Global Information Assurance Certification Paper

Intrusion Detection In Depth
GCIA Practical Assignment
Version 2.9
Current as of May 22, 2001

Ben Thomas

Network Detects

Network Detect 1.

(Security@auckland)

On Fri 06 Apr 2001 at 13:20 (UTC) we detected a scan of tcp-23 ports in part of our network. This incident appears to have originated from 217.57.19.30. Immediately after the scan an attempt was made to compromise one system using the sgi telnet bug. A few hours later several machines were attacked via ftp (see appended logs). Sample logs, times are UTC + 1200, GPS synchronized:

```
07 Apr 01 01:20:49      tcp    217.57.19.30.1711  <|    130.216.4.179.23    sR
07 Apr 01 01:20:49      tcp    217.57.19.30.1713  <|    130.216.4.181.23    sR
07 Apr 01 01:20:49      tcp    217.57.19.30.1728  <|    130.216.4.194.23    sR
07 Apr 01 01:20:55      tcp    217.57.19.30.2347  <|     130.216.7.41.23    sR
07 Apr 01 01:20:55      tcp    217.57.19.30.2384  <|     130.216.7.78.23    sR
07 Apr 01 01:20:55      tcp    217.57.19.30.2390  <|     130.216.7.84.23    sR
07 Apr 01 01:20:55      tcp    217.57.19.30.2391  <|     130.216.7.85.23    sR
07 Apr 01 01:20:55      tcp    217.57.19.30.2394  <|     130.216.7.88.23    sR
07 Apr 01 01:20:56      tcp    217.57.19.30.2419  <|    130.216.7.113.23    sR
07 Apr 01 01:20:56      tcp    217.57.19.30.2423  <|    130.216.7.117.23    sR

Apr  7 01:43:12 takahe snort[64292]: IDS304 - TELNET - SGI telnetd format bug:
  217.57.19.30:4183 -> 130.216.21.9:23
Apr  7 04:42:34 takahe snort[64292]: IDS440 - FTP - wuftp260 Linux venglin
  parbobek: 217.57.19.30:2707 -> 130.216.7.14:21
Apr  7 04:42:47 takahe snort[64292]: IDS440 - FTP - wuftp260 Linux venglin
  parbobek: 217.57.19.30:2710 -> 130.216.7.21:21
Apr  7 04:43:02 takahe snort[64292]: IDS440 - FTP - wuftp260 Linux venglin
  parbobek: 217.57.19.30:2712 -> 130.216.7.11:21
Apr  7 04:43:11 takahe snort[64292]: IDS440 - FTP - wuftp260 Linux venglin
  parbobek: 217.57.19.30:2714 -> 130.216.15.8:21
Apr  7 04:43:31 takahe snort[64292]: IDS440 - FTP - wuftp260 Linux venglin
  parbobek: 217.57.19.30:2718 -> 130.216.93.9:21
Apr  7 04:43:32 takahe snort[64292]: IDS317 - FTP-site-exec: 217.57.19.30:2718 ->
  130.216.93.9:21

Source: 217.57.19.30
Ports: tcp-23, 21
Incident type: Network_scan, exploit attempts
re-distribute: yes
timezone: UTC + 1200
reply: no
Time: Fri 06 Apr 2001 at 13:20 (UTC)
```

Source of trace:

GIAC http://www.sans.org/y2k/041001-1600.htm

Detect was generated by

Snort IDS (http://www.snort.org). Portscan was probably detected by the Snort Portscan preprocessor, and the alerts were sent to the syslog daemon running on the host "takahe". Fields are <date> <time> <source IP address>:<source port> <direction> <dest IP address>:<dest port> <TCP flag(s) (ASCII)> <TCP flag bits>.

Syslog entry format is: <Date> <Time> <syslog host name> <application name>[<PID>]: <message text>

Alert rules triggered were probably (lines may be wrapped for readability):

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg:"TELNET SGI telnetd format
bug"; flags: A+; content: "_RLD"; content: "/bin/sh";reference:arachnids,304;)
```

Causes snort to "alert" (i.e. send a syslog message in this case) on any TCP packet from any external address, any source port to any internal address (address on the network being protected), port 23 (telnet), that has the ACK flag (and any others) set. Contents of the packet must contain the string "_RLD" and "/bin/sh". "msg" is the description, and "reference" is a reference to IDS304 in the arachnids database (http://whitehat.com/info/IDS304)

```
alert TCP $EXTERNAL_NET any -> $HOME_NET 21 (msg:
"IDS440/ftp_ftp-wuftp260-linux-venglin-parbobek"; flags: A+; content:
"|2e2e3131|venglin@";)
```

Causes snort to "alert" on any TCP packet from an external address, any source port to an internal address, port 21 (ftp), that has the ACK flag (and any others) set. The payload of the packet contains the string 0x2e2e3131 concatenated with the string "venglin@" (case sensitive).

```
alert TCP $EXTERNAL_NET any -> $HOME_NET 21 (msg: "IDS317/ftp_ftp-site-exec";
flags: A+; content: "site exec"; nocase;)
```

Causes snort to "alert" on any TCP packet from an external address, any source port to an internal address, port 21 (ftp), that has the ACK flag (and any others) set. The payload of the packet contains the string "site exec" in either upper or lower case, or a mix of both.

The first rule was taken from 'telnet.rules' from www.snort.org, and the other two from whitehats.com IDS signature database.

Probability the source address was spoofed

Probably not spoofed.

It is unlikely that the source address is spoofed as this appears to be a scan followed by an exploit attempt. In order for the scan to be effective, the attacker needs to receive packets back on the TCP connection.

Also, from the description of IDS304 at whitehats.com:

> The packet that caused this event is normally a part of an established TCP session, indicating that the source IP address has not been spoofed. If you are using a firewall that supports stateful inspection, and are not vulnerable to sequence number prediction attacks, then you can be fairly certain that the source IP address of the event is accurate. It has been noted that the intruder is likely to expect or desire a response to their packets, so it may be likely that the source IP address is not spoofed.

As we cannot be certain about the state of the network under attack, we cannot be absolutely sure, but I put my quarter on this not being spoofed.

The source of the attack…

From RIPE:

```
217.57.19.30


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


inetnum:      217.57.19.0 - 217.57.19.31
netname:      CDC-COMPUTER-DATA-CONTROL
descr:        CDC COMPUTER DATA CONTROL
country:      IT
admin-c:      EP4295-RIPE
tech-c:       EP4295-RIPE
status:       ASSIGNED PA
notify:       network@cgi.interbusiness.it
mnt-by:       INTERB-MNT
changed:      network@cgi.interbusiness.it 20010321
source:       RIPE
```

And the destination of the attack is Auckland University:

```
Auckland University (NET-AUKUNI-NET)
        IT Center,  24 Symonds Street
        Auckland, 1001
        NZ

        Netname: AUKUNI-NET
        Netblock: 130.216.0.0 - 130.216.255.255

        Coordinator:
           ITSS, The Auckland University  (TAI1-ARIN)
postmaster@Auckland.AC.NZ
```

+64 9 373 7599 ext. 8955

Description of attack

This appears to be a fast portscan of a (possibly) Class B network, looking for open telnet ports, followed by attempted exploits against the telnet port of an SGI system (IDS304, CVE-2000-0733, Bugtraq 1572, AdvICE 2000908). And, much later, attempted FTP exploits against some Linux systems running the wu-ftp daemon, that were likely identified by the scan (IDS440, CVE: CAN-2000-0574, Bugtraq 1387, AdvICE 2001307 and IDS317, CVE-1999-0080, AdvICE 2001322).

The portscan appears to be a stimulus, judging from the SYN/FIN sent in each TCP packet (assuming that the "sR" at the end of each line means that the SYN and RESET flags are set in the TCP header).

Attack mechanism

Examination of the time and port numbers of the source address, this appears to be a fast, widespread scan of a large network. Without knowing more about the placing of the sensor and the destination addresses being monitored, we can only surmise that we are seeing part of what was probably a much larger scan. Note that the source ports increase almost arithmetically with the target IP address, with a gap of about 6 secs between the addresses 130.216.4.x and 130.216.7.x and a corresponding gap in source port numbers.

The difference between the last source port and the first source port used is 712, which suggests 712 addresses (or more) were scanned in this session. This is very close to the total number of available addresses between 130.216.4.179 and 130.216.7.117, viz

130.216.4.179 – 130.216.4.255 = 76 addresses
130.216.5.x = 255 addresses
130.216.6.x = 255 addresses
130.216.7.1 -- 130.216.7.117 = 117 addresses
for a total of 703 addresses

So this looks like a fast scan of a large network probing port 23 (telnet) and probably reading back the banner information from the returned packet. This would yield useful information to an attacker, most likely by identifying the OS of the targeted system that is running a telnet daemon. For example, this is what is returned when I establish a conventional connection to one of our internal hosts:

```
Trying...
Connected to 0.
Escape character is '^]'.
Local flow control on
```

```
Telnet TERMINAL-SPEED option ON

HP-UX hp20 B.11.00 C 9000/889 (tb)

login:
```

This clearly shows that my system is running HP-UX version 11.00, and, thus, may be vulnerable to any HP-UX exploits.

The first attempted exploit occurred 23 minutes after the portscan was detected. This suggests that the attacking program had either to complete that scan before it would attempt any exploits, or that the output was read by the attacker, who then launched the SGI telnetd exploit (IDS304) against 217.57.19.30. It is reasonable to assume, then, that this (target) system was probably an SGI system with an open telnet port, or a system that was made to look like an SGI system.

Further evidence that the results of the scan had to be manually compiled is the delay between the scan cycle and the attempted exploits against the Linux systems running wu-ftpd. The scan seems to have found an older version (2.4) of wu-ftpd running on the host at 130.216.93.9, judging from the attempted exploit (IDS317) tried on this system. From the description at *whitehats.com*:

> This event indicates that an attempt has been made to exec a command on an ftp server. Some old versions of wu-ftpd 2.4 and earlier are vulnerable to remote compromise due to poor security restrictions of the site exec command.

Note that there could have been other traffic from this host that was not detected.

Correlations

Checking the searchable database as DSHIELD (http://www1.dshield.org/ipinfo.php?ip=217.57.19.30), it appears that this host has been implicated in a larger number of attacks against various hosts – starting on 6th June, 2001. All these detects are against port 21 (FTP).

As the Dshield database is a relatively new endeavour, it is possible that this host has been used as an attacking host since long before this date.

From the Dshield database:

```
IP Address:  217.57.19.30
HostName:    217.57.19.30
DShield Profile:
Country:      IT
Contact E-mail: ginocchi@cgi.interbusiness.it
Total Records against IP: 879
```

```
Number of targets: 249
Date Range: 2001-06-06 to 2001-06-19
```

There are no firewall logs or syslog data from any of the targeted systems available, which may have helped us correlate this event.

Evidence of active targeting

The portscan phase of this attack was clearly NOT targeted at specific hosts, but at (possibly) the whole class B network. The attempted exploits, though, were very definitely targeted at specific hosts, presumably identified in the scan.

Severity

This is difficult to determine, so this is just a guess…

Criticality: 4
    This is a large network with a mix of systems
Lethality: 5
    The exploit would give root access
Network countermeasures: 3
    This is a university, so freedom is valued above security.
System countermeasures: 3
    We have no idea, but presumably, these hosts were not actually compromised… or where they????

    (4 + 5) – (3 + 3) = +3

Defensive recommendation

Keep the firewalls in place, with patches kept up to date. Ensure that all systems are hardened as much as possible. Install TCP wrappers with an allow-these-sites-only policy. Replace the telnet banner with a banner that does not give away any OS information – this applies to other applications, such as FTP as well.

If possible, replace telnet and ftp with SSH and sftp.

Employ more IDS sensors on the network.

Multiple choice test question

In the following trace, what is the significance of the number in brackets [64292] (lines may be wrapped for readability):

```
Apr  7 04:43:31 takahe snort[64292]: IDS440 - FTP - wuftp260 Linux venglin
parbobek: 217.57.19.30:2718 -> 130.216.93.9:21

a)    The number of times this event has been detected
b)    The process id of the snort application
c)    The arachnids reference number for this exploit
d)    The source port used

      Answer is b)
```

Network Detect 2.

```
(Laurie@.edu)
 =-=-=-=-=-=-=-=-=-=-=
Server used for this query: [ whois.arin.net ]
         California Regional Internet, Inc. (NETBLK-CARI)
         8929A COMPLEX DRIVE SAN DIEGO, CA 92123 US
         Netname: CARI
         Netblock: 209.126.128.0 - 209.126.175.255
         Maintainer: CALI

 Apr 13 11:48:25 209.126.168.231:4504 -> a.b.c.114:53 SYN ******S*
 Apr 13 11:48:25 209.126.168.231:4597 -> a.b.c.207:53 SYN ******S*
 Apr 13 11:48:28 209.126.168.231:4420 -> a.b.c.30:53 SYN ******S*
 Apr 13 11:48:28 209.126.168.231:4441 -> a.b.c.51:53 SYN ******S*
 Apr 13 11:48:28 209.126.168.231:4461 -> a.b.c.71:53 SYN ******S*
 Apr 13 11:48:28 209.126.168.231:4472 -> a.b.c.82:53 SYN ******S*
 Apr 13 11:48:28 209.126.168.231:4557 -> a.b.c.167:53 SYN ******S*
 Apr 13 11:48:28 209.126.168.231:1388 -> a.b.c.225:53 SYN ******S*
 Apr 13 11:48:28 209.126.168.231:1107 -> a.b.c.225:53 UDP
 Apr 13 11:48:28 209.126.168.231:1407 -> a.b.c.244:53 SYN ******S*
 Apr 13 11:48:31 209.126.168.231:1528 -> a.b.d.52:53 SYN ******S*
 Apr 13 11:48:31 209.126.168.231:1678 -> a.b.d.202:53 SYN ******S*
 Apr 13 11:48:31 209.126.168.231:1928 -> a.b.e.195:53 SYN ******S*
 Apr 13 11:48:31 209.126.168.231:1947 -> a.b.e.214:53 SYN ******S*
 Apr 13 11:48:32 209.126.168.231:1952 -> a.b.e.219:53 SYN ******S*
 Apr 13 11:48:35 209.126.168.231:1971 -> a.b.e.238:53 SYN ******S*
 Apr 13 11:48:37 209.126.168.231:2938 -> a.b.f.145:53 SYN ******S*
 Apr 13 11:48:40 209.126.168.231:2941 -> a.b.f.148:53 SYN ******S*
 Apr 13 11:48:37 209.126.168.231:2942 -> a.b.f.149:53 SYN ******S*
 Apr 13 11:48:40 209.126.168.231:2947 -> a.b.f.154:53 SYN ******S*
 Apr 13 11:48:37 209.126.168.231:2957 -> a.b.f.164:53 SYN ******S*
 Apr 13 11:48:37 209.126.168.231:2959 -> a.b.f.166:53 SYN ******S*
 Apr 13 11:48:37 209.126.168.231:2974 -> a.b.f.181:53 SYN ******S*
 Apr 13 11:48:37 209.126.168.231:2976 -> a.b.f.183:53 SYN ******S*
 Apr 13 11:48:40 209.126.168.231:2985 -> a.b.f.192:53 SYN ******S*
 Apr 13 11:48:40 209.126.168.231:3041 -> a.b.f.246:53 SYN ******S*
 Apr 13 11:48:40 209.126.168.231:1713 -> a.b.d.237:53 SYN ******S*
 Apr 13 11:48:40 209.126.168.231:1947 -> a.b.e.214:53 SYN ******S*
 Apr 13 11:48:41 209.126.168.231:1971 -> a.b.e.238:53 SYN ******S*
 Apr 13 11:48:41 209.126.168.231:2340 -> a.b.f.18:53 SYN ******S*
```

```
Apr 13 11:48:28 hostka named[17373]: security: notice: denied query from
  [209.126.168.231].1107 for "version.bind"
Apr 13 11:47:52 hosth /kernel: Connection attempt to TCP a.b.c.62:53 from
  209.126.168.231:4452
Apr 13 11:48:28 hostka named[17373]: security: notice: denied query from
  [209.126.168.231].1107 for "version.bind"
Apr 13 11:48:54 hostmf /kernel: Connection attempt to TCP a.b.f.167:53 from
  209.126.168.231:2960
Apr 13 11:48:28 hostka snort: DNS named version attempt: 209.126.168.231:1107
  -> a.b.c.225:53
Apr 13 11:48:28 hostka snort: DNS named version attempt: 209.126.168.231:1107
  ➔  a.b.c.225:53
```

Source of trace:

GIAC http://www.sans.org/y2k/042401.htm

Detect was generated by

Snort Portscan pre-processor, snort IDS (alerting to syslog) running on “hostka”, possibly TCP wrappers running on “hostmf” and “hosth”.

Format of portscan and syslog entries is described above.

Probability the source address was spoofed

Probably not spoofed.

It is unlikely that the source address is spoofed as this appears to be a scan followed by an exploit attempt. In order for the scan to be effective, the attacker needs to receive packets back on the TCP connection.

Description of attack

If the time is synchronized between these systems, then it seems that a reasonable conclusion is that this is an automated exploit attempt. In other words, the attacker is running a program that will scan a range of addresses looking for DNS servers that will accept a TCP connection. Once a target has been identified, then a regular DNS query may be made (notice the UDP connection to port 53 of host a.b.c.225). This may be to confirm that the host is indeed a DNS server, at which point an attempt is made to find out what version of BIND is running on that server.

Attack mechanism

I can only hazard a guess as to the next step, but I assume that, if a BIND version 4 or 8 server was found, then this would be immediately followed by a buffer overflow attempt. (http://www.sans.org/newlook/resources/IDFAQ/TSIG.htm). Which, if successful, would compromise the server by giving the attacker the same privileges as the user running BIND. As this is normally 'root', this would be a 'root compromise'.

Correlations

This has been a very popular exploit recently, and there have been many, many attempts against BIND.

> http://www.incidents.org/archives/intrusions/msg00480.html
>
> http://www.isc.org/products/BIND/bind-security.html lists the known "bugs" in the various versions of BIND. Twelve of the "bugs" are listed, along with the relative severity of the compromise.
>
> http://www1.dshield.org/port_report.php?port=53 shows that this is currently a very popular target port, world-wide.

Evidence of active targeting

There appears to have been some preliminary reconnaissance, as not all possible addresses are listed in the scan. On the other hand, these hosts are not likely to ALL be DNS servers! So there is some evidence of active targeting, but not much. This is more likely a random scan of active hosts on the networks being monitored.

Severity

Criticality: 4
> This is a large network with a mix of systems

Lethality: 5
> Root is compromised if the attack is successful

Network Countermeasures: 3
> Average value – we do not know what measures are in place

System Countermeasures: 4
> The hosts seem to have some protective elements installed

Defensive recommendation

Replace any instance of BIND version 4 or 8 with the latest version 9 release.

Multiple choice test question

In the following trace, which description most accurately describes the event?

```
Apr 13 11:48:41 209.126.168.231:2340 -> a.b.f.18:53 SYN ******S*
```

a) a port scan of port 53 on host a.b.f.18
b) a port scan of port 2340 host 209.126.231
c) a DNS query
d) an intial TCP connection to port 53 on host a.b.f.18

The answer is d)

Network Detect 3.

```
Jul 10 02:38:40 211.250.158.2:3710 -> x.y.z.2:111 SYN ******S*
Jul 10 02:38:40 211.250.158.2:3713 -> x.y.z.5:111 SYN ******S*
Jul 10 02:38:40 211.250.158.2:3718 -> x.y.z.10:111 SYN ******S*
Jul 10 02:38:40 211.250.158.2:3714 -> x.y.z.6:111 SYN ******S*
Jul 10 02:38:43 211.250.158.2:3715 -> x.y.z.7:111 SYN ******S*
Jul 10 02:38:40 211.250.158.2:3722 -> x.y.z.14:111 SYN ******S*
Jul 10 02:38:44 211.250.158.2:4811 -> x.y.z.129:111 SYN ******S*
Jul 10 02:38:44 211.250.158.2:4814 -> x.y.z.130:111 SYN ******S*

All times EDT (UTC + 4)
```

Source of Trace

My company network

Detect was generated by

Snort's portscan pre-processor. Fields are <date> <time> <source IP address>:<source port> <direction> <dest IP address>:<dest port> <TCP flag(s) (ASCII)> <TCP flag bits>.

Probability the source address was spoofed

Probably not spoofed, BUT the source may have been anonomized by a Squid proxy server. APNIC reports this address as originating in Korea.

Connection back to the web server at this site shows it is "Dangson Cyber School" – the rest of the message appears to be in Korean, which I do not understand.

Port 3128 (Squid proxy) is open – determined by telnetting to 211.250.158.2 port 3128 – so this site may well be running a public proxy server.

Description of attack

This appears to be a targeted scan looking for RPC services that may be exploitable. No further activity was detected from this system – probably because there are no RPC services running on the targeted systems.

Attack mechanism

This appears to be reconnaissance looking for available RPC services. Once an exploitable service is found, I would assume that an exploit attempt would be immediately attempted.

Correlation

Checking the database at Dshield.org:

```
IP Address:  211.250.158.2
HostName:  211.250.158.2
DShield Profile:
Country:  KR
Contact E-mail: dangsan2@kornet.net
Total Records against IP:  25
Number of targets:  14
Date Range: 2001-06-28 to 2001-07-08

Ports Attacked (up to 10):
Port   Attacks
111       1
515       2

Whois:
```

```
IP Address          : 211.250.158.0-211.250.158.63
Connect ISP Name    : PUBNET
Connect Date        : 20001214
Registration Date   : 20001129
Network Name        : DANGSAN-MSCH

[ Organization Information ]
Orgnization ID      : ORG153843
Name                : Dangsan middle school
State               : SEOUL
Address             : 121 6KA DANGSANDONG YOUNGDEUNGPOKU
Zip Code            : 150-723

[ Admin Contact Information]
Name                : Kwangsoo Hong
Org Name            : Dangsan middle school
State               : SEOUL
Address             : 121 6KA DANGSANDONG YOUNGDEUNGPOKU
Zip Code            : 150-723
Phone               : +82-2-2631-1637
Fax                 : +82-2-2632-6420
E-Mail              : dangsan2@kornet.net

[ Technical Contact Information ]
Name                : Kwangsoo Hong
Org Name            : Dangsan middle school
State               : SEOUL
Address             : 121 6KA DANGSANDONG YOUNGDEUNGPOKU
Zip Code            : 150-723
Phone               : +82-2-2631-1637
Fax                 : +82-2-2632-6420
E-Mail              : dangsan2@kornet.net
```


Evidence of active targeting

This appears to be targeted at active hosts on the network, so this was likely preceded by a network scan that uncovered the host IP addresses.

Severity

Criticality: 5

This network primarily contains firewalls, DNS servers, and corporate web and mail servers

Lethality: 2

This was a reconnaissance scan that did not uncover any vulnerabilities

Network countermeasures: 4

This is a network is monitored by Snort IDS, routers managed by the ISP, and patched up-to-date

System countermeasures: 5

This is a well-protected network with proxy firewalls and up-to-date patches on all systems

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

Defensive recommendation

No further action is necessary. This site has already been identified as an unfriendly source….

Multiple choice test question

What created the following log entries (lines may be wrapped for readability):

```
Jul 10 02:38:40 211.250.158.2:3710 -> x.y.z.2:111 SYN ******S*
Jul 10 02:38:40 211.250.158.2:3713 -> x.y.z.5:111 SYN ******S*
Jul 10 02:38:40 211.250.158.2:3718 -> x.y.z.10:111 SYN ******S*
```

a) PortSentry IDS
b) Shadow
c) Snort portscan pre-processor
d) Snort portscan post-processor

Answer is c).

Network Detect 4.

```
[**] Possible Queso Fingerprint attempt [**]
07/11-11:28:01.980433 0:E0:1E:B9:7E:A0 -> 0:10:83:18:99:C7 type:0x800
len:0x4A
205.150.254.48:3185 -> 207.61.166.5:25 TCP TTL:55 TOS:0x0 ID:0 IpLen:20
DgmLen:60 DF
12****S* Seq: 0x50D7B743  Ack: 0x0  Win: 0x16B0  TcpLen: 40
TCP Options (5) => MSS: 1412 SackOK TS: 110495690 0 NOP WS: 0
```

Source of Trace

My company network

Detect was generated by

Snort IDS sensor.
Field descriptions:

[**] <message text from snort rule that was triggered> [**]
<date>-<local time> <source MAC address> -> <destination MAC address>
type:<network type> len:<ethernet datagram length>
<source IP address>:<source port> -> <destination IP address>:<destination port>
<protocol> TTL:<time-to-live (hop count)> TOS: <type of service byte> ID:<IP id #>
IpLen:<length of IP header (bytes)> DgmLen:<size of IP packet (bytes)> <frag flag>
<TCP flags> Seq: <IP sequence #> Ack: <IP acknowledgement #> Win: <max window size available> TcpLen: <length of TCP packet>
TCP Options <list of tcp options included in header>

Probability the source address was spoofed

Probably not, but could be. From http://www.whitehats.com/IDS/29:

> Although this event was caused by a TCP packet, the packet is not thought to be a part of an existing TCP session. Therefor (sic) the source IP address could be easily forged. It has been noted that the intruder is likely to expect or desire a response to their packets, so it may be likely that the source IP address is not spoofed.

Description of attack

If this was indeed a Queso fingerprint attempt, then this would be an attempt to find out what operating system the target is running, and what version. This is determined by seeing how different system react to "invalid" tcp packets.

Note, however, in this case that this is probably NOT a fingerprint, but a "false positive". Apparently Queso sets the initial TTL to 255, so, unless we are 200 hops away from the source (which would put it somewhere on Mars!), this is not Queso.

A more likely explanation is that this is an ECN-enabled host starting a normal connection to the SMTP port on the firewall – which is also the email gateway.

Checking the IP address, DNS yields:

```
Name:    lance.adgdesign.net
Address:  205.150.254.48
lance.adgdesign.net     internet address = 205.150.254.48
adgdesign.net   nameserver = ns1.travel-net.com
adgdesign.net   nameserver = ns2.travel-net.com
ns1.travel-net.com      internet address = 204.92.71.5
ns2.travel-net.com      internet address = 204.92.71.6
```

which looks legitimate.

Attack mechanism

Queso fingerprinting is classified as "Information Gathering" and is an attempt to determine information about the targeted system. It may be a prelude to an attack, or it may be just curiosity. I know that Netcraft (http://www.netcraft.com) can identify operating systems, though they do not divulge how this is done.

Correlation

If this was Queso (which it is not), relevant references would be whitehats IDS29, CVE CAN-1999-0454, and advICE 2000313

Evidence of active targeting

This was probably part of a normal SMTP session, so, yes, there was "active targeting"

Severity

Criticality: 5

This network primarily contains firewalls, DNS servers, and corporate web and mail servers

Lethality: 0

This was probably a false positive

Network countermeasures: 4

This is a network is monitored by Snort IDS, routers a managed by the ISP and patched up-to-date

System countermeasures: 5

This is a well-protected network with proxy firewalls and up-to-date patches on all systems

(5 + 0) - (4 + 5) = -4

Defensive recommendation

None. All reasonable defences are already in place.

Multiple choice test question

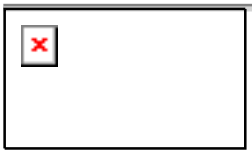
What probably caused the following snort rule to be triggered:

```
[**] Possible Queso Fingerprint attempt [**]
07/11-11:28:01.980433 0:E0:1E:B9:7E:A0 -> 0:10:83:18:99:C7 type:0x800
len:0x4A
205.150.254.48:3185 -> 207.61.166.5:25 TCP TTL:55 TOS:0x0 ID:0 IpLen:20
DgmLen:60 DF
12****S* Seq: 0x50D7B743  Ack: 0x0  Win: 0x16B0  TcpLen: 40
TCP Options (5) => MSS: 1412 SackOK TS: 110495690 0 NOP WS: 0
```

a) Strange or Invalid TCP options
b) Strange or Invalid TCP flags
c) Invalid IP id number
d) None of the above

Answer is b).

Network Detect 5.



http://www.silicondefense.com/
http://www.silicondefense.com/

SnortSnarf alert page

Source: *209.126.172.254*

SnortSnarf v052301.1

9 such alerts found using input module SnortFileInput, with sources:

- /home/snort/LOGS/ShoppersDrugMart/20010722/20010722.07/snort.alert
- /home/snort/LOGS/ShoppersDrugMart/20010722/20010722.07/snort_portscan.log

Earliest: **07:09:32** *on 7/22/2001*
Latest: **07:09:35** *on 7/22/2001*
2 different signatures are present for *209.126.172.254* as a source

- 1 instances of *OVERFLOW-NOOP-X86*
- 8 instances of *TCP ******S* scan*

There are 8 distinct destination IPs in the alerts of the type on this page.

209.126.172.254	Whois lookup at:	ARIN	RIPE	APNIC	Geektools
	DNS lookup at:	Amenesi	TRIUMF	Princeton	
See also 209.126.172.254 as an alert destination [1 alerts]					

Jul 22 07:09:32 209.126.172.254:2045 -> 207.61.166.2:80 SYN ******S*
Jul 22 07:09:32 209.126.172.254:2050 -> 207.61.166.5:80 SYN ******S*
Jul 22 07:09:32 209.126.172.254:2055 -> 207.61.166.10:80 SYN ******S*

```
Jul 22 07:09:32 209.126.172.254:2063 -> 207.61.166.14:80 SYN ******S*
Jul 22 07:09:33 209.126.172.254:2201 -> 207.61.166.6:80 SYN ******S*
Jul 22 07:09:33 209.126.172.254:2209 -> 207.61.166.129:80 SYN ******S*
Jul 22 07:09:33 209.126.172.254:2210 -> 207.61.166.130:80 SYN ******S*
[**] OVERFLOW-NOOP-X86 [**]
07/22-07:09:33.140433 0:E0:1E:B9:7E:A0 -> 0:60:8:9F:19:43 type:0x800
len:0x4E0
209.126.172.254:2201 -> 207.61.166.6:80 TCP TTL:49 TOS:0x0 ID:2211 IpLen:20
DgmLen:1234 DF
***AP*** Seq: 0xC52796CF Ack: 0xFB67DCC3 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1601166138 9323623 [Snort log]
Jul 22 07:09:35 209.126.172.254:2052 -> 207.61.166.7:80 SYN ******S*
```

SnortSnarf brought to you courtesy of Silicon Defense
Authors: Jim Hoagland and Stuart Staniford
See also the Snort Page by Marty Roesch
Page generated at Sun Jul 22 08:05:05 2001

```
[**] OVERFLOW-NOOP-X86 [**]
07/22-07:09:33.140433 0:E0:1E:B9:7E:A0 -> 0:60:8:9F:19:43 type:0x800
len:0x4E0
209.126.172.254:2201 -> 207.61.166.6:80 TCP TTL:49 TOS:0x0 ID:2211
IpLen:20 DgmLen:1234 DF
***AP*** Seq: 0xC52796CF  Ack: 0xFB67DCC3  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1601166138 9323623
47 45 54 20 2F 4E 55 4C 4C 2E 70 72 69 6E 74 65  GET /NULL.printe
72 20 48 54 54 50 2F 31 2E 30 0D 0A 42 65 61 76  r HTTP/1.0..Beav
75 68 3A 20 90 90 90 90 90 90 90 90 90 90 90 90  uh: ............
90 90 90 90 90 90 90 90 EB 03 5D EB 05 E8 F8 FF  ..........].....
FF FF 83 C5 15 90 90 90 8B C5 33 C9 66 B9 D7 02  ..........3.f...
50 80 30 95 40 E2 FA 2D 95 95 64 E2 14 AD D8 CF  P.0.@..-..d.....
05 95 E1 96 DD 7E 60 7D 95 95 95 95 C8 1E 40 14  .....~`}......@.
7F 9A 6B 6A 6A 1E 4D 1E E6 A9 96 66 1E E3 ED 96  ..kjj.M....f....
66 1E EB B5 96 6E 1E DB 81 A6 78 C3 C2 C4 1E AA  f....n....x.....
96 6E 1E 67 2C 9B 95 95 95 66 33 E1 9D CC CA 16  .n.g,....f3.....
52 91 D0 77 72 CC CA CB 1E 58 1E D3 B1 96 56 44  R..wr....X....VD
74 96 54 A6 5C F3 1E 9D 1E D3 89 96 56 54 74 97  t.T.\........VTt.
96 54 1E 95 96 56 1E 67 1E 6B 1E 45 2C 9E 95 95  .T...V.g.k.E,...
95 7D E1 94 95 95 A6 55 39 10 55 E0 6C C7 C3 6A  .}.....U9.U.l..j
C2 41 CF 1E 4D 2C 93 95 95 95 7D CE 94 95 95 52  .A..M,....}....R
D2 F1 99 95 95 95 52 D2 FD 95 95 95 95 52 D2 F9  ......R......R..
94 95 95 95 FF 95 18 D2 F1 C5 18 D2 85 C5 18 D2  ................
81 C5 6A C2 55 FF 95 18 D2 F1 C5 18 D2 8D C5 18  ..j.U...........
D2 89 C5 6A C2 55 52 D2 B5 D1 95 95 95 18 D2 B5  ...j.UR.........
C5 6A C2 51 1E D2 85 1C D2 C9 1C D2 F5 1E D2 89  .j.Q............
1C D2 CD 14 DA D9 94 94 95 95 F3 52 D2 C5 95 95  ...........R....
18 D2 E5 C5 18 D2 B5 C5 A6 55 C5 C5 C5 FF 94 C5  .........U......
C5 7D 95 95 95 95 C8 14 78 D5 6B 6A 6A C0 C5 6A  .}......x.kjj..j
C2 5D 6A E2 85 6A C2 71 6A E2 89 6A C2 71 FD 95  .]j..j.qj..j.q..
91 95 95 FF D5 6A C2 45 1E 7D C5 FD 94 94 95 95  .....j.E.}......
6A C2 7D 10 55 9A 10 3F 95 95 95 A6 55 C5 D5 C5  j.}.U..?....U...
D5 C5 6A C2 79 16 6D 6A 9A 11 02 95 95 95 1E 4D  ..j.y.mj.......M
F3 52 92 97 95 F3 52 D2 97 A7 F1 52 D2 91 44 EB  .R....R....R..D.
39 6B FF 85 18 92 C5 C6 6A C2 61 FF A7 6A C2 49  9k......j.a..j.I
A6 5C C4 C3 C4 C4 C4 6A E2 81 6A C2 59 10 55 E1  .\.....j..j.Y.U.
F5 05 05 05 05 15 AB 95 E1 BA 05 05 05 05 FF 95  ................
C3 FD 95 91 95 95 C0 6A E2 81 6A C2 4D 10 55 E1  .......j..j.M.U.
D5 05 05 05 05 FF 95 6A A3 C0 C6 6A C2 6D 16 6D  .......j...j.m.m
6A E1 BB 05 05 05 05 7E 27 FF 95 FD 95 91 95 95  j......~'.......
```

```
C0 C6 6A C2 69 10 55 E9 8D 05 05 05 05 E1 09 FF  ..j.i.U.........
95 C3 C5 C0 6A E2 8D 6A C2 41 FF A7 6A C2 49 7E  ....j..j.A..j.I~
1F C6 6A C2 65 FF 95 6A C2 75 A6 55 39 10 55 E0  ..j.e..j.u.U9.U.
6C C4 C7 C3 C6 6A 47 CF CC 3E 77 7B 56 D2 F0 E1  l....jG..>w{V...
C5 E7 FA F6 D4 F1 F1 E7 F0 E6 E6 95 D9 FA F4 F1  ................
D9 FC F7 E7 F4 E7 EC D4 95 D6 E7 F0 F4 E1 F0 C5  ................
FC E5 F0 95 D2 F0 E1 C6 E1 F4 E7 E1 E0 E5 DC FB  ................
F3 FA D4 95 D6 E7 F0 F4 E1 F0 C5 E7 FA F6 F0 E6  ................
E6 D4 95 C5 F0 F0 FE DB F4 F8 F0 F1 C5 FC E5 F0  ................
95 D2 F9 FA F7 F4 F9 D4 F9 F9 FA F6 95 C2 E7 FC  ................
E1 F0 D3 FC F9 F0 95 C7 F0 F4 F1 D3 FC F9 F0 95  ................
C6 F9 F0 F0 E5 95 D0 ED FC E1 C5 E7 FA F6 F0 E6  ................
E6 95 D6 F9 FA E6 F0 DD F4 FB F1 F9 F0 95 C2 C6  ................
DA D6 DE A6 A7 95 C2 C6 D4 C6 E1 F4 E7 E1 E0 E5  ................
95 E6 FA F6 FE F0 E1 95 F6 F9 FA E6 F0 E6 FA F6  ................
FE F0 E1 95 F6 FA FB FB F0 F6 E1 95 E6 F0 FB F1  ................
95 E7 F0 F6 E3 95 F6 F8 F1 BB F0 ED F0 95 0D 0A  ................
48 6F 73 74 3A 20 90 90 90 90 90 90 90 90 90 90  Host: ..........
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 33  ...............3
C0 B0 90 03 D8 8B 03 8B 40 60 33 DB B3 24 03 C3  ........@`3..$..
FF E0 EB B9 90 90 05 31 8C 6A 0D 0A 0D 0A        .......1.j....

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

Source of Trace

My network

Detect was generated by

Snort IDS running verion 1.7 ruleset, as analyzed by SnortSnarf and managed by snorticus.
The rule that was triggered was:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"OVERFLOW-NOOP-X86";flags:PA;
content:"|9090 9090 9090 9090 9090 9090 9090 9090|";)
```

found in the "overflow-lib" file.

The Snort log entry consists of the ruleset message (i.e. msg:" [**] OVERFLOW-NOOP-X86 [**]";) found in the ruleset,
followed by 4-lines from the IP header, and finally the packet contents in hex (on the left) and ascii (on the right). Characters that cannot be represented in ascii are displayed as a '.'
The 4-line IP header is as follows:
<date>-<time> <source MAC address> -> <destination MAC address> type:<ethernet type> len:<datagram length>
<source IP address>:<source port> -> <destination IP address>:<destination port> <IP protocol> TTL:<time to live> TOS:<type of service byte> ID:<IP id#> IpLen:<length of IP header> DgmLen:<length of ip datagram> <Frag flag>
<TCP flags> Seq: <TCP seq #> Ack: <TCP Ack #> Win: <max window size supported> TcpLen: <TCP header length (bytes)>

Probability the source address was spoofed

Not very likely. This is part of an established TCP connection, that occurred in the middle of a scan. So the attacker was probably getting data back on the connection.

Description of attack

A SYN scan is carried out against systems listening on port 80 (HTTP). There is nothing in the portscan log to how it is done, but this most likely was a fingerprinting exercise. Once a host that appears to be running on Intel processors is identified, an immediate buffer overflow is attempted against that port.

Attack mechanism

Once a target host has been identified, a very large HTTP request is fabricated apparently in an attempt to overflow the buffer on the target system, and cause code fragments in the request to be executed.

Correlation

From ARIN:
California Regional Internet, Inc. (NETBLK-CARI)
8929A COMPLEX DRIVE
SAN DIEGO, CA 92123
US

Netname: CARI

Netblock: 209.126.128.0 - 209.126.175.255
Maintainer: CALI

Coordinator:
California Regional Intranet, Inc. (IC63-ARIN) sysadmin@cari.net
858-974-5080

Domain System inverse mapping provided by:

NS1.CARI.NET 216.98.128.70
NS2.CARI.NET 216.98.138.70

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

Record last updated on 22-Aug-2000.
Database last updated on 21-Jul-2001 23:13:10 EDT.

and from DNS
Name: 4db172254.aspadmin.net
Address: 209.126.172.254

Checking Dshield.org shows that this host has been implicated in attacks against port 80 on 2 other systems recently.

I do not have access to this system, so cannot check the syslog.

Evidence of active targeting

Some. Only hosts that are listening on TCP port 80 appear to have been targeted by the scan, so presumably some reconnaissance had been done earlier.

Severity

Criticality: 5
This network primarily contains firewalls, DNS servers, and corporate web and mail servers
Lethality: 5
This system is a firewall
Network countermeasures: 5
This is a network is monitored by Snort IDS, routers a managed by the ISP and patched up-to-date
System countermeasures: 5
This is a well-protected network with proxy firewalls and up-to-date patches on all systems

(5 + 5) - (5 + 5) = 0

Defensive recommendation

Keep looking! All reasonable defences are already in place, but that is not enough in itself.

Multiple choice test question

What probably caused the following snort rule to be triggered:

```
[**] OVERFLOW-NOOP-X86 [**]
07/22-07:09:33.140433 0:E0:1E:B9:7E:A0 -> 0:60:8:9F:19:43 type:0x800
len:0x4E0
209.126.172.254:2201 -> 207.61.166.6:80 TCP TTL:49 TOS:0x0 ID:2211
IpLen:20 DgmLen:1234 DF
***AP*** Seq: 0xC52796CF  Ack: 0xFB67DCC3  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1601166138 9323623
47 45 54 20 2F 4E 55 4C 4C 2E 70 72 69 6E 74 65  GET /NULL.printe
72 20 48 54 54 50 2F 31 2E 30 0D 0A 42 65 61 76  r HTTP/1.0..Beav
75 68 3A 20 90 90 90 90 90 90 90 90 90 90 90 90  uh: ............
90 90 90 90 90 90 90 90 EB 03 5D EB 05 E8 F8 FF  ..........].....
```

a) GET /NULL.printer in the packet payload
b) NOP NOP in the TCP header
c) 90 90 90 90 90 90 in the packet payload
d) Beavuh: in the packet payload

Answer is c).

Describe the State of Intrusion Detection

Introduction

I will describe the current state of our company network configuration as is pertains to our connection to the internet, with a particular emphasis placed on the IDS component. The location and composition of IDS sensors and analyzers are described, along with a description of some of the other measures that are currently in place. There is a description of the centralized syslog server and how it is used to monitor log data not only from the IDS sensors but also from the firewalls and all other hosts inside the perimeter of our network.

IDS sensor

We currently have one sensor located on the external class C network – we "own" the class C address space.
The sensor is running on a RedHat linux system with two ethernet cards installed. One card is attached to an internal network (i.e. inside the "firewall") and the other is attached to the class C network in such a way that it can "see" all the traffic that traverses that network.
Only the "internal" network card has an IP address. The "external" card is run in promiscuous mode, i.e. it listens to all the traffic on the network, but cannot send data. Thus, to all intents and purposes, it is invisible.

The software running on this system is version 1.7 of Snort. Snort requires 'libpcap' to be installed.

From the README's:

```
Snort Version 1.7


by Martin Roesch (roesch@clark.net)

Distribution Site:
http://www.snort.org
http://snort.sourceforge.net

Alternate Sites:

US:
http://www.technotronic.com
http://packetstorm.securify.com
http://snort.whitehats.com

Europe:
http://gd.tuwien.ac.at/infosys/security/snort
ftp://gd.tuwien.ac.at/infosys/security/snort
http://www.centus.com/snort/security.html

South America:
http://snort.safenetworks.com

Austrailia: (sic)
ftp://the.wiretapped.net/pub/security/network-intrusion-detection/snort


********************************************************************************
COPYRIGHT

Copyright (C)1998,1999,2000,2001 Martin Roesch

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
```

```
LIBPCAP 0.4
Lawrence Berkeley National Laboratory
Network Research Group
libpcap@ee.lbl.gov
ftp://ftp.ee.lbl.gov/libpcap.tar.Z
```

The only modification made to the snort.conf file are:

```
var HOME_NET MY.NET.x.0/24
var DNS_SERVERS [MY.NET.dns.server/32,MY.service.provider.dns/32]
output alert_syslog: LOG_AUTH LOG_ALERT
```

In order for this sensor to send syslog messages to a central server, the file /etc/syslog.conf was modified to include the following line, and the syslog daemon was re-started:
***.info;mail.none;authpriv.none;cron.none @10.x.y.10**
where "10.x.y.10" is the IP address of the centralized syslog server.

Finally, snort is run using this syntax:
snort –i eth0 –b –c snort.conf where "eth0" is the network interface connected the DMZ.

Firewalls

The outer perimeter of our network is protected by proxy-based firewalls, with a deny-all default policy. These firewalls are also configured to send syslog events to the central syslog server.
The primary firewall also has a "split" DNS. i.e. it answers queries from both internal and external sources, but will only recurse internal queries. It is the primary DNS server for our address space. It also acts as our email gateway, running SMAP on port 25, and passing approved messages to the backend SENDMAIL process for delivery.
No inbound TELNET, FTP, or SSH services are allowed – in fact, the only inbound service supported is SMTP (and DNS, of course).
The syslog daemon is configured as above, with the following entries in the file /etc/syslog.conf:

```
# Corp TS
*.notice                @ov
mail.debug              @ov
```

As the firewall is running DNS, we can use a domain name to specify the central syslog server. It is known as "ov" as one of the tools running on that system is "HP Openview", but more about that later.

Web servers

The company web servers are located behind a separate firewall, with a similar deny-all policy. The advertised internet addresses are NAT-ed (Network Addresss Translation) by this firewall to an isolated network. The web server itself does not run any unnecessary network services – in fact 'inetd' is not even running! The only network services are HTTP, SSL, and SSH. Admin access is via SSH from our internal network via another proxy-based firewall. Access to the HTTP admin ports is only allowed from 2 ip addresses inside our network, via the internal firewall.

Syslog server

This is a system running HP-UX 11.00 inside our network. One of this system's primary purposes is to monitor all systems in our network. In order to facilitate this, is has HP's IT Operation Centre installed (HP renamed this to Vantagepoint, and it NOW called "HP Openview operations"!). For the sake of consistency, I will refer to it as ITO in this document.

The architecture of ITO is to have clients running on all the monitored systems reporting back to the management server (this host). Each of these clients can be configured in many different ways. One of these ways is to monitor arbitrary log files. We have one of these monitoring agents installed on the Syslog server monitoring the syslog output file, which HP defaults to */var/adm/syslog/syslog.log*. Once an event is identified, an alert is sent to the Operator's console, which is monitored 24-hours a day.

The drawback to this is that only messages that match a pre-defined pattern generate an alert. So anything new or unusual will be silently ignored – at least that is true the way we have it configured at the present time. It would be possible (and fairly simple) to add all the Snort alert messages to the filter such that an ITO alert would be issued for every Snort alert. However, the operations staff are not ID analysts, and so could not take any meaningful action.

If we cannot use ITO, then, we need some other mechanism to filter the syslog files and only key on "interesting" entries. To this end, there is a program called "logcheck". The following is excerpted from 'logcheck.sh':

```
#   logcheck.sh: Log file checker
#   Written by Craig Rowland <crowland@psionic.com>
#
#   This file needs the program logtail.c to run
#
#   This script checks logs for unusual activity and blatant
#   attempts at hacking. All items are mailed to administrators
#   for review. This script and the logtail.c program are based upon
#       the frequentcheck.sh script idea from the Gauntlet(tm) Firewall
#   (c)Trusted Information Systems Inc. The original authors are
#   Marcus J. Ranum and Fred Avolio.
#
#   Default search files are tuned towards the TIS Firewall toolkit
#   the TCP Wrapper program. Custom daemons and reporting facilites
#   can be accounted for as well...read the rest of the script for
#   details.
```

Which is very interesting, as the firewalls we use are Gauntlet TIS (http://www.pgp.com/products/gauntlet/default.asp)

Once installed, the only modification that needed to be made was to enter the email address of the analyst(s) responsible, and to add an entry to the crontab file. I have it setup to be run once an hour during a normal working day and once just before midnight every day, as we roll the syslog files over daily at midnight.

Logcheck program

From the README file:
"Logcheck is based upon a log checking program called frequentcheck.sh featured in the Gauntlet(tm) firewall package by Trusted Information Systems Inc. (http://www.tis.com). The logcheck shell script and logtail.c program have been completely re-written from scratch and is implemented in a slightly different manner to accommodate for two methods of log file auditing:

1) By reporting everything you tell it to specifically look for via keywords.

2) By reporting everything you didn't tell it to ignore via keywords.

This ensures that important messages are specifically brought to your attention (via the keywords you look for) and that important messages that you may have overlooked are also reported (by only ignoring items you tell it to). The original frequentcheck.sh script was implemented in a somewhat similar manner."

It quickly becomes obvious that the filter rules need to be modified for your particular installation. There are so many messages being generated that we do not need to know about – at least not for this purpose.

Where to from here?

The configuration above works well, and, with a little bit of tuning, does not produce too much information. It has the enormous advantage that it not only reports on the Snort IDS entries but also reports on the firewall log data as well. This correlation between events can sometimes help pinpoint particular items of interest.

However, as with all things, there are drawbacks. One of these is that you see all the alerts individually, so it is not easy to see a pattern developing. Also, there is no easy way to look at the actual packets that Snort has captured. These are still located on the Sensor IDS.

So, another approach has been tried…

Snortsnarf

These two tools (SnortSnarf and logcheck) together provide some powerful analysis and management tools. Snortsnarf is a collection of utilities built around a perl script. From the README (wonderful things, readme's….):

```
README file for SnortSnarf v052301.1
------------------------------------

Welcome to the release of SnortSnarf v052301.1
(http://www.silicondefense.com/software/snortsnarf/).  This program creates a
set of HTML pages to allow you to quickly and conveniently navigate around
output files of the Snort intrusion detection system (http://www.snort.org/).
```

which just about sums the whole thing up in one sentence!

SnortSnarf works best on the default output files that Snort produces. Unfortunately, this seems to mean turning off the syslog output from Snort. This is accomplished by commenting out the line:
#output alert_syslog: LOG_AUTH LOG_ALERT
in the 'snort.conf' file, and NOT using the '-b' option when running Snort.

I installed SnortSnarf on yet another HP-UX system that also happens to run an intranet web server. This allows me to copy the Snort files from the Sensor system and analyze the results with SnortSnarf. Very powerful, and very useful, but the process is **still** manual…., so enter yet another wonderful tool, "snorticus".

Snorticus

"snorticus" is a set of scripts and a methodology for managing a Snort / SnortSnarf environment. It provides the essential glue to tie the whole thing together. To quote from http://www.snort.org, "Snorticus is a collection of shell scripts designed to allow easy management of Snort sensors. It allows you to routinely collect Snort sensor data, analyze the data in SnortSnarf, and easily maintain rule files." It was written by Paul Ritchey

To make this work, I created a new account "snort" on the web server where SnortSnarf was also installed. On the sensor, a similar account was created with an identical home directory, and 'snorticus' was copied to both systems.

The 'hourly_wrapup.sh' is run on the Sensor site from root's cron once an hour using the following:

0 * * * * /home/snort/snorticus-1.0.3/hourly_wrapup.sh
which causes the command "/home/snort/snorticus-1.0.3/hourly_wrapup.sh" to be run every hour on the hour, 24-hours a day, 7-days a week (assuming 'cron' is running!).

From the script:

```
# Algorithm used:
#       1.  pkill the snort process(es)
#       2.  restart the snort process(es)
#               a.  Retrieve netblocks to watch.
#               b.  Create new directories
#                       1.  Create new hour directory.
#                       2.  Create new netblock subdirectories.:wq
#               c.  Start snort instance for each netblock.
#                       1.  Determine CIDR info for current netblock
#                       2.  Start instance of snort with appropriate params.
#       3.  Tar and gzip the last hour's data so it can be pulled
#           down by the analyst box.
#       4.  Now delete data over a certain age (user specified)
```

'pkill' was removed in version 1.0.1, but has been replaced with a sequence that effectively kills the running snort process.

On our intranet web server site, aka "analyst box", the "retrieve_wrapup.sh" is run by the user "snort". This script uses SSH to copy the recently created tar file from the sensor system, and run the SnortSnarf program against the extracted datafiles.

From the script, again:

```
#!/usr/bin/csh -f
# This script is designed to be run through a cron job
# once an hour, every hour.

# It's purpose is to retrieve the hourly wrapup from
# the snort sensor to the analysis box.  Called
# without any parameter it will retrieve the
# previous hour's wrapup from the sensor.  Called with
# a parameter it will retrieve the requested hour's
# wrapup data from the sensor.

# After retrieving the hourly wrapup, it will process
# the data (after untarring and ungzipping it) through
# SnortSnarf to create the html pages for the analyst(s).
```

And the corresponding cron entry:
5 * * * * /home/snort/snorticus-1.0.3/retrieve_wrapup.sh XXXXXX 10.x.y.100 > /dev/null 2>&1
Note that 10.x.y.100 is the IP address of the Sensor's internal network card, and that XXXXX is the "name" of the network being monitored.
You will notice that this allows us to monitor several networks using multiple sensors without getting the data confused.

Conclusions

It is possible to build a large scale sensor array using a combination of snorticus, SnortSnart, and Snort, and have one simple browser interface displaying the resultant information. However, this ignores the data generated by the firewalls and the other hosts in the network. An ideal solution would incorporate all the features of both the snorticus and the logcheck approach. Perhaps this has already been done by someone, and I have yet find it!

"Analyze This" Scenario

Introduction

We have been asked to provide a security audit for the University of XXX. We have been provided with incomplete data from a Snort system (http://www.snort.org), equipped with a standard ruleset. We will analyze this dataset with a view to identifying some common events and evidence of possibly compromised systems. A list of detects will be given with a brief description of each, a list of often occurring remote addresses will be identified, and a link graph and analysis of out-of-spec files will be provided.

Analysis of Log Files

A total of 68 files were provided as source material for this analysis, ranging it date from 20th January, 2001 to 12th March, 2001. There were three types of files provided, Snort (http://www.snort.org) Alert files, portscan files generated by the Snort portscan preprocessor, and Snort packets captured by an OOS filter (68 bytes captured).

The following table shows the files supplied and the dates for which they contained data. In each case the file contained data for a 24-hour period, with the exception of the files for March 9th, which terminated at 17:15. Of the total dataset supplied, there are only 6 days when all 3 file types were available.

Date	OOS Filename	Portscan Filename	Alert Filename
20th Jan	OOSche32		
21st Jan		SnortS8	
23rd Jan	OOSche29		
30th Jan			SnortA6
31st Jan	OOSche5	SnortS7	
1st Feb	OOSche34	SnortS26	
2nd Feb	OOSche4		

3rd Feb			SnortA3
4th Feb	OOScheck	SnortS2	SnortAle
5th Feb		SnortSca	
6th Feb	OOSche33	SnortS34	SnortA36 *
		SnortS32	
8th Feb	OOSche31		
9th Feb	OOSche30	SnortS29	
10th Feb	OOSche28		
11th Feb	OOSche26	SnortS27	SnortA25
12th Feb	OOSche24		
20th Feb		UMBCNI4	UMBCNI3 *
21st Feb		UMBCNI2	
22nd Feb		UMBCNI61	UMBCNI60
23rd Feb		UMBCNI25 *	UMBCNI27
24th Feb	UMBCNI37	UMBCNI35	UMBCNI30
25th Feb		UMBCNI59	UMBCNI58
26th Feb		UMBCNI57	
27th Feb	UMBCNI56	UMBCNI55	UMBCNI54
28th Feb		UMBCNI53	UMBCNI52
1st Mar	UMBCNI50	UMBCNI51	
2nd Mar	UMBCNI48	UMBCNI46	
3rd Mar	UMBCNI49	UMBCNI47	
4th Mar	UMBCNI45	UMBCNI44	
5th Mar	UMBCNI38	UMBCNI36	
6th Mar	UMBCNI42	UMBCNI43	UMBCNI44
7th Mar		UMBCNI40	UMBCNI39
8th Mar	UMBCNI33		
9th Mar		UMBCNI34	UMBCNI32
10th Mar		UMBCNI29	UMBCNI28
12th Mar		UMBCNI26	

*Files SnortA35 and SnortA36 are identical, UMBCNI3 and UMBCNI5 are identical, and UMBCNI25 and UMBCNI31 are identical.

Various unix command line tools, such as perl, awk, sed, and grep are available to analyze these files. For an initial analysis, I wrote a simple sed script to replace all instances of MY.NET with 10.10 (after having checked that this network address was not used in any file supplied). The 10.10.0.0 network was selected because it is defined as a "private" network by RFC1918 (ftp://ftp.isi.edu/in-notes/rfc1918.txt), and as such is not routed across the internet. Thus, we should never see addresses in this range unless it is our own private network. As this is not a private network, this address can be safely used in place of MY.NET.

Personally, I use 'vi' a lot, so I ran the actual conversion from inside 'vi', but there are many more elegant ways to do this. This is very hard to describe graphically, so I will have to describe the steps in words:

First, run 'ls –1 *.txt > /tmp/filename', then 'vi /tmp/filename'
With the cursor on the first line, type
!Gawk '{print "sed s/MY.NET/10.10/g "$1,$1}'
This replaces each line in the file with "sed s/MY.NET/10.10/ filename.txt filename.txt"
The next command
:%s/txt$/in/
replaces the second .txt with .in. Finally, save the file, and type
:w
:!sh %
which executes the file, and creates a new set of files with the .in extension in the same directory as the .txt files.

I have seen more elegant ways of doing this, but this works…….

Once I had made the conversion, I ran SnortSnarf (http://www.silicondefense.com/snortsnarf/) on the individual files using the following script:

```
while read f
do
  /home/root/SnortSnarf-052301.1/snortsnarf.pl -homenet 10.10.0.0/16 \
    -d /opt/netscape/docs/it/snort/$f ~xyz/snortfiles/$f.in
done < /tmp/xyz
```

where /tmp/xyz is a list of the files to be processed, created by executing 'ls –1 *.in > /tmp/xyz'.

As there were 68 files to process, this naturally took some time to execute, and a lot of disk space to store the output. Also, the OOS (Out Of Spec) files were not in a format that SnortSnarf recognized, so produced no output. This was not a major concern, as a few grep command would elicit the same information – perhaps not in such an elegant, easy to read format, but still very useful.

While the files were being processed, I took a look at each file and found that the chronological order of the files did not match the ASCII collating sequence. In fact, I could not be certain the precise naming convention. However, a couple of things became obvious.

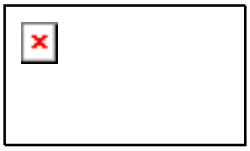
There was a significant gap between the Monday, 12th Jan and Tuesday, 20th Jan, with only one file being created on 12th.

There is significant change in the naming convention used for the output files. From the 20th onwards, all the files are named "UMBCNIxx", where xx is one or two digits, though I cannot determine what those two digits are supposed to represent. Also, the words "[UMBC NIDS}" are inserted in each file.

So, what does this mean. Well, perhaps the analyst was on vacation, or at a SANS conference the week on the 12th, and his/her sensor continued to operate until there was no more space available to store the output. Then it stopped running. When the analyst returned, he/she modified the scripts that create the files in the first place in such a manner that they can be incorporated into a network of NIDS.

The other event of interest occurred at about 17:15 on March 9th, when the log files stop abruptly. This could be caused by a sensor crash and reboot, or a power outage that caused the sensor to reboot. Unless there was a startup script for the snort process, there would be no more output until the midnight rollover process, which starts a new instance of snort running.

Detection of Events



http://www.silicondefense.com/
http://www.silicondefense.com/

SnortSnarf start page

All Snort signatures

SnortSnarf v052301.1

22522 alerts found using input module SnortFileInput, with sources:

- /home/dev/ben/snortfiles/UMBCNI30.in

Earliest alert at **00:00:08**.796574 *on 02/24/2001*
Latest alert at **23:51:28**.756145 *on 02/24/2001*

Signature (click for sig info)	# Alerts	# Sources	# Destinations
NMAP TCP ping!	1	1	1
Null scan!	7	6	5
Possible RAMEN server activity	8	8	8
Back Orifice	9	1	9
Watchlist 000222 NET-NCFC	17	5	4
WinGate 1080 Attempt	17	11	15
Queso fingerprint	32	9	11
SMB Name Wildcard	67	38	54
Watchlist 000220 IL-ISDNNET-990517	164	6	8
TCP SRC and DST outside network	841	8	9
UDP SRC and DST outside network	21359	94	38

32147 alerts found using input module SnortFileInput, with sources:

- /home/dev/ben/snortfiles/UMBCNI58.in

Earliest alert at **00:00:10**.324284 *on 02/25/2001*
Latest alert at **23:51:59**.864544 *on 02/25/2001*

Signature (click for sig info)	# Alerts	# Sources	# Destinations
Null scan!	9	9	6
TCP SRC and DST outside network	9	6	6
Queso fingerprint	10	5	5
WinGate 1080 Attempt	12	9	11
Watchlist 000222 NET-NCFC	19	1	1
Attempted Sun RPC high port access	23	2	2
SMB Name Wildcard	97	51	60
Possible RAMEN server activity	449	87	244
Watchlist 000220 IL-ISDNNET-990517	979	8	10
SYN-FIN scan!	9336	1	8681
UDP SRC and DST outside network	21204	101	212

No data available for 2001-02-26

20450 alerts found using input module SnortFileInput, with sources:

- /home/dev/ben/snortfiles/UMBCNI54.in

Earliest alert at **00:00:03**.393116 *on 02/27/2001*
Latest alert at **23:51:28**.313911 *on 02/27/2001*

Signature (click for sig info)	# Alerts	# Sources	# Destinations
NMAP TCP ping!	1	1	1
Probable NMAP fingerprint attempt	1	1	1
connect to 515 from inside	1	1	1
Possible RAMEN server activity	3	3	3
Watchlist 000222 NET-NCFC	3	1	1
Null scan!	7	6	6
ICMP SRC and DST outside network	8	1	1
WinGate 1080 Attempt	9	6	7
TCP SRC and DST outside network	16	5	7
Queso fingerprint	82	8	12
SMB Name Wildcard	103	78	67
Watchlist 000220 IL-ISDNNET-990517	284	12	13
SNMP public access	336	2	5
UDP SRC and DST outside network	19596	124	189

38628 alerts found using input module SnortFileInput, with sources:

- /home/dev/ben/snortfiles/UMBCNI52.in

Earliest alert at **00:00:05**.218675 *on 02/28/2001*
Latest alert at **23:52:57**.751617 *on 02/28/2001*

Signature (click for sig info)	# Alerts	# Sources	# Destinations
Tiny Fragments - Possible Hostile Activity	1	1	1
SYN-FIN scan!	1	1	1
ICMP SRC and DST outside network	1	1	1
Null scan!	2	2	2
Watchlist 000222 NET-NCFC	2	1	1
NMAP TCP ping!	3	3	3

Possible RAMEN server activity	12	9	9
Queso fingerprint	12	8	9
TCP SRC and DST outside network	14	8	11
WinGate 1080 Attempt	28	13	18
SMB Name Wildcard	66	41	44
SNMP public access	386	1	3
Watchlist 000220 IL-ISDNNET-990517	3161	10	11
UDP SRC and DST outside network	34939	118	137

SnortSnarf brought to you courtesy of Silicon Defense
Authors: Jim Hoagland and Stuart Staniford
See also the Snort Page by Marty Roesch
Page generated at Wed Jul 18 12:22:03 2001

UDP SRC and DST outside network
TCP SRC and DST outside network

97098 UDP packets were detected that appeared to be both to and from outside MY.NET. Of these 69586 were to the 224.0.0.0 network, i.e. were multicast packets and can be discounted. Of the remaining packets, most originated from, or where destined for, "private" network addresses (RFC1918). Similarly, there were 866 TCP packets detected. Of these, 830 where from 127.0.0.1 (localhost) to 1.1.1.1, which is a "reserved" address.

Watchlist 000220 IL-ISDNNET-990517

This alert is identifying certain addresses from China and Israel. These alerts have been removed from the current Snort ruleset. (per Mark Evans, GCIA)

Possible RAMEN server activity

This rule appears to be triggered by any connection to or from port 27374, which is commonly associated with the Ramen worm in that once a host is "infected", it establishes an HTTP server listening on port 27374. Once the worm compromises another host, it will download the worm from this port on the original host.
http://www.sans.org/y2k/ramen.htm

SNMP public access

This indicates an attempt to read and/or write to the "public" community of an SNMP (Simple Network Management Protocol) enabled host. SNMP by default usually enables read access using the public community, but is not usually configured to allow write access. Often used to monirtor/manage network printers

and other simple devices such as UPS (Uninterruptible Power Supply). SNMP is primarily used to generate alerts to a central station.

SMB Name Wildcard

SMB Name Wildcard is used to request a system's NETBIOS name. Only applies to Windows systems running netbios.

Queso fingerprint

This rule looks for any TCP packet with both reserved bits and the SYN flag set. This used to be a reasonably reliable indication of a fingerprinting attempt. However, this rule is often triggered by ECN-enabled hosts. Newer version of the rule will check for high TTL values as well to try and reduce the false positives. This will be discussed in more detail in the next section.
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids29&view=event

WinGate 1080 Attempt

This rule is triggered by an attempt to access TCP port 1080, which is commonly associated with the SOCKS protocol. This protocol is often used to tunnel new protocols through a firewall. More information can be found at http://www.whitehats.com/info/IDS175.

SYN-FIN scan!

This detect is triggered by TCP packets that have both the SYN and FIN flags set. This in one of the default scanning methods used by NMAP, and was probably originally conceived to avoid detection by IDS systems. As this almost universally now triggers IDS alarms, it can be used as a decoy.

Null Scan!

This appears to be triggered by a TCP packet with **no** flags set, viz:

```
UMBCNI58.in:02/25-19:15:44.335111  [**] Null scan! [**]
129.3.142.137:3423 -> 10.10.224.66:6346

UMBCNI59.in:Feb 25 19:15:44 129.3.142.137:3423 -> 10.10.224.66:6346 NULL
********
```

Both these entries appear to be triggered by the same event.

ICMP SRC and DST outside network

There were 8 packets with this signature all from 10.3.41.11 -> 10.1.40.102, which are both "private" addresses per RFC1918. (See discussion above.)

Top Talkers

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
155.101.21.38	4691	4691	1	1
140.142.19.72	2156	2156	1	1
171.69.248.71	1258	1258	1	1
130.161.180.141	1211	1211	1	1
171.69.33.40	1202	1202	1	1
130.225.127.87	1167	1167	1	1
129.116.65.3	1150	1150	1	1
152.1.1.79	1084	1084	1	1
128.223.83.33	965	965	1	1
130.240.64.20	873	873	1	1

SnortSnarf brought to you courtesy of Silicon Defense
Authors: Jim Hoagland and Stuart Staniford
See also the Snort Page by Marty Roesch
Page generated at Wed Jul 18 12:22:23 2001

Using DNS lookup for each of the above hosts, we find:

Name: bonfire.crsim.utah.edu
Address: 155.101.21.38

Name: netfx.uwtv.washington.edu
Address: 140.142.19.72

Name: tower-u1.cisco.com
Address: 171.69.248.71

Name: dssf.tudelft.nl
Address: 130.161.180.141

can't find 171.69.33.40: Non-existent domain

Name: pc-127-087.adm.ku.dk
Address: 130.225.127.87

Name: vbrick1.ots.utexas.edu
Address: 129.116.65.3

Name: ping.cc.ncsu.edu
Address: 152.1.1.79

Name: iptvhost.uoregon.edu
Address: 128.223.83.33

Name: fix.cdt.luth.se
Address: 130.240.64.20

And checking ARIN for the site without a DNS entry, we find the following netblock entry:

Cisco Systems, Inc. (NETBLK-NETBLK-CISCO-BBLOCK)
170 W. Tasman Dr.
San Jose, CA 95134
US

Netname: NETBLK-CISCO-BBLOCK
Netblock: 171.68.0.0 - 171.69.255.255

Coordinator:
Cisco Systems, Inc. (DN5-ORG-ARIN) dns-info@cisco.com
408.527.9223

Domain System inverse mapping provided by:

NS1.CISCO.COM 192.31.7.92
NS2.CISCO.COM 192.135.250.69
DNS-SJ6.CISCO.COM 192.31.7.93
DNS-RTP4.CISCO.COM 192.135.250.70

Record last updated on 27-Sep-2000.
Database last updated on 20-Jul-2001 23:08:43 EDT.

If we wanted to find out more about, say, the first entry, DNS lookup yields more information:

bonfire.crsim.utah.edu internet address = 155.101.21.38
bonfire.crsim.utah.edu preference = 10, mail exchanger = khoj.crsim.utah.edu
crsim.utah.edu nameserver = ns.inscc.utah.edu
crsim.utah.edu nameserver = ns.utah.edu
crsim.utah.edu nameserver = fiber.utah.edu

khoj.crsim.utah.edu internet address = 155.101.21.95
ns.inscc.utah.edu internet address = 155.101.3.10
ns.utah.edu internet address = 128.110.124.120
fiber.utah.edu internet address = 128.110.132.99

And examining the SOA record:

crsim.utah.edu
origin = ns.inscc.utah.edu
mail addr = root.ns.inscc.utah.edu

we find an email address for the domain at root@ns.inscc.utah.edu, which could be used as a contact of last resort.

We could use sources such as ARIN, RIPE, Network Solutions, etc, etc. to find our more information about these sites if necessary.

List of external source addresses with registration information

This list was collected for the section that follows as potential ECN-enabled hosts (take everything with a pinch of salt!). As above, DNS and ARIN's whois are the main tools used:

129.71.49.10

No DNS records found directly. ARIN reports the following about the netblock:

West Virginia Network for Educational Telecomputing (NET-WVNET)
837 Chestnut Ridge Road
Morgantown, WV 26505
US

Netname: WVNET
Netblock: 129.71.0.0 - 129.71.255.255

Coordinator:
Lynch, Rich (RL104-ARIN) rich@WVNVM.WVNET.EDU
(304) 293-5192

Domain System inverse mapping provided by:

NAMESERV.WVNET.EDU 129.71.1.1
WVNVAXA.WVNET.EDU 129.71.2.1

Record last updated on 27-Feb-1993.
Database last updated on 20-Jul-2001 23:08:43 EDT.

140.247.155.66

Name: roam155-66.student.harvard.edu
Address: 140.247.155.66

Harvard University (NET-HARVARD-COLL)
1 Oxford Street
Cambridge, MA 02138
US

Netname: HARVARD-COLL
Netblock: 140.247.0.0 - 140.247.255.255

Coordinator:
Ouchark, William J. (WJO3-ARIN) ouchark@fas.harvard.edu
617-495-1262 (FAX) 617-495-9285

Domain System inverse mapping provided by:

NS1.HARVARD.EDU 128.103.200.101
NS2.HARVARD.EDU 128.103.1.1

Record last updated on 13-Apr-2001.
Database last updated on 20-Jul-2001 23:08:43 EDT.

141.30.228.58

Name: x05m5b.wh2.tu-dresden.de
Address: 141.30.228.58

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

inetnum: 141.30.0.0 - 141.30.255.255
netname: TUDR
descr: Technische Universitaet Dresden
country: DE
admin-c: WW155
tech-c: VR27-RIPE
status: ASSIGNED PI
mnt-by: DFN-NTFY

changed: knocke@nic.de 19931220
changed: poldi@dfn.de 20000719
source: RIPE

158.75.57.4

No DNS records, but ARIN reports:

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

Domain System inverse mapping provided by:

ALFA.CS.TORUN.PL 158.75.10.75
BILBO.NASK.ORG.PL 148.81.16.51

Record last updated on 11-Oct-1995.
Database last updated on 20-Jul-2001 23:08:43 EDT.

And RIPE seems to agree:

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

inetnum: 158.75.0.0 - 158.75.255.255
netname: TORUNPOLIP2
descr: Nicolaus Copernicus University
descr: University Networking Technology Centre
country: PL
admin-c: JZ56
tech-c: MGW29
status: ASSIGNED PI
notify: M.Gorecka-Wolniewicz@cc.uni.torun.pl
mnt-by: RIPE-NCC-NONE-MNT
changed: M.Gorecka-Wolniewicz@cc.uni.torun.pl 20000614

source: RIPE

203.45.200.122

Name: CPE-203-45-200-122.qld.bigpond.net.au
Address: 203.45.200.122

From APNIC (we seem to be getting further and further afield – now we are in Australia!). Search the APNIC Whois database

Search results for '203.45.200.122'

inetnum 203.40.0.0 - 203.47.255.255
netname TELSTRAINTERNET2-AU
descr Telstra Internet
descr Locked Bag 5744
descr Canberra
descr ACT 2601
country AU
admin-c GH169-AP, inverse
tech-c GH169-AP, inverse
remarks ** Conversion note - reference 'GH29-AU' changed to 'GH169-AP'
remarks Record imported from AUNIC as part of AUNIC->APNIC migration
remarks Please see http://www.apnic.net/db/aunic/
mnt-by MAINT-AU-GH169-AP, inverse
changed register@aunic.net 19961120
changed register@aunic.net 20000105
changed aunic-transfer@apnic.net 20010525
source APNIC

207.96.122.8

Name: gwyn.tux.org
Address: 207.96.122.8

And from ARIN:

RCN Corporation (NET-RCN-BLK-1)
105 Carnegie Center
Princeton, NJ 08540
US

Netname: RCN-BLK-1
Netblock: 207.96.0.0 - 207.96.127.255

Maintainer: RCN

Coordinator:
RCN Corporation (ZR40-ARIN) noc@rcn.com
703-426-4335

Domain System inverse mapping provided by:

AUTH1.DNS.RCN.NET	207.172.3.20
AUTH2.DNS.RCN.NET	206.138.112.20
AUTH3.DNS.RCN.NET	207.172.3.21
AUTH4.DNS.RCN.NET	207.172.3.22

Record last updated on 04-Apr-2001.
Database last updated on 20-Jul-2001 23:08:43 EDT.

209.250.47.5

Name: cfelts.apex.net
Address: 209.250.47.5
From ARIN:
Apex Internet Services (NETBLK-APEX-BLK-1)
1158 Jefferson St.
PADUCAH, KY 42001
US

Netname: APEX-BLK-1
Netblock: 209.250.32.0 - 209.250.63.255
Maintainer: APEX

Coordinator:
Campbell, Jay (JC1328-ARIN) jay@APEX.NET
502-442-5363 (FAX) (502) 442-2685

Domain System inverse mapping provided by:

APEX.APEX.NET	207.87.196.20
RABBIT.APEX.NET	207.87.196.24

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

Record last updated on 01-May-2001.
Database last updated on 20-Jul-2001 23:08:43 EDT.

Also, from Network Solutions:

Registrant:

Apex Internet Services (APEX2-DOM)
1301 Broadway
Paducah, KY 42001
US

Domain Name: APEX.NET

Administrative Contact, Billing Contact:
Campbell, Jay (JC1328) jay@APEX.NET
1158 Jefferson St.
Paducah, KY 42003
(502) 442-5363 (FAX) (502) 442-2685
Technical Contact:
Carneal, Jeff (JC7725) jeff@APEX.NET
Apex Internet Services, Inc.
1301 Broadway
Paducah, KY 42001
270-442-5363 (FAX) 270-442-2685

Record last updated on 11-Nov-2000.
Record expires on 24-Oct-2001.
Record created on 23-Oct-1995.
Database last updated on 21-Jul-2001 02:52:00 EDT.

Domain servers in listed order:

APEX.APEX.NET 209.250.48.20
NS2.APEX.NET 209.250.48.24
DCA-ANS-01.INET.QWEST.NET 205.171.9.242
SVL-ANS-01.INET.QWEST.NET 205.171.14.195

213.132.151.11

No DNS information, but RIPE gives us a Belgian address:

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

inetnum: 213.132.128.0 - 213.132.155.255
netname: TVD-INTERNET

descr: TVD Internet - UPC Belgium - Chello
descr: ISP - CATV operator Brussels/Leuven
descr: Customer DHCP pools
country: BE
admin-c: TVD-RIPE
tech-c: TVD-RIPE
rev-srv: bruns00.chello.com
rev-srv: bruns01.chello.be
rev-srv: ns1.chello.at
status: ASSIGNED PA
remarks: If you suspect unusual activity originating from this network:
remarks: send relevant logfiles, IP adresses, protocol and port numbers,
remarks: UTC timestamps and other useful information to abuse@chello.be
remarks: in plain ascII text. Do not send HTML, proprietary encodings,
remarks: graphical formats or mime attachments.
notify: ripe-notify@tvd.be
mnt-by: AS8733-MNT
changed: steven@tvd.be 20001222
source: RIPE

24.169.71.180

Name: syr-24-169-71-180.twcny.rr.com
Address: 24.169.71.180
From ARIN:
ServiceCo LLC - Road Runner (NET-ROAD-RUNNER-5)
13241 Woodland Park Road
Herndon, VA 20171
US

Netname: ROAD-RUNNER-5
Netblock: 24.160.0.0 - 24.170.127.255
Maintainer: SCRR

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

Domain System inverse mapping provided by:

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

Record last updated on 14-Jun-2001.
Database last updated on 20-Jul-2001 23:08:43 EDT.

This sounds like cable-modem land….

24.68.101.99

No DNS entry, but ARIN reports:

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

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

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

Domain System inverse mapping provided by:

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

Record last updated on 12-Jul-2000.
Database last updated on 20-Jul-2001 23:08:43 EDT.

This is definitely cable-modem land – they used to be my network services supplier!

Link Graph and Analysis of OOS Files

Examining UMBCIN37.in:

```
awk '
$0 ~ /=\+=\+=/  { getline; ip = $2
          getline
              if ( $3 ~ /TOS:0x0/ ) {
                  getline
                  if ( $1 ~ /21S/ ) { print ip }
              }
}
' <filename> | cut -d: -f1 | sort -u
```

generates a list of unique source IP addresses for all packets that have a TOS = 0x0, and have the SYN, ECN-CWR, and ECN-echo bits set in the TCP flags field of the IP header. This is the signature of an initial SYN connection from an ECN-capable host. It is also a possible signature of a QUESO or NMAP fingerprinting attempt. The response to an ECN SYN by an ECN-capable host is to set the ECN-echo and SYN/ACK flags in the TCP flags field. Thereafter, all packets (except pure ACKs) will have bit 6 of the TOS field set (ECT).

Running the above script on the file UMBCIN37.in, we get 25 hosts that **may** be ECN capable. This list is sorted, and MY.NET appears first, so examining them:

10.10.220.6 --not likely, TCP flags are 21SFR*** for the only packet logged
10.10.97.96 --probably ECN-capable, no OOS response to any SYN
10.10.98.123 --ditto
10.10.98.161 --ditto
10.10.98.164 --ditto
10.10.98.201 --ditto – this site had an incoming OOS packet:

```
01/23-10:31:53.524389 162.33.212.88:22 -> 10.10.98.201:37277
TCP TTL:56 TOS:0x0 ID:14351
21S***A* Seq: 0x53065837   Ack: 0x4EC8543A   Win: 0x7FE0
TCP Options => MSS: 1460
00 00                                          ..
```

From my reading of Toby Miller's note on ECN (http://www.sans.org/y2k/ecn.htm), this is not a valid response as it has both ECN-CWR and ECN-echo flags set. However, it is probably an acceptable interpretation of the RFC. Examining both packets together, this looks like two parts of the three-way handshake to the SSH port from one ECN-capable host to another.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/23-10:31:53.356471 10.10.98.201:37277 -> 162.33.212.88:22
TCP TTL:62 TOS:0x0 ID:0  DF
21S***** Seq: 0x4EC85439   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 118306184 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/23-10:31:53.524389 162.33.212.88:22 -> 10.10.98.201:37277
TCP TTL:56 TOS:0x0 ID:14351
21S***A* Seq: 0x53065837   Ack: 0x4EC8543A   Win: 0x7FE0
TCP Options => MSS: 1460
00 00                                          ..

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

The other sites found are all external to MY.NET (aka 10.10.0.0). As we have already determined that there were no valid ECN-host to ECN-host connections established, we cannot tell from this analysis if these sites are ECN-capable, or not.

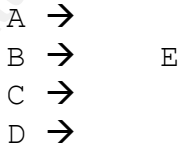
129.71.49.10
140.247.155.66

141.30.223.141
141.30.228.120
141.30.228.122
141.30.228.155
141.30.228.182
141.30.228.226
141.30.228.58
158.75.57.4
203.45.200.122
204.210.120.196
205.215.217.90
207.96.122.8
209.250.47.5
213.132.151.11
24.169.71.180
24.179.205.229
24.68.101.99

Using the following script on all the OOS files in the target period 2001-02-24 to 2001-02-28:

```
grep '\->' *37* *56* | awk '{print $2,$3,$4}' | grep 10.10.229.242 | sort
-u
```

Graphing the results, we see:
141.30.228.182 -> 10.10.229.242:6346
141.30.228.199 -> 10.10.229.242:6346
194.51.109.194 -> 10.10.229.242:6346
202.166.38.138 -> 10.10.229.242:6346



This appears to be one-way communications, but it is more likely that the other half of the communication was not trapped.

Using the portscan logs only for the period 2001-02-24 to 2001-02-28, looking for Gnutella activity (http://www.sans.org/y2k/gnutella.htm), as follows:

```
grep :6346 UMBCNI35.in UMBCNI59.in UMBCNI57.in UMBCNI55.in
UMBCNI53.in | awk '{print $4, $6}' > gnutella
cut -d: -f1 gnutella | sort -u
```

yielded a total of 23 different systems within MY.NET and another 35 hosts outside MY.NET. This was found by only looking for port 6346, which is a configurable parameter in the Gnutella protocol.

Compromised systems?

The Snort Alerts identified some possible Ramen activity. Further examination of the events that caused the rule to be triggered is warranted.

There were two hosts inside MY.NET that appeared to be receiving or generating this traffic.

10.10.60.11

This host was the source of a large number of high-port (RPC?) scans of two particular hosts – 209.73.164.91 on Jan 21st at 20:33, and 4.17.168.6 at 23:01 on the same day.

```
SnortA3.in:02/03-08:39:26.465688  [**] spp_portscan: PORTSCAN DETECTED
from 10.10.60.11 (THRESHOLD 7 connections in 2 seconds) [**]
SnortA3.in:02/03-08:39:27.866973  [**] spp_portscan: portscan status from
10.10.60.11: 16 connections across 15 hosts: TCP(0), UDP(16) [**]
SnortA3.in:02/03-08:39:30.057556  [**] spp_portscan: End of portscan from
10.10.60.11 (TOTAL HOSTS:15 TCP:0 UDP:16) [**]
SnortA3.in:02/03-15:54:29.479974  [**] spp_portscan: PORTSCAN DETECTED
from 10.10.60.11 (THRESHOLD 7 connections in 2 seconds) [**]
SnortA3.in:02/03-15:54:31.707434  [**] spp_portscan: portscan status from
10.10.60.11: 9 connections across 1 hosts: TCP(0), UDP(9) [**]
SnortA3.in:02/03-15:54:34.119538  [**] spp_portscan: End of portscan from
10.10.60.11 (TOTAL HOSTS:1 TCP:0 UDP:9) [**]
SnortA3.in:02/03-20:04:21.221498  [**] spp_portscan: PORTSCAN DETECTED
from 10.10.60.11 (THRESHOLD 7 connections in 2 seconds) [**]
…….
```

This continued until 22:38 (with a gap from 20:36 to 22:17), during which time about 350 hosts were scanned, each scan ranging from about 1000 to 3000 ports per host.

The Ramen activity alert was triggered on Feb 23rd at 8:44. There seems to have been a period of about 18 minutes where traffic was exchanged:

```
UMBCNI27.in:02/23-08:44:38.793830  [**] Possible RAMEN server activity
[**] 10.10.60.11:23 -> 148.129.143.2:27374
UMBCNI27.in:02/23-08:44:38.822907  [**] Possible RAMEN server activity
[**] 148.129.143.2:27374 -> 10.10.60.11:23
...
UMBCNI27.in:02/23-09:02:41.741020  [**] Possible RAMEN server activity
[**] 10.10.60.11:23 -> 148.129.143.2:27374
UMBCNI27.in:02/23-09:02:43.917828  [**] Possible RAMEN server activity
[**] 10.10.60.11:23 -> 148.129.143.2:27374
```

However, closer examination shows that the ports involved were 27374 (the Ramen http port) and port 23 (telnet). So this is much more likely to be a normal telnet session in which the remote site just happened to select the ephemeral port 27374 for the connection. This is not conclusive, but is a reasonable supposition.

There is more evidence that this is indeed a telnet server, as a connection was made on Feb 20th from 4:41 to 4:47 from 159.226.45.108:1046. This may seem like an odd time,

but the source IP is located in Beijing, and the connection was detected as this site is on the now-defunct watch list. It would certainly be worthwhile to look at the syslog files for this host to see what was accessed. Perhaps we have a student in Beijing?

On Feb 23rd, our host was scanned apparently from 24.7.160.69 at 10:32. About 90 ports appear to have been scanned at random.

Also on Feb 23rd at 20:19, there was traffic to or from host 216.136.173:110 to 10.10.60.11:27374. Is this evidence of Ramen activity? I do not think so! It is more likely that a user is picking up his email -- port 110 is POP3 – and, once again, the system may have selected 27374 as an ephemeral port.

10.10.201.146

This host was the source of a large number of UDP scans of various hosts throughout the month. These are just a few examples taken at random, merely to show the consistency:

```
SnortS2.in:Feb  4 15:41:06 10.10.201.146:3598 -> 213.221.174.8:27060 UDP
SnortS2.in:Feb  4 15:41:06 10.10.201.146:3605 -> 213.221.174.35:27018 UDP
SnortS2.in:Feb  4 15:41:06 10.10.201.146:3613 -> 213.221.174.200:28503
UDP
SnortS2.in:Feb  4 15:41:06 10.10.201.146:3622 -> 213.221.174.101:27001
UDP
SnortS2.in:Feb  4 15:41:06 10.10.201.146:3679 -> 212.21.97.81:27051 UDP
SnortSca.in:Feb  5 12:12:39 10.10.201.146:4551 -> 63.236.5.137:27040 UDP
SnortSca.in:Feb  5 12:12:39 10.10.201.146:4553 -> 63.236.5.135:27040 UDP
SnortSca.in:Feb  5 12:12:41 10.10.201.146:4584 -> 62.236.88.9:27020 UDP
SnortSca.in:Feb  5 12:12:39 10.10.201.146:4588 -> 62.180.250.184:27005
UDP
SnortSca.in:Feb  5 12:12:39 10.10.201.146:4589 -> 62.180.250.182:27033
UDP
SnortS34.in:Feb  6 03:15:57 10.10.201.146:2213 -> 194.158.97.90:27018 UDP
SnortS34.in:Feb  6 03:15:58 10.10.201.146:2213 -> 203.164.49.162:1025 UDP
SnortS34.in:Feb  6 03:15:59 10.10.201.146:2213 -> 213.61.7.82:27103 UDP
SnortS34.in:Feb  6 03:16:00 10.10.201.146:2213 -> 151.39.100.34:27023 UDP
SnortS34.in:Feb  6 03:16:00 10.10.201.146:2213 -> 195.149.21.17:27065 UDP
SnortS34.in:Feb  6 03:16:00 10.10.201.146:2213 -> 130.233.22.34:27777 UDP
SnortS32.in:Feb  7 00:14:29 10.10.201.146:3719 -> 212.67.96.97:27018 UDP
SnortS32.in:Feb  7 00:14:29 10.10.201.146:3756 -> 212.162.240.10:27026
UDP
SnortS32.in:Feb  7 00:14:29 10.10.201.146:3768 -> 212.140.216.26:37033
UDP
SnortS32.in:Feb  7 00:14:29 10.10.201.146:3769 -> 212.140.216.25:37032
UDP
SnortS32.in:Feb  7 00:14:29 10.10.201.146:3731 -> 212.40.5.36:27045 UDP
SnortS29.in:Feb  9 00:42:52 10.10.201.146:3715 -> 216.185.105.10:27018
UDP
SnortS29.in:Feb  9 00:42:52 10.10.201.146:3752 -> 213.83.18.165:20240 UDP
SnortS29.in:Feb  9 00:42:52 10.10.201.146:3753 -> 213.83.18.163:20100 UDP
SnortS29.in:Feb  9 00:42:52 10.10.201.146:3768 -> 213.224.236.250:61022
UDP
SnortS29.in:Feb  9 00:42:52 10.10.201.146:3767 -> 213.229.11.81:27020 UDP
SnortS29.in:Feb  9 00:42:52 10.10.201.146:3783 -> 213.221.174.200:28507
UDP
SnortA25.in:02/11-02:54:24.161069  [**] spp_portscan: End of portscan
from 10.10.201.146 (TOTAL HOSTS:40 TCP:0 UDP:44) [**]
```

The event(s) that triggered the Ramen activity rule:

```
UMBCNI27.in:02/23-03:15:50.748824  [**] Possible RAMEN server activity
[**] 128.138.2.112:27374 -> 10.10.201.146:4781
UMBCNI27.in:02/23-03:15:50.750040  [**] Possible RAMEN server activity
[**] 128.138.2.112:27374 -> 10.10.201.146:4781
UMBCNI27.in:02/23-03:15:50.989401  [**] Possible RAMEN server activity
[**] 128.138.2.112:27374 -> 10.10.201.146:4781
UMBCNI27.in:02/23-03:15:51.046519  [**] Possible RAMEN server activity
[**] 128.138.2.112:27374 -> 10.10.201.146:4781
UMBCNI27.in:02/23-03:15:51.229937  [**] Possible RAMEN server activity
[**] 128.138.2.112:27374 -> 10.10.201.146:4781
...
UMBCNI27.in:02/23-03:17:51.299698  [**] Possible RAMEN server activity
[**] 128.138.2.112:27374 -> 10.10.201.146:4781
UMBCNI27.in:02/23-03:17:51.362339  [**] Possible RAMEN server activity
[**] 128.138.2.112:27374 -> 10.10.201.146:4781
UMBCNI27.in:02/23-03:17:51.381122  [**] Possible RAMEN server activity
[**] 10.10.201.146:4781 -> 128.138.2.112:27374
UMBCNI27.in:02/23-03:17:51.381172  [**] Possible RAMEN server activity
[**] 10.10.201.146:4781 -> 128.138.2.112:27374
```

Fairly clearly, there was two-way traffic during this period. Also, unlike the previous event, port 4781 is not a WKS port (Well Known Service) but is an ephemeral port. So this is quite likely another indication that this host has been compromised

It seems reasonable to assume that this system needs to be investigated. Even if it is does not have the Ramen worm, it is still acting in an unacceptable manner.

Recommendations

Create an "Acceptable Use Policy" that is acceptable to all users, publish it where it must be seen and read, and ensure that it has sufficient response provisions in the event that the policy is contravened. Without such a policy in place, all attempts at security are doomed to fail.

Install (or update) firewalls at all access points (i.e. connections to the internet and any other networks). Use a default deny-all policy. This may take a while to enforce, but is worth the effort. Also install egress filtering to ensure that we are not the source of attacks!

Position IDS sensors to give the widest possible coverage. Monitor as much of the internal network as possible, as well as the gateways.

Remove (or at least discourage) the use of modems on any system that is connected to the network. A central modem pool could be created if necessary.

Ensure that all hosts under your control are managed by qualified, security-conscious system administrators.

Make a selection of easily installable tools available on-line – preferably pre-configured for automatic installation, and encourage their use. The list of tools might include tcp-wrappers (ftp://ftp.cerias.purdue.edu/pub/tools/unix/netutils/tcp_wrappers/) for those systems that do not have this functionality built-in, COPS, L6 or tripwire, etc. etc. The list goes on….

Try and replace telnet, ftp, and especially the 'r' commands (rsh, rlogin, rexec) with SSH.

And, finally, keep looking! Even with the best defences in place, we are still vulnerable.

Analysis process

I have tried to include the tools that I have used, and the steps that I took at each stage of the analysis. Rather than describe then all again here, I will just list them:

SnortSnarf v052301.1 -- perl based toolkit that parses Snort-generated files, and presents results as HTML pages.

Standard unix command line tools, such as awk, sed, grep, cut, vi, more.

Many internet resources, which I have tried to list as I referred to them. My apologies to any that I may have inadvertently omitted.

Conclusions

We have analyzed the data provided by the University of XXX. We have identified some common events and have found evidence of possibly compromised systems. We have also prepared a list of recommendations that should help the university establish a leading position in the security field.

References

Stevens, W. R. 1994. *TCP/IP Illustrated: the protocols*. Addison-Wesley, Reading, Mass.

Aho, A. V. 1988. *The AWK programming language*. Addison-Wesley, Reading, Mass.

Garfinkel, S., Spafford, G. 1996. *Practical UNIX and Internet Security, second edition*. O'Reilly & Associates, Sebastopol, Calif.

Northcutt, S. 1999. *Network Intrusion Detection: an analyst's handbook*. New Riders, Indianapolis, Indiana.

Useful links that I have bookmarked in my browser

Google - general internet search engine

Current RFC Standards

Welcome to incidents.org - By The SANS Institute

Incidents Diary Handler's Diary at incidents.org

Search - Neohapsis Archives - http://archives.neohapsis.com/ Google powered search

Global Incident Analysis Center

CERT Coordination Center

Common Vulnerabilities and Exposures

Whitehats Network Security Resource

SILICON DEFENSE SnortSnar f Log Alert analyzer

s e c u r i t y f o c u s -- Bugtraq

Computer Security Information

Security Software

Yahoo! Computers and Internet:Software:Operating Systems:Unix:Security

L0pht Heavy Industries

ISS - Internet Security Systems

SSH - Products - SSH IPSEC Express

Computer Forensics Column -- links to articles by Dan Farmer and Wietse Venema

The Digital Offense

Sys-Security.com - Because Security is not Trivial

DShield. - Distributed Intrusion Detection System

myNetWatchman.com - Intrusion Reporting and Response

Passive Fingerprinting