/26-21:12:14.081579 MY.NET.208.226:1389 -> 129.2.249.90:5190
TCP TTL:126 TOS:0x0 ID:32564  DF
21SFRPA* Seq: 0x201E27   Ack: 0x8601CC   Win: 0x5010
00 20 1E 27 00 86 01 CC 18 DF 50 10 22 38 6D 62  . .'......P."8mb
20 20 20 20 20 00                                .
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/26-21:12:15.052044 MY.NET.208.226:1389 -> 129.2.249.90:5190
TCP TTL:126 TOS:0x0 ID:51765  DF
21SFRPA* Seq: 0x3D0020   Ack: 0x1E2701D5   Win: 0x5010
TCP Options => Opt 32 (32): 2020 2000 0402 3031 3233 3435 0000 0000 0000 0000 0000 0000 0000
0000 0000 EOL EOL EOL EOL
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/26-21:12:18.969510 MY.NET.208.226:84 -> 129.2.249.90:1389
TCP TTL:126 TOS:0x0 ID:63545  DF
21SFRPA* Seq: 0x14460020   Ack: 0x1E2701F5   Win: 0x5010
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/26-21:12:21.243550 MY.NET.208.226:1389 -> 129.2.249.90:5190
TCP TTL:126 TOS:0x0 ID:65084  DF
21SFRPA* Seq: 0xD00020   Ack: 0x1E27020C   Win: 0x5010
```

On March 27 there is some unusual activity between host 192.38.4.249 from the Danish Computer Center for Research and Education and host MY.NET.225.182. The destination for most of the traffic is the well-known port for gnutella, however the activity is not gnutella.

The first two packets from port 1582 to port 6346 have all TCP flags set and the reserved flag. This would appear to be a Christmas tree scan. Over an hour later there is packet from port 1 to port 1582 of the MY.NET host with the same flags set.



Almost two hours after the initial packet from 192.38.4.249 another packet from port 1582 to port 6346 is sent, this time with another odd assortment of flags. The flags for this packet are SYN, FIN, ACK and URGENT. The last packet we see from 192.38.4.249 is from port 228 to port 6346 almost 20 minutes after the previous packet with all flags set.

There are a few unusual aspects to this traffic. The anomalous flags would appear to be a scan attempt. But why such a slow scan to one host? The alert and scan logs stop at March 26 so there is no way to correlate this data with those logs. The packets appear to be crafted the source port is 1582 at 3:54 and also at 5:05 when the conversation supposedly continues. With no intervening packets this would appear to be a new communication. Or is the record incomplete? There is not enough information in the available packets to determine if this is simply a scan, some stealth communication or a snippet of a larger conversation. The hosts could bear further scrutiny.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/27-03:54:10.974627 192.38.4.249:1582 -> MY.NET.225.182:6346
TCP TTL:116 TOS:0x0 ID:59486  DF
2*SFRPAU Seq: 0x28AB3E   Ack: 0xB68C34   Win: 0x5010
```

```
TCP Options => EOL EOL EOL EOL EOL EOL
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/27-03:54:21.260560 192.38.4.249:1582 -> MY.NET.225.182:6346
TCP TTL:116 TOS:0x0 ID:45663  DF
2*SFRPAU Seq: 0x28   Ack: 0xAB3E8C36   Win: 0x5010
1E 38 C5 5C 00 00 00 00 00 00                .8.\......
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/27-05:05:39.420116 192.38.4.249:1 -> MY.NET.225.182:1582
TCP TTL:116 TOS:0x0 ID:55350  DF
2*SFRPAU Seq: 0x18CA0028   Ack: 0xAB3E8E99   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/27-05:34:33.857972 192.38.4.249:1582 -> MY.NET.225.182:6346
TCP TTL:116 TOS:0x0 ID:9096  DF
**SF**AU Seq: 0xBC0028   Ack: 0xAB3E8F87   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/27-05:51:31.615834 192.38.4.249:228 -> MY.NET.225.182:1582
TCP TTL:116 TOS:0x0 ID:36871  DF
2*SFRPAU Seq: 0x18CA0028   Ack: 0xAB3E9013   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK NOP NOP TS: 0 0 EOL EOL EOL EOL EOL
EOL EOL
```

Internal Machines to Watch

Analysis of the network traffic from March 23 through March 27 reveals some potentially compromised hosts and some suspect communication with other networks. The following internal machines need to be analyzed further. The hosts are listed by level of urgency. Hosts that are probably compromised are at the top, hosts that are merely suspect are at the bottom.

The first group of hosts that should be audited are the four hosts that are the source for Ramen or SubSeven traffic. The listed hosts are attempting to make connections to port 27374 to hosts external to the network. MY.NET.222.154, MY.NET.6.47, MY.NET.97.93, and MY.NET.98.171. These connection attempts make it exceedingly likely that the MY.NET hosts have already been infected with SubSeven or Ramen.

A host on Exodus' network, 216.136.171.195, successfully connects to the SUNRPC high port 32771 on host MY.NET.100.225. This is a serious incident and may indicate a root compromise on the MY.NET host. The host needs to be audited and the firewall rules need to be examined to determine why access to SUNRPC was allowed through.

The host MY.NET.179.78 is connecting to port 515 on the external host 24.13.123.8. This host is on the @home network. It is possible this is someone printing from a connection on MY.NET to their host connected to @home. This connection needs to be investigated to determine if this traffic was legitimate.

The host MY.NET.218.86 has some suspicious connections. This server is alerting on the possible myserver connections. Although we were unable to determine what protocol the myserver alert is concerned with. The connections themselves are suspicious. The host MY.NET.218.86 always connects to the destination port 55856 from source port 2000. The

connections are with two hosts,172.154.1.109 in the AOL network and 213.44.175.50 in the Online France Club Internet. The use of the same source port indicates a possibly crafted packet. The host should be audited to determine if it is running an illicit service.

There is a strong possibility that the host MY.NET.219.14 has been sharing files via gnutella with a host in Israel 212.179.28.66. The following two lines from the alert log represent the first and last entry of 831 similar alerts. The source port for the host 212.179.28.66 remains the same indicating that this is an ongoing communication. Further investigation should be done to determine what files were transferred. It is strongly recommended that gnutella be blocked at the firewall.

```
03/22-04:17:07.372757  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.28.66:37074 ->
MY.NET.219.14:6346
>>> Additional alerts removed
03/22-04:24:04.540949  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.28.66:37074 ->
MY.NET.219.14:6346
```

The hosts MY.NET.97.180 and MY.NET.217.174 spend considerable time scanning for port 6112. This is a gaming port used for both Diablo and Starcraft games. These are not business-related activities.

Defensive recommendations

There are some simple steps to improve perimeter defenses. There would be a significant impact by following recommended practices as given in CERT advisory CA-1996-21 http://www.cert.org/advisories/CA-1996-21.html. Appendix A has recommendations for filtering at the edge routers. Applying ingress filters that only permit traffic destined for the MY.NET network and drop any packets with a source destination of one of the private or IANA reserved networks would eliminate the traffic not related to MY.NET. Installing egress filters that only allow traffic with a source of MY.NET would eliminate spoofed packets from being generated within the network.

In addition to adding standard filters the perimeter defenses need to be evaluated to ensure that only required business activity is permitted. There is evidence that the current configuration allows gnutella and napster. It is doubtful that these protocols are being used for business purposes. The firewall rules need to be evaluated to see if undesirable protocols have crept into the allowed rules and to make sure that security policy is being implemented. Rules to eliminate the non-business traffic should be added. Personnel should be notified of the security policy and systems that are in violation of the policy should be examined for evidence of compromise.

The MY.NET network is a sizable network. It could improve defenses by reviewing network topology and possibly deploying new internal firewalls to segment internal traffic. This could contain the spread of worms or other malign software. Any business area that requires substantially different access or has privacy requirements for data should be considered for an internal firewall. Areas to consider for additional protection could be human resources, accounting and development.

Internal hosts that were identified as possibly compromised need to be examined and where necessary rebuilt. Rebuilding a host would be necessary if there is any doubt that the host has been compromised. Procedures should be put in place to systematically audit all hosts within the network. Starting with hosts that are on the same network as compromised hosts. Hosts should be audited on an ongoing basis to identify anomalous file changes, addition of new services or software. Ongoing audits would potentially find a compromised host before it could spread the malign software throughout the network.

The snort rules that were used to generate the alerts appear to be several revisions out of date. Some rules are no longer included in current versions of the snort rule base. If the rules being used are out of date its highly probable that new signatures have not been installed and attacks are being overlooked. The snort rules and possibly the snort code need to be updated. For this network it may be worthwhile to add an additional rule alerting on potential gnutella traffic. The use of port 6346 for reconnaissance is suspicious and bears tracking.

Analysis methodology

The analysis process for the MY.NET traffic involved downloading and modeling the data and correlation of logs. The alerts, out of spec and scan logs from March 23 to March 27 was downloaded from the web site onto a Solaris workstation.

On the alert files the header information was removed, the logs were concatenated and the resulting combined log was processed through SnortSnarf release 5/23/01. SnortSnarf is a snort log processing tool that takes alert logs as input and outputs html pages. The html pages arrange the alerts by type sorted by number of occurrences. Each alert type has underlying pages that drill down through source and destination for each alert, ultimately ending up at the original alert. SnortSnarf is very useful for processing large amounts of alert data. In order to pull meaningful IP addresses out of SnortSnarf for the MY.NET hosts the prefix MY.NET was changed to the private network 172.17. This network did not appear in any of the other data.

The out of spec data required a two step process to analyze. First, I extracted just the barebones of traffic the date, time, source and destination. I used a simple perl script to process each OOS file and create a comma separated output file. The script was used as part of a pipeline just cat the OOS file to the perl script and redirect the output to a file.

```
#!/usr/bin/perl
#
while (<>){
        if (/^\d\d\/\d\d/){
        ($time, $temp1, $null, $temp2, $proto)  = split(" ", );
        ($src, $src_port) = split (/:/, $temp1);
        ($dest, $dest_port) = split (/:/, $temp2);
        print ($time,",", $src,",",$src_port,",",$dest,",",$dest_port,"\n");
        }

}
```

The resulting output was imported to a spreadsheet where the data was sorted by source address and then by destination address to identify any trends. Once hosts of interest were identified the full out of spec logs were revisited. The full description was used for in-depth analysis. The following script was used to pull out the full output for each packet. The $IP would be replaced with the IP address to match.

```
nawk 'BEGIN { FS ="\n"; RS = "" } /$IP/ {print $0}' OOS*
```

The SnortScan logs were processed with the following perl script to create a comma-delimited file from the SnortScan files.

```
#!/usr/bin/perl

while (<>){

        ($mon, $day, $time, $temp1, $null, $temp2, $proto)  = split(" ", );

        ($src, $src_port) = split (/:/, $temp1);
        ($dest, $dest_port) = split (/:/, $temp2);
        print ($mon," ",$day,",",$time,",", $src,",",$src_port,",",$dest,",",$dest_port,"\n");

}
```

The resulting files were loaded into a spreadsheet and sorted similarly to the out of spec logs. The count feature of the database was used to identify the top scanners.

When possible data was correlated between the logs to identify hosts that were generating multiple types of bad behavior.