Global Information Assurance Certification Paper

Chris Murphy

GIAC LevelTwo Intrusion Detection In Depth
Practical Assignment for Capitol SANS
December 10 – 15, 2000

Assignment 1- Network Detects (30 Points)

Notes about the following detects and log fidelity:

I collected the following detects with the Snort intrusion detection system (http://www.snort.org) running in "performance configuration" mode – *./snort -b -A fast -c vision.conf* – because I was concerned with the speed of the processor and the network adapter on the sensor and the likelihood of dropped packets. Therefore, the following detects do not include the application layer info which would have provided more detail and helped further analysis in some cases.

Key to understanding Snort alert log format:

```
[**] IDS198/SYN FIN Scan [**]
01/07-08:04:32.424823 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
a.b.c.d:110-> w.x.y.z:110 TCP TTL:26 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x70D962EE Ack: 0x1D5DAF0C Win: 0x404 TcpLen: 20
```

- arachNIDS IDSKEY and EVENT NAME (http://www.whitehats.com/ids/index.html)
- date and time stamp (accurate to milliseconds)
- source physical (MAC) address
- destination physical (MAC) address
- Ethernet frame type
- Ethernet header length
- source IP address
- source port
- destination IP address
- destination port
- protocol
- time to live
- type of service
- IP identification
- IP header length
- datagram length
- TCP flags
- sequence number
- acknowledgment number
- TCP window size
- TCP header length

Detect 1

```
[**] IDS448/ping-SING Echo from Sun Solaris [**]
01/05-23:40:25.335976 0:E0:B0:63:DE:2A -> 0:E0:1E:0:0:0 type:0x800 len:0x3C
a.b.c.d -> w.x.y.222 ICMP TTL:50 TOS:0x0 ID:2850 IpLen:20 DgmLen:36
Type:8 Code:0 ID:1913 Seq:26581 ECHO
```

```
[**] IDS448/ping-SING Echo from Sun Solaris [**]
01/05-23:41:47.434929 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
a.b.c.d -> w.x.y.33 ICMP TTL:50 TOS:0x0 ID:7108 IpLen:20 DgmLen:36
Type:8 Code:0 ID:6265 Seq:58393 ECHO

[**] IDS448/ping-SING Echo from Sun Solaris [**]
01/05-23:48:23.318041 0:E0:B0:63:DE:2A -> 0:D0:B7:0:0:0 type:0x800 len:0x3C
a.b.c.d -> w.x.y.215 ICMP TTL:50 TOS:0x0 ID:27447 IpLen:20 DgmLen:36
Type:8 Code:0 ID:8825 Seq:28227 ECHO

[**] IDS448/ping-SING Echo from Sun Solaris [**]
01/05-23:48:50.783632 0:E0:B0:63:DE:2A -> 0:8:C7:0:0:0 type:0x800 len:0x3C
a.b.c.d -> w.x.y.251 ICMP TTL:50 TOS:0x0 ID:28741 IpLen:20 DgmLen:36
Type:8 Code:0 ID:30073 Seq:65167 ECHO

[**] IDS448/ping-SING Echo from Sun Solaris [**]
01/05-23:53:29.529876 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
a.b.c.d -> w.x.y.26 ICMP TTL:50 TOS:0x0 ID:45521 IpLen:20 DgmLen:36
Type:8 Code:0 ID:13433 Seq:49804 ECHO
```

1. Source of Trace

My employer's network

2. Detect was generated by:

Snort intrusion detection system

Triggered by this rule:

```
alert ICMP $EXTERNAL any -> $INTERNAL any (msg: "IDS448/ping-SING Echo
from Sun Solaris"; dsize: 8; itype: 8;)
```

3. Probability the source address was spoofed:

There is almost zero probability the source address was spoofed. The SING utility illustrated in the trace above is used to gather information about a host. The attacker needs to see the results provided by the SING scan so he/she will use a valid IP address.

4. Description of attack:

The SING ("Send ICMP Nasty Garbage") tool allows users to craft ICMP datagrams. It compiles on Solaris, *BSD and Linux and is available from http://sourceforge.net/projects/sing.

Ofir Arkin, in his paper entitled *Identifying ICMP Hackery Tools Used In The Wild Today* (http://www.sys-security.com/archive/securityfocus/icmptools.html), says that crafted ICMP datagrams "can be used for various tasks: host detection, advanced host detection, Operating System Fingerprinting and more."

This attack was reconnaissance. The attacker either wanted to know if these hosts were alive or what operating systems they were running.

The operating system on a host is a valuable piece of information to an attacker. Once the OS is known, the vulnerabilities specific to that platform are also known, and the attacker can focus the search for relevant pre-fabricated exploit code to use to attack the target.

5. Attack mechanism:

This is an example of SING in action against a test host running the Windows NT 4.0 operating system:

```
bigeye# sing -c 1 -O 192.168.0.1
SINGing to 192.168.0.1 (192.168.0.1): 16 data bytes
16 bytes from 192.168.0.1: seq=0 ttl=128 TOS=0 time=6.932 ms WIN!

--- 192.168.0.1 sing statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 6.932/6.932/6.932 ms
```

Note that it correctly identifies the operating system as Windows – "WIN!" – after sending only one datagram.

6. Correlations:

I could find no evidence of SING scans reported by others, nor could I associate the source IP addresses with other malicious activity. This may just be because there are more popular utilities available for host detection and OS fingerprinting -- for example, PING and nmap.

Ofir Arkin has done extensive research on the SING utility in particular and the malicious uses of ICMP in general. Related documents can be found at his web site – http://www.sys-security.com.

7. Evidence of active targeting:

Assuming that the Snort sensor did not experience a marked packet loss, the attacker appeared to target five hosts on my employer's class C subnet over the course of fifteen minutes.

8. Severity:

(criticality + lethality) – (system countermeasures + network countermeasures) = severity

Severity is (4 + 1) – (4 + 0) = 1

Five hosts were scanned. Some are critical (DNS) and others aren't (web, ftp): 4
There is no evidence of a lethal attack here. This was just information gathering: 1
OSs are hardened and applications are patched: 4
There is no firewall. These hosts reside in a true DMZ: 0

9. Defensive recommendation:

Restrict ACLs on border router to block the potentially dangerous ICMP protocols.

```
access-list 110 deny ICMP 0.0.0.0 255.255.255.255 time-exceeded
access-list 110 deny ICMP 0.0.0.0 255.255.255.255 echo-reply
```

10. Multiple choice test question:

The ICMP protocol was designed for error reporting on IP networks but it can be abused for malicious purposes. Which of the following are examples of ICMP protocol misuse?

a) denial of service attacks
b) OS fingerprinting

c) clandestine tunneling mechanism
d) network mapping and reconnaissance
e) none of the above
f) all of the above

The correct answer is f, “all of the above”

Detect 2

```
[**] IDS296/http-whisker-splicing-attack-space [**]
01/06-01:10:09.603096 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
a.b.c.d:16518-> w.x.y.30:80 TCP TTL:114 TOS:0x0 ID:28025 IpLen:20 DgmLen:41 DF
***AP*** Seq: 0xFD6AAAFA Ack: 0x84E19BB1 Win: 0x4470 TcpLen: 20

[**] IDS296/http-whisker-splicing-attack-space [**]
01/06-01:10:09.603796 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
a.b.c.d:16518-> w.x.y.30:80 TCP TTL:114 TOS:0x0 ID:28027 IpLen:20 DgmLen:41 DF
***AP*** Seq: 0xFD6AAAFC Ack: 0x84E19BB1 Win: 0x4470 TcpLen: 20

[**] IDS296/http-whisker-splicing-attack-space [**]
01/06-13:49:43.710621 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
a.b.c.d:15720-> w.x.y.32:80 TCP TTL:114 TOS:0x0 ID:54250 IpLen:20 DgmLen:41 DF
***AP*** Seq: 0xC3119EE2 Ack: 0x8798EE59 Win: 0x4470 TcpLen: 20

[**] IDS296/http-whisker-splicing-attack-space [**]
01/06-13:49:43.712110 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
a.b.c.d:15720-> w.x.y.32:80 TCP TTL:114 TOS:0x0 ID:54252 IpLen:20 DgmLen:41 DF
***AP*** Seq: 0xC3119EE4 Ack: 0x8798EE59 Win: 0x4470 TcpLen: 20

[**] IDS296/http-whisker-splicing-attack-space [**]
01/06-13:49:54.026095 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
a.b.c.d:15720-> w.x.y.32:80 TCP TTL:114 TOS:0x0 ID:62150 IpLen:20 DgmLen:41 DF
***AP*** Seq: 0xC3119EF2 Ack: 0x8798F3A3 Win: 0x3F26 TcpLen: 20

[**] IDS296/http-whisker-splicing-attack-space [**]
01/06-13:49:54.029328 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
a.b.c.d:15720-> w.x.y.32:80 TCP TTL:114 TOS:0x0 ID:62160 IpLen:20 DgmLen:41 DF
***AP*** Seq: 0xC3119EFC Ack: 0x8798F3A3 Win: 0x3F26 TcpLen: 20
```

1. Source of Trace

My employer’s network

2. Detect was generated by:

Snort intrusion detection system

Triggered by this rule:

```
alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS296/http-whisker-splicing-
attack-space"; dsize: <5; flags: AP; content: "|20|";)
```

3. Probability the source address was spoofed:

There is almost zero probability the source address was spoofed. The Whisker utility is a web server vulnerability scanner. The attacker needs to see the results provided by the scan so he/she will use a valid IP address.

4. Description of attack:

Whisker is web/cgi scanner written in Perl by Rain Forest Puppy. The latest version, v1.4, is available on the web at http://www.wiretrip.net/rfp/p/doc.asp?id=21&iface=2.

It includes a switch to break the scan into very small fragments in order to circumvent intrusion detection systems. In this case, it was detected by Snort nonetheless.

5. Attack mechanism:

The command used by the attacker was something similar to this:

```
./whisker.pl -v -i -h w.x.y.30 -s custom.db -I 9
```

A Whisker scan using the default database (scan.db) and the "-I 9" IDS evasion switch will produce 3176 Snort alerts. In this case, Snort only detected six alerts across two hosts. Assuming that the Snort sensor did not experience a marked packet loss, the attacker either was using a highly tuned scan.db or terminated the scan prematurely. Most likely, the attacker was probing for one particular vulnerability—for example, the msadcs.dll hole which widely affected Microsoft's Internet Information Server – or only using Whisker's "server type check" to identify the web server platform.

Without the application layer info or the web server logs (which I do not have) its impossible to say precisely what the attacker was probing for.

6. Correlations:

I could find no evidence of Whisker scans by others, nor could I associate the source IPs with other malicious activity.

Whisker's release notes provide detailed information on usage.

7. Evidence of active targeting:

Yes, the attacker focused only on two hosts in a class C subnet—both publicly accessible web servers.

8. Severity:

(criticality + lethality) – (system countermeasures + network countermeasures) = severity

Severity is (3 + 3) – (4 + 0) = 2

These are public web servers—important but not critical: 3
This attack was a scan for vulnerabilities--not dangerous by itself but may precede a root compromise: 3
These web servers are patched against the popular IIS exploits so Whisker will not have reported any vulnerabilities: 4
There is no firewall. These hosts reside in a true DMZ: 0

9. Defensive recommendation:

These are publicly accessible web servers, and since Whisker uses the well-known port 80 for httpd, firewall policies and restrictive router ACLs would not prevent these scans without hindering legitimate traffic. I recommend focusing on the OS and web server

software and continue to monitor vendor security bulletins and apply security patches as necessary.

10. Multiple choice test question:

In the traces above, the TCP PSH (or "push") flag is set. This flag indicates:

a) the TCP sliding window is too small
b) the receiver should pass this data to the application as soon as possible
c) an error occurred so the sender should re-transmit the data
d) the receive buffer on the destination host is full

The correct answer is b, "the receiver should pass this data to the application as soon as possible"

Detect 3

```
[**] IDS10/portmap-request-rstatd [**]
01/07-05:27:14.861994 0:E0:B0:63:DE:2A -> 0:D0:B7:0:0:0 type:0x800 len:0x62
a.b.c.d:923-> w.x.y.215:111 UDP TTL:46 TOS:0x0 ID:60598 IpLen:20 DgmLen:84
Len: 64
```

1. Source of Trace

My employer's network

2. Detect was generated by:

Snort intrusion detection system

Triggered by this rule:

```
alert UDP $EXTERNAL any -> $INTERNAL 111 (msg: "IDS10/portmap-request-
rstatd"; content: "|01 86 A0 00 00|";)
```

3. Probability the source address was spoofed:

There is almost zero probability that the source address is spoofed. The trace above illustrates a remote request for system information from the rpc.statd server. The attacker wants to see the response provided by the inquiry so he/she will use a valid IP address.

4. Description of attack:

The attacker is querying the RPC portmapper daemon to see whether the rpc.statd service is running. The rpc.statd service provides performance statistics obtained from the kernel in support of the NFS (Network File System) service on UNIX-like systems.

Vulnerabilities in the rpc.statd service were reported in August 2000 and affected several distributions of Linux. These vulnerabilities were widely exploited and thousands of Linux machines were compromised.

Reference:

CVE-2000-0666

Bugtraq ID 1480
CERT:CA-2000-17

Note: the time and date of this detect preceded the prolific “Ramen worm” which incorporated rpc.statd exploit code to compromise vulnerable Red Hat Linux 6.2 hosts and was first reported by SANS on January 18, 2001.

5. Attack mechanism:

The tool being used could be part of an automated RPC scanner and exploit program.

A search of the vulnerability database at http://www.securityfocus.com revealed four exploit programs for the rpc.statd vulnerability:

statd-toy.c
rpc-statd-xpl.c
statdx.c
statdx2.tar.gz

Here are the usage instructions for statdx:

```
bigeye# ./statdx -h
statdx by ron1n <shellcode@hotmail.com>
Usage: ./statdx [-t] [-p port] [-a addr] [-l len]
        [-o offset] [-w num] [-s secs] [-d type] <target>
-t      attack a tcp dispatcher [udp]
-p      rpc.statd serves requests on <port> [query]
-a      the stack address of the buffer is <addr>
-l      the length of the buffer is <len> [1024]
-o      the offset to return to is <offset> [600]
-w      the number of dwords to wipe is <num> [9]
-s      set timeout in seconds to <secs> [5]
-d      use a hardcoded <type>
Available types:
0       Redhat 6.2 (nfs-utils-0.1.6-2)
1       Redhat 6.1 (knfsd-1.4.7-7)
2       Redhat 6.0 (knfsd-1.2.2-4)
```

However, it might just as well be the rpcinfo utility available on most UNIX-like systems.

```
bigeye# rpcinfo -p www.xxx.yyy.215
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
```

In this case, the only RPC program registered with the portmap daemon is the portmapper itself.

6. Correlations:

I could not associate the source IP with other malicious activity. However, this vulnerability was widespread and numerous Linux hosts compromised as a result.

In early August 2000, a security newsletter from UC Berkeley reported "several campus Red Hat Linux systems" were compromised with the rpc.statd exploit.
http://www.cs.berkeley.edu/idsg/security/redhat-rpc.statd.shtml

On August 21, 2000 the Computer Incident Advisory Capability (CIAC) released an information bulletin warning of rpc.statd vulnerabilities.
http://ciac.llnl.gov/ciac/bulletins/k-069.shtml

7. Evidence of active targeting:

There is evidence of active targeting. The attacker focused on one host in a class C subnet.

8. Severity:

(criticality + lethality) – (system countermeasures + network countermeasures) = severity

Severity is (1+5) – (5+0) = 1

The ftp server is not critical: 1
The rpc.statd attack is severe on vulnerable hosts and results in a remote root compromise if successful: 5
The OS is hardened and is not running any vulnerable RPC services: 5
There is no firewall. This machine is a "bastion" host and resides in a true DMZ: 0

9. Defensive recommendation:

The server in question is not running any vulnerable RPC services. Nevertheless, the portmapper daemon should be disabled since it is not necessary.

10. Multiple choice test question:

What is the function of the rpc.statd service on UNIX systems?

a) provides performance statistics obtained from the kernel
b) provides system "uptime" information
c) provides a list of current network connections and their status
d) allocates and tracks port numbers assigned to RPC services

The correct answer is a, "provides performance statistics obtained from the kernel".

Detect 4

```
[**] IDS28/probe-nmap_tcp_ping [**]
01/06-12:23:05.591384 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
194.133.58.129:80-> w.x.y.26:53 TCP TTL:50 TOS:0x0 ID:48529 IpLen:20 DgmLen:40
***A**** Seq: 0x3E6 Ack: 0x0 Win: 0x578 TcpLen: 20

[**] IDS7/SourcePortTraffic-53-tcp [**]
01/06-12:23:05.591639 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
194.133.58.129:53-> w.x.y.26:53 TCP TTL:50 TOS:0x0 ID:48530 IpLen:20 DgmLen:40
******S* Seq: 0x1E38D4E1 Ack: 0x0 Win: 0x578 TcpLen: 20
```

1. Source of Trace

My employer's network

2. Detect was generated by:

Snort intrusion detection system, triggered by these rules:

```
alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS28/probe-
nmap_tcp_ping"; ack: 0; flags: A;)

alert TCP $EXTERNAL 53 -> $INTERNAL 0:1023 (msg: "IDS7/SourcePortTraffic-
53-tcp"; flags: S;)
```

3. Probability the source address was spoofed:

There is almost zero probability the source address was spoofed. The traces in the detect above likely indicate a load-balancing network device. The device needs the information returned to determine proximity so it can direct network traffic using the shortest and/or the most efficient path.

4. Description of attack:

The interesting thing about these detects is that they triggered two different Snort rules. This is due to the fact that one packet used the source port of 80 and the other used 53. Also, the ACK flag is set in the first packet whereas the SYN flag is set in the second. Both packets were sent to a publicly accessible DNS server.

They are not likely to be generic DNS lookups because those use the UDP protocol instead of TCP.

5. Attack mechanism:

The packets were definitely crafted, since an acknowledgement number of zero doesn't occur naturally in a TCP header with the ACK flag set.

The source ports are 80 and 53 respectively. Since these are crafted packets, at first look, it appears as though the attacker used these to bypass firewall polices and/or router ACLs. The logic is that some firewalls and screening routers are misconfigured to allow inbound traffic with source ports of 80 (httpd) and 53 (DNS) since these are well-known ports for web services and DNS.

The target host is a nameserver. Most DNS servers run the Berkeley Internet Naming Daemon (BIND). BIND has a long and well-publicized security history. There have been numerous publicly available BIND exploits so probes to discover BIND version information are common.

So, my first theory was a malicious probe for BIND information. However, after I began following a thread on the Snort-users mailing list, I did some more research and determined that these detects are likely the result of a commercial network load balancing device.

6. Correlations:

On February 10, 2001, I began to follow a thread on the Snort-users mailing list called "Strange probes source port 80 dest port 53". Other list subscribers were reporting these probes from the same source IP address as the one I saw, 194.133.58.129, and other addresses. The address in my detect was assigned to a netblock in Europe that I couldn't resolve with nslookup. Since I reached a dead end with my source address, I began to examine some of the other source IPs that were reported.

Other posters speculated that the culprit was a misconfigured load-balancing device, instead of a malicious BIND version query. After I resolved two of the source IPs as “lp1” and “lp2”, I began looking for load-balancing products that might correspond to those hostnames, assuming that network administrators often name devices using a logical scheme, for example, ftp3 or ns1. I checked the websites of F5, Resonate, and Radware. On Radware's site, I discovered that they offer a load-balancing product called LinkProof.

I posted my premise to the Snort-users list on February 11 to see if anyone could corroborate.

```
I got a pair of these probes back on January 6, 2001 to my
employer's DNS server from 194.133.58.129, which I can't resolve.

[**] IDS28/probe-nmap_tcp_ping [**]
01/06-12:23:05.591384 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
194.133.58.129:80-> w.x.y.z:53 TCP TTL:50 TOS:0x0 ID:48529 IpLen:20 DgmLen:40
***A**** Seq: 0x3E6 Ack: 0x0 Win: 0x578 TcpLen: 20

[**] IDS7/SourcePortTraffic-53-tcp [**]
01/06-12:23:05.591639 0:E0:B0:63:DE:2A -> 0:80:5F:0:0:0 type:0x800 len:0x3C
194.133.58.129:53-> w.x.y.z:53 TCP TTL:50 TOS:0x0 ID:48530 IpLen:20 DgmLen:40
******S* Seq: 0x1E38D4E1 Ack: 0x0 Win: 0x578 TcpLen: 20

But two other IPs, reported by Daniel Otis-Vigil, resolve as
follows:

Name:    lp1.sealedair.com
Address:  208.205.199.9
Cryovac (NETBLK-UU-208-205-199) UU-208-205-199 208.205.199.0 -
208.205.199.255

Name:    lp2.sealedair.com
Address:  12.21.190.9
SEALED AIR CORPORATION (NETBLK-ATT1651166626-190) ATT1651166626-
190 12.21.190.0 - 12.21.190.255

This is just a stab in the dark, but could Radware's LinkProof
appliance http://www.radware.com/content/products/link.htm) be
the culprit here?

lp1 and lp2 would seem to be obvious host names for LinkProof
devices.  Moreover, Radware says that LinkProof "ensures the
fastest content delivery for networks with multiple connections
to the Internet."  Cryovac is a division of the Sealed Air
Corporation but they appear to be on separate networks.

Does anyone have access to a LinkProof box to run a test?
```

Although I only have circumstantial evidence, I believe that those paired nameserver probes that many are detecting are generated by Radware LinkProof devices calculating the shortest path back to hosts that visit web servers in their domain.

7. Evidence of active targeting:

There is evidence of active targeting in this case. The two probes were directed at a single nameserver on my employer's network.

8. Severity:

(criticality + lethality) – (system countermeasures + network countermeasures) = severity.

Severity is (5 + 0) – (5 + 0) = 0

The target is a nameserver so it is rather important: 5
The scanner in this case appears to be a load-balancing device, not an attacker gathering information: 0
The OS is hardened: 5
There is no firewall. This server resides in a true DMZ: 0

9. Defensive recommendation:

This is a publicly accessible nameserver, and since DNS uses the well-known port 53 for lookups (UDP) and zone transfers (TCP), firewall policies and restrictive router ACLs will not prevent these scans without hindering legitimate traffic. My recommendation is to continue to monitor vendor security bulletins and apply security patches as necessary.

10. Multiple choice test question:

In the second trace in the detect above, the source and destination ports are the same – 53. This is an example of:

a) port synchronization
b) mirrored ports
c) port matching
d) reflexive ports

The correct answer is d, "reflexive ports".

Assignment 2 - "Analyze This" Scenario (40 Points)

Statement of Work

GIAC Enterprises has contracted Nonspecific Security Systems to analyze network intrusion detection data for evidence of attacks and provide defensive recommendations.

Your organization has provided Nonspecific Security Systems with the following data set:

- Snort alert logs from October 4, 2000 through November 17, 2000, in this format:

```
10/03-04:46:19.742073  [**] WinGate 1080 Attempt [**] 212.158.123.66:3693 ->
MY.NET.218.166:1080
```

- Snort portscan logs from October 4, 2000 through November 19, 2000, in this format:

```
Oct  3 08:41:21 209.92.40.32:9704 -> MY.NET.1.36:9704 SYNFIN **SF****
```

- "Out-of-spec" logs with packet contents from August 17, 2000 through November 11, 2000, in this format:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/17-08:44:01.136063 130.239.11.230:6699 -> MY.NET.181.173:4554
TCP TTL:118 TOS:0x0 ID:19023  DF
**SFR*A* Seq: 0xE4   Ack: 0x1FB61D68   Win: 0x8010
3E 17 80 10 22 20 27 68 00 00 01 01 05 0A 1D 68  >..." 'h.......h
40 67 1D 68                                      @g.h
```

We understand these logs are not complete due to power problems and disk failures.

Summary of Detects

These are the predominant "attacks" evident after a detailed analysis of your Snort logs. Some events are critical and require immediate attention, while others are harmless but irksome "scans" experienced by most organizations with a network connected to the Internet.

Snort Alerts

Signature	# Alerts	# Sources	# Destinations
SYN-FIN scan!	56250	30	25751
Watchlist 000220 IL-ISDNNET-990517	30998	61	108
Watchlist 000222 NET-NCFC	8166	45	26
WinGate 1080 Attempt	4802	570	2655
TCP SMTP Source Port traffic	2893	4	2836
Attempted Sun RPC high port access	2542	20	33
Broadcast Ping to subnet 70	1813	216	1
Back Orifice	1697	40	932
SNMP public access	468	23	1
Null scan!	283	204	196
SMB Name Wildcard	218	33	33
Queso fingerprint	142	29	58
NMAP TCP ping!	96	21	20
SUNRPC highport access!	60	13	12
Connect to 515 from inside	56	2	3
Probable NMAP fingerprint attempt	15	14	13
External RPC call	13	8	3
Tiny Fragments - Possible Hostile Activity	7	5	6
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	7	1	4

site exec - Possible wu-ftpd exploit - GIAC000623	6	4	4
Happy 99 Virus	2	2	2

SYN-FIN scan!

This alert indicates that a probe was detected in which the SYN and FIN flags were set in the TCP header. This combination does not occur naturally in networks and could indicate an operating system fingerprinting attempt.

The operating system on a host is a valuable piece of information to an attacker. Once the OS is known, the vulnerabilities specific to that platform are also known, and the attacker can focus the search for relevant pre-fabricated exploit code to use to attack the target.

Watchlist 000220 IL-ISDNNET-990517

These alerts indicate activity from hosts on the isdn.net.il netblock in Israel. This address block has been placed in a "watch list" due to a history of previous malicious activity.

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

Watchlist 000222 NET-NCFC

These alerts indicate activity from hosts on the Institute of Computing Technology Chinese Academy of Sciences netblock. These addresses have been placed on a "watch list" due to a history of previous malicious activity.

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

Netname: NCFC
Netblock: 159.226.0.0 - 159.226.255.255

WinGate 1080 Attempt

These alerts indicate a scan for a listening SOCKS proxy. These proxies can be valuable for an attacker who wants to conceal his/her identity while attacking another host. The attacker often assumes that the admin of the proxy may not keep detailed logs.

However, these alerts may also be harmless IRC chat servers. Angry users have victimized IRC servers relentlessly so many IRC server hosts have tried to curb attacks by rejecting users connecting from an identity-concealing proxy.

TCP-SMTP Source Port Traffic

This alert indicates someone is attempting to connect to a privileged port (0 – 1023) using the well-known source port 25 for SMTP. This could be an attempt to circumvent router ACLs or firewall policies, since they will sometimes allow all traffic with source port 25 (SMTP).

Attempted Sun RPC high port access

This alert indicates that someone attempted to access a port above 32770. In 1997, there was a vulnerability affecting Sun Microsystems Solaris 2.x hosts wherein rcpbind listened on a port above 32770. This high, non-standard port was not protected in many cases by router access control lists and firewall policies.

Broadcast Ping to Subnet 70

This alert indicates that someone sent a ping to the MY.NET.70.255 broadcast address in hopes that all hosts would respond. This might indicate an attempt at network mapping.

Back Orifice

This alert indicates that someone sent a probe to determine if the Back Orifice trojan is running on the host.

Reference:
CAN-1999-0660

SNMP public access

This alert indicates that someone accessed Simple Network Management Protocol (SNMP) information using the default community string of "public", which is used for authentication. Legitimate network monitoring software can also produce these alerts.

Null scan!

This alert indicates that someone sent a packet in which no flags were set in the TCP header. Since this situation does not occur naturally on networks, we can assume that the packet was crafted and that it is probably malicious.

SMB Name Wildcard

This alert indicates that someone is trying to retrieve the NetBIOS name table on a remote Windows host. Legitimate file and print sharing activities can also produce this alert.

Queso fingerprint

This alert indicates that someone has used the Queso utility for fingerprint the operating system on the target host.

Reference:
CAN-1999-0454

NMAP TCP ping!

This alert indicates that someone sent a ping request using the nmap utility.

Reference:
CAN-1999-0523

SUNRPC highport access!

This alert indicates that someone successfully accessed a port above 32770. There was an old vulnerability affecting Sun Microsystems Solaris 2.x hosts from 1997 wherein rcpbind listened on a port above 32770, usually 32771. Often, this high, non-standard port was not protected by router access control lists and firewall policies.

Connect to 515 from inside

This alert indicates that someone on the internal network connected to port 515. These alerts could be legitimate uses of the lpr or LPRng printing services. But they may also indicate malicious activity since these services have known vulnerabilities.

Reference:
CAN-2000-0917

Probable NMAP fingerprint attempt

This alert indicates that someone used the nmap utility to fingerprint the host's operating system.

Reference:
CAN-1999-0454

External RPC Call

This alert indicates that an external host has attempted to connect to port 111 (RPC portmapper) on a host on the internal network.

Tiny Fragments - Possible Hostile Activity

This alert indicates that someone sent very small, fragmented packets. Since this condition does not naturally occur on networks, it is likely a malicious attempt to evade a firewall or intrusion detection system.

SITE EXEC - Possible wu-ftpd exploit - GIAC000623

This alert indicates that someone launched a remote attack against the wu-2.6.0 ftp daemon.

Reference:

CVE CAN-2000-0574
BugtraqID 1387

Happy 99 virus

This alert indicates that someone sent an e-mail message containing the Happy 99 virus (also known as the W32/Ska worm) through the SMTP gateway.

Reference:
MCAFEE ID 10144

Snort Scans

Signature	# Alerts	# Sources	# Destinations
TCP ***F**** scan	454	28	369
TCP *****P** scan	351	4	349
TCP **S*R*A* scan	281	16	5
TCP ******** scan	226	166	160
TCP 21S***** scan	104	21	38
TCP ***FR*A* scan	60	36	40
TCP *1SF*P** scan	29	10	10
TCP 21SFRPAU scan	28	23	23
TCP 21SF**** scan	16	14	14
TCP *1*F**** scan	16	9	9
TCP *1**RP** scan	14	6	6
TCP **S*R*** scan	14	10	9
TCP 2*SFR*** scan	14	11	11
TCP 2*S***A* scan	14	11	11
TCP *1SFRP** scan	14	12	12
TCP *1SF*PA* scan	13	11	11
TCP **S**P** scan	13	9	8
TCP **SF***U scan	13	5	5
TCP **SFRP*U scan	13	12	10
TCP *1*FRP** scan	13	9	9
TCP 2*SF**A* scan	13	8	8
TCP *1S**P** scan	13	11	11
TCP *1S**P*U scan	12	10	8

TCP 2*S*RPAU scan	12	11	11
TCP *1S***A* scan	12	11	11
TCP 2**FR*A* scan	11	7	10
TCP *1*FR**U scan	11	9	8
TCP 2*****A* scan	11	9	9
TCP 2**FR**U scan	11	10	10
TCP ****RPAU scan	11	3	3
TCP ***F*P*U scan	11	7	7
TCP 21S*R*** scan	11	8	8
TCP *1S**PA* scan	11	10	9
TCP *1S*R**U scan	11	10	10
TCP *1**R*** scan	11	7	7
TCP ***FRPA* scan	10	6	8
TCP ***FR*AU scan	10	8	8
TCP 21***PAU scan	10	7	6
TCP 21*FRPAU scan	10	8	8
TCP 2**FRPA* scan	10	9	9
TCP 2***RPA* scan	10	9	9
TCP 21S**P** scan	10	10	10
TCP 21**RP** scan	10	7	7
TCP 2**FR*AU scan	10	8	8
TCP 21SF*PA* scan	10	5	5
TCP *1S*R*** scan	10	8	8
TCP *1SF*PAU scan	10	10	10
TCP 2**F*P** scan	10	9	9
TCP ***FR*** scan	9	7	7
TCP 2***RP** scan	9	9	9
TCP 21SF*P*U scan	9	8	8
TCP *1S*RP** scan	9	8	8
TCP 21S*R*A* scan	9	7	7
TCP *1S*RPA* scan	9	7	7
TCP 21**R*AU scan	9	8	8
TCP *1*F*P** scan	9	8	8

TCP ****RP** scan	9	7	8
TCP *1****** scan	9	8	7
TCP **SF**A* scan	9	8	8
TCP 2*SF*PA* scan	9	9	8
TCP **S*R**U scan	9	6	7
TCP 21***P*U scan	9	8	8
TCP 21S***A* scan	9	7	6
TCP *1****AU scan	9	6	6
TCP 2***R*AU scan	9	8	8
TCP 2*SF**** scan	8	5	5
TCP **S*RPA* scan	8	7	7
TCP 21****AU scan	8	7	7
TCP 2*SFRPAU scan	8	7	7
TCP 21**R**U scan	8	7	7
TCP 21SFR*A* scan	8	7	7
TCP 21*****U scan	8	6	6
TCP 2*SF*P*U scan	8	7	7
TCP *1*F*PA* scan	8	5	5
TCP *1SF**A* scan	8	6	6
TCP 21*FR**U scan	8	6	6
TCP 21SFRP** scan	8	8	8
TCP *****P*U scan	8	7	7
TCP **SF*P*U scan	8	7	7
TCP ***FRP** scan	8	8	8
TCP 2******* scan	8	6	6
TCP 21SF*P** scan	7	7	7
TCP 21S**P*U scan	7	6	6
TCP *1S*RPAU scan	7	5	5
TCP 2****PAU scan	7	7	7
TCP ***F***U scan	7	6	6
TCP 2*S**PA* scan	7	5	5
TCP 2******U scan	7	7	7
TCP 21S*RP*U scan	7	7	7

TCP 21*FRP** scan	7	6	6
TCP 2*S**PAU scan	7	5	5
TCP 21*F*PAU scan	7	6	6
TCP **SF*PA* scan	7	6	6
TCP **S**PAU scan	7	6	6
TCP **S**P*U scan	7	7	7
TCP 21****A* scan	7	7	7
TCP ***FRPAU scan	7	7	7
TCP 21*F**** scan	7	6	6
TCP 2****PA* scan	7	7	7
TCP *1SFRPAU scan	7	6	6
TCP *******U scan	7	7	7
TCP 21**RPA* scan	7	7	7
TCP 2*S*R**U scan	7	6	6
TCP 2*S*RP** scan	7	5	5
TCP *1*F**A* scan	7	7	7
TCP 2*S****U scan	7	7	7
TCP 21*FR*AU scan	7	7	7
TCP 2*SFR**U scan	7	7	7
TCP **SFR*AU scan	7	7	7
TCP 21S*R**U scan	7	7	7
TCP **S***AU scan	7	5	5
TCP 2*SF***U scan	7	6	6
TCP 21S****U scan	7	5	5
TCP 2*SF*P** scan	6	5	5
TCP 21S**PAU scan	6	5	5
TCP *1S*RP*U scan	6	6	6
TCP ****RP*U scan	6	6	6
TCP **S****U scan	6	5	5
TCP *1S****U scan	6	6	6
TCP 2**F**A* scan	6	4	4
TCP *1SFR**U scan	6	6	6
TCP 2*SF*PAU scan	6	6	6

TCP *1SFR*** scan	6	5	5
TCP 21**R*** scan	6	6	6
TCP 2*S*R*AU scan	6	4	3
TCP 21SFR*** scan	6	6	6
TCP *1**R*AU scan	6	6	6
TCP *1S*R*A* scan	6	6	6
TCP *1SF*P*U scan	6	5	5
TCP 2****P*U scan	6	6	6
TCP *1*F***U scan	6	6	6
TCP 21SF***U scan	6	5	6
TCP *1S***AU scan	6	5	5
TCP 21S*RPAU scan	6	4	4
TCP **SF*P** scan	6	4	4
TCP *1*F*P*U scan	6	6	6
TCP 2**F***U scan	6	6	6
TCP 21***P** scan	6	6	6
TCP 21**RPAU scan	6	4	4
TCP *1SFR*A* scan	6	5	5
TCP 21S*RPA* scan	6	5	5
TCP 21SFR*AU scan	6	5	5
TCP 21SF**AU scan	5	3	3
TCP *1SF**** scan	5	5	5
TCP **S*RPAU scan	5	4	4
TCP 21S**PA* scan	5	5	5
TCP *1*F**AU scan	5	5	5
TCP *1***P** scan	5	4	4
TCP *1S**PAU scan	5	3	3
TCP 21***PA* scan	5	4	4
TCP 2***R*A* scan	5	5	5
TCP 21S*R*AU scan	5	5	5
TCP *1*FRPAU scan	5	5	5
TCP ****R*AU scan	5	5	5
TCP 2**FR*** scan	5	5	5

TCP 2*S***** scan	5	5	5
TCP 2*****AU scan	5	4	4
TCP 2**F**** scan	5	5	5
TCP 2*S*R*A* scan	5	4	4
TCP **SFR*** scan	5	4	4
TCP 21*F*P** scan	5	4	4
TCP 2***R**U scan	5	5	5
TCP *1**R**U scan	5	3	3
TCP **SFRPAU scan	5	4	4
TCP 21*FR*** scan	5	4	4
TCP 2*S***AU scan	5	5	5
TCP 21SFRP*U scan	5	5	5
TCP 21*F**AU scan	4	3	3
TCP *1S***** scan	4	4	4
TCP **SFRPA* scan	4	4	4
TCP *1**RPAU scan	4	4	4
TCP ***FR**U scan	4	4	4
TCP **S*RP*U scan	4	4	4
TCP *1SFR*AU scan	4	4	4
TCP **SF**AU scan	4	4	4
TCP 2*S*R*** scan	4	4	4
TCP *1SF***U scan	4	4	4
TCP *1*FR*A* scan	4	4	4
TCP *1**RP*U scan	4	4	4
TCP *1*FR*AU scan	4	4	4
TCP 2*S*RP*U scan	4	4	4
TCP 21S***AU scan	4	4	4
TCP ***F*P** scan	4	4	4
TCP 2*SFRP*U scan	4	4	4
TCP 2*S**P*U scan	4	4	4
TCP 2**F*PAU scan	4	4	4
TCP *1***PA* scan	4	4	4
TCP 2***R*** scan	4	4	4

TCP 2**FRP*U scan	4	4	4
TCP 21SF**A* scan	4	3	3
TCP 21*F*P*U scan	4	4	4
TCP 2*SFRPA* scan	4	4	4
TCP 21S*RP** scan	4	4	4
TCP 2**FRPAU scan	4	4	4
TCP 21*F*PA* scan	4	4	4
TCP 21*F***U scan	3	3	3
TCP **S*RP** scan	3	3	3
TCP *1**R*A* scan	3	3	3
TCP 2*S**P** scan	3	3	3
TCP *1SF**AU scan	3	3	3
TCP *1*****U scan	3	3	3
TCP *1SFRP*U scan	3	3	3
TCP 2*SFRP** scan	3	3	3
TCP *1*FRP*U scan	3	3	3
TCP *1***P*U scan	3	3	3
TCP 21*F**A* scan	3	3	3
TCP *1*FR*** scan	3	3	3
TCP 21**RP*U scan	3	3	3
TCP *1S*R*AU scan	3	3	3
TCP ****R**U scan	3	3	3
TCP 2*SFR*A* scan	3	3	3
TCP **S*R*AU scan	3	3	3
TCP **SFRP** scan	3	3	3
TCP 2****P** scan	3	3	3
TCP 2**F*P*U scan	3	3	3
TCP 21**R*A* scan	3	3	3
TCP **SFR**U scan	3	2	2
TCP 2***RPAU scan	3	3	3
TCP 2***RP*U scan	3	3	3
TCP **SFR*A* scan	3	3	3
TCP ***FRP*U scan	3	3	3

TCP 2**F*PA* scan	3	3	3
TCP 2*SFR*AU scan	2	2	2
TCP 2**F**AU scan	2	2	2
TCP 21*FRPA* scan	2	2	2
TCP 21****** scan	2	2	2
TCP *1SFRPA* scan	2	2	2
TCP 21SFR**U scan	2	2	2
TCP *1***PAU scan	2	2	2
TCP 21*FR*A* scan	2	2	2
TCP *1****A* scan	2	2	2
TCP *1*F*PAU scan	2	2	2
TCP **S**PA* scan	2	2	2
TCP 21SFRPA* scan	2	2	2
TCP *1*FRPA* scan	2	2	2
TCP 2**FRP** scan	2	2	2
TCP 2*S*RPA* scan	2	2	2
TCP *1**RPA* scan	2	2	2
TCP 2*SF**AU scan	1	1	1
TCP **SF*PAU scan	1	1	1
TCP 21SF*PAU scan	1	1	1
TCP 21*FRP*U scan	1	1	1

These alerts were produced by the Snort port scan preprocessor (spp_portscan). False alerts are common so the preprocessor must be tuned properly. The key to understanding their significance is in the description. In the following example, Snort detected a scan because the TCP header of an IP datagram showed **two reserved bits set** along with the **SYN, FIN, PSH, ACK and URG flags set**. Since these combinations do not occur naturally on networks, we can assume the datagram has likely been crafted and is probably malicious.

```
TCP 21SF*PAU scan
```

Most TCP scans are intended to search for open ports, but some are designed to “fingerprint” a remote operating system. The general strategy behind these TCP scans is to craft an anomalous packet and send it off to a host. Since the host’s operating system designers probably have not accounted for anomalous TCP traffic, the host’s TCP/IP stack will respond in a strange way, and likely respond differently than other operating systems. Therefore, an attacker can compare the response to a database of known responses to determine the operating system on the remote host. nmap and SING are examples of utilities than can fingerprint a remote operating system.

Top Talkers

The addresses in the following table are the “top talkers” identified in the alert data that was provided. Each of these source addresses generated over a thousand Snort alerts!

Source IP Addr	Host Name	Registration Info	Alerts
160.78.49.191	ema.chim.unipr.it	Centro di Calcolo di Ateneo (NET-PARMANET1) Centro di Calcolo di Ateneo Universita` di Parma Viale Delle Scienze 43100 PARMA - ITALIA Netname: PARMANET Netblock: 160.78.0.0 - 160.78.255.255	7199
208.61.4.207	adsl-61-4-207.mia.bellsouth.net	BellSouth.net Inc. (NETBLK-BELLSNET-BLK7) 301 Perimeter Center North Suite 400 Atlanta, GA 30346 US Netname: BELLSNET-BLK7 Netblock: 208.60.0.0 - 208.63.255.255 Maintainer: BELL	6635
159.226.45.3	aphy.iphy.ac.cn	The Computer Network Center Chinese Academy of Sciences (NET-NCFC) P.O. Box 2704-10, Institute of Computing Technology Chinese Academy of Sciences Beijing 100080, China Netname: NCFC Netblock: 159.226.0.0 - 159.226.255.255	6297
212.179.95.5	cable-95005.bezeqint.net	inetnum: 212.179.95.0 - 212.179.99.255 netname: CABLE-XPRMNT descr: Cable-Modem-Experiment country: IL admin-c: NP469-RIPE tech-c: NP469-RIPE status: ASSIGNED PA notify: hostmaster@isdn.net.il changed: hostmaster@isdn.net.il 20000103 source: RIPE	6117
209.92.40.32	dslcv1-32.fast.net	FASTNET(tm)-You Tools Corporation (NETBLK-NETBLK-FAST3) NETBLK-FAST3 209.92.0.0 - 209.92.255.255 FASTNET Corporation (NETBLK-DSL1-FASTNET) DSL1-FASTNET 209.92.40.0 - 209.92.47.255	4967
212.179.27.6	clnt-27006.bezeqint.net	inetnum: 212.179.27.4 - 212.179.27.7 netname: ADI-ASSOCIATION descr: ADI-ASSOCIATION-SERIAL country: IL admin-c: NP469-RIPE tech-c: NP469-RIPE status: ASSIGNED PA notify: hostmaster@isdn.net.il changed: hostmaster@isdn.net.il 20000106 source: RIPE	4011
212.179.79.2	could not resolve	inetnum: 212.179.79.0 - 212.179.79.63 netname: CREOSCITEX descr: CREOSCITEX-SIFRA country: IL admin-c: ZV140-RIPE tech-c: NP469-RIPE	3950

		status: ASSIGNED PA notify: hostmaster@isdn.net.il changed: hostmaster@isdn.net.il 20001109 source: RIPE	
212.179.44.115	bzq-44-115.bezeqint.net	inetnum: 212.179.44.64 - 212.179.44.127 netname: GIVAT-BRENER descr: GIVAT-BRENER-LAN country: IL admin-c: ES4966-RIPE tech-c: NP469-RIPE status: ASSIGNED PA notify: hostmaster@isdn.net.il changed: hostmaster@isdn.net.il 20000501 source: RIPE	3938
63.195.56.20	adsl-63-195-56-20.dsl.snfc21.pacbell.net	Pacific Bell Internet Services,Inc. (NETBLK-PBI-NET-7) PBI-NET-7 63.192.0.0 - 63.207.255.255 ADSL BASIC-rback7-snfc21 (NETBLK-SBCIS990913-39) SBCIS990913-39 63.193.210.0 - 63.193.211.25	3897
130.89.229.48	cal032044.student.utwente.nl	University Twente (NET-UTNET) Postbox 217 7500 AE Enschede NETHERLANDS Netname: UTNET Netblock: 130.89.0.0 - 130.89.255.255	3860
210.113.89.200	could not resolve	inetnum 210.113.0.0 - 210.113.255.255 netname KORNET descr Korea Telecom descr 100 Sejong-no Chongno-gu Seoul, Korea descr 110-777 country KR admin-c GC1-AP, inverse tech-c JK14-AP, inverse remarks ISP in Korea changed hostmast@rs.krnic.net 980707 source APNIC	3572
203.32.161.197	adnet.imgserv.com	inetnum 203.0.0.0 - 203.63.255.255 netname AUNIC-AU descr Australian Network Information Center descr 5/490 Northbourne Av. descr Dickson descr ACT 2602 country AU admin-c GH105, inverse tech-c GH105, inverse remarks AUNIC - AUstralian Network Information Centre notify register@aunic.net, inverse mnt-by MAINT-APNIC-AP, inverse mnt-lower MAINT-APNIC-AP, inverse changed dbmon@apnic.net 19990914 source APNIC	3545
213.41.69.52	hosting-52.69.rev.fr.colt.net	inetnum: 213.41.69.0 - 213.41.69.255 netname: FR-COLT-FRANCE-BESS-SHAREDHOSTING5 descr: FR-COLT-FRANCE-BESS-SHAREDHOSTING5 country: FR admin-c: PM876-RIPE tech-c: TT997-RIPE status: ASSIGNED PA	3399

		mnt-by: COLT-FR-MNT mnt-lower: COLT-FR-MNT changed: tarik.tifour@fr.colt.net 20000622 source: RIPE	
193.64.114.10	net10.printeq.fi	inetnum: 193.64.114.0 - 193.64.114.127 netname: PRINTEQ-FI-1 descr: Chemitalic-Printeq Oy descr: Joensuunkatu 13, FI-24100 Salo country: FI admin-c: MP5537-RIPE tech-c: MP5537-RIPE status: ASSIGNED PA notify: hostmaster@kpnqwest.fi mnt-by: KQFI-NOC-MNT changed: hostmaster@kpnqwest.fi 20001023 source: RIPE	3295
195.103.69.159	proxy.guest.net	inetnum: 195.103.69.128 - 195.103.69.191 netname: TOPSOFT-NET descr: Top Software sa descr: Internet service provider country: IT admin-c: AG527-RIPE tech-c: RP367-RIPE status: ASSIGNED PA notify: network@cgi.interbusiness.it mnt-by: INTERB-MNT changed: cgiadmin@cgi.interbusiness.it 19970509 source: RIPE	3292
210.101.101.110	could not resolve	inetnum 210.101.64.0 - 210.101.127.255 netname KORNET descr Korea Telecom descr 100 Sejong-no Chongno-gu Seoul, Korea descr 110-777 country KR admin-c GC1-AP, inversetech-c JK14-AP, inverse remarks ISP in Korea changed hostmast@rs.krnic.net 980707 source APNIC	2582
212.0.107.107	could not resolve	inetnum: 212.0.107.96 - 212.0.107.111 netname: TELSON descr: TELSON country: ES admin-c: JRA22-RIPE tech-c: JMC61-RIPE tech-c: JBGP4-RIPE status: ASSIGNED PA notify: fernovo@telson.es changed: jmacia@colt-telecom.es 20000301 source: RIPE	2338
63.193.210.208	adsl-63-193-210-208.dsl.snfc21.pacbell.net	Pacific Bell Internet Services,Inc. (NETBLK-PBI-NET-7) PBI-NET-7 63.192.0.0 - 63.207.255.255 ADSL BASIC-rback7-snfc21 (NETBLK-SBCIS990913-39) SBCIS990913-39 63.193.210.0 - 63.193.211.255	1883
211.46.110.81	could not resolve	inetnum 211.42.0.0 - 211.51.255.255 netname KRNIC-KR-23 descr KRNIC descr Korea Network Information Center country KR	1789

		admin-c WK1-AP, inverse tech-c SL119-AP, inverse remarks KRNIC Allocation Block mnt-by APNIC-HM, inverse mnt-lower MNT-KRNIC-AP, inverse changed hostmaster@apnic.net 19991118 source APNIC	
212.179.72.226	could not resolve	inetnum: 212.179.72.224 - 212.179.72.239 netname: KESHET descr: KESHET-LAN country: IL admin-c: ES4966-RIPE tech-c: NP469-RIPE status: ASSIGNED PA notify: hostmaster@isdn.net.il changed: hostmaster@isdn.net.il 20000320 source: RIPE	1591
143.89.13.3	ustlnx6.ust.hk	Hong Kong University of Science & Technology (NET-USTHK) Clear Water Bay Road Sai Kung HONG KONG Netname: USTHK-NET Netblock: 143.89.0.0 - 143.89.255.255	1584
128.2.81.133	8TH-DWARF.REM.CMU.EDU	Carnegie-Mellon University (NET-CMU-NET) 5000 Forbes Avenue Pittsburgh, PA 15213 US Netname: CMU-NET Netblock: 128.2.0.0 - 128.2.255.255	1569
63.167.58.13	could not resolve	Sprint (NETBLK-SPRN-BLKS) PRN-BLKS 63.160.0.0 - 63.175.255.255 CELLUAR RENTALS DBA (NETBLK-FON-106792396847600) FON-106792396847600 63.167.58.0 - 63.167.58.127	1531
212.179.41.24	fr-c41024.bezeqint.net	inetnum: 212.179.41.0 - 212.179.41.63 netname: YTV-VILLEGE descr: YTV-villege-LAN country: IL admin-c: TP1233-RIPE tech-c: NP469-RIPE status: ASSIGNED PA notify: hostmaster@isdn.net.il changed: hostmaster@isdn.net.il 20000109 source: RIPE	1353
159.226.91.20	could not resolve	The Computer Network Center Chinese Academy of Sciences (NET-NCFC) P.O. Box 2704-10, Institute of Computing Technology Chinese Academy of Sciences Beijing 100080, China Netname: NCFC Netblock: 159.226.0.0 - 159.226.255.255	1212
163.10.19.34	Decanato.exactas.unlp.edu.ar	Universidad Nacional de La Plata (NET-REDUNLP) 50 y 115, 3er piso CeSPI, UNLP 1900 La Plata ARGENTINA Netname: REDUNLP Netblock: 163.10.0.0 - 163.10.255.255 Coordinator: Cozzi, Rodolfo (RC202-ARIN) rcozzi@NETVERK.COM.AR +54 21 24 1049 (FAX) +54 21 24 1049	1105

24.7.227.215	could not resolve	@Home Network (NETBLK-ATHOME) ATHOME 24.0.0.0 - 24.23.255.255 @Home Network (NETBLK-BB1-RDC2-TX-2) BB1-RDC2-TX-2 24.7.224.0 - 24.7.239.255	1096
212.187.21.156	c21156.upc-c.chello.nl	inetnum: 212.187.20.0 - 212.187.23.255 netname: TK-APL-2 descr: Telekabel Apeldoor ndescr: cablemodem block 2 country: NL admin-c: WD294-RIPE tech-c: CBHM-RIPE status: ASSIGNED PA notify: ripemaster@chello.com mnt-by: AS9141-MNT changed: ripemaster@chello.com 19990610 source: RIPE	1085

Significant Findings

From the Watchlist 000220 IL-ISDNNET-990517 alerts:

The host cable-95005.bezeqint.net (212.179.95.5) is responsible for generating 6177 Snort alerts on your network. This link graph summarizes the traffic between this host and hosts on your network.

None of the source or destination ports are well-known or even associated with a known service. This traffic is extremely anomalous. Without further correlations we can't identify what this attacker was looking for.

From the Watchlist 000222 NET-NCFC alerts:

Hosts from the NCFC netblock seem to be probing your network for mail servers and UNIX-like servers running the identd authentication daemon almost exclusively. The target ports in their scans are nearly always 25 and 113. It difficult to say what they were after without corroboration from system logs or more detailed Snort alert logs.

Sendmail is notorious for its long security history. There are no current vulnerabilities affecting the identd service.

From the WinGate 1080 Attempt alerts:

MY.NET. 206.118 and MY.NET.255.154 may be misconfigured SOCKS proxies.

These hosts should be investigated immediately to determine if they have been compromised or if their SOCKS proxies are just improperly secured.

At any rate, traffic to port 1080 on these hosts is high and the patterns indicate much more than simple port scans. The detect below has the look of web browser traffic, i.e., the multiple high source ports on 212.72.75.236.

```
11/19-05:57:18.433007 [**] WinGate 1080 Attempt [**] 212.72.75.236:3067-> MY.NET.206.118:1080
11/19-05:57:18.592016 [**] WinGate 1080 Attempt [**] 212.72.75.236:3073-> MY.NET.206.118:1080
11/19-05:57:18.592072 [**] WinGate 1080 Attempt [**] 212.72.75.236:3074-> MY.NET.206.118:1080
11/19-05:57:19.111252 [**] WinGate 1080 Attempt [**] 212.72.75.236:3087-> MY.NET.206.118:1080
11/19-05:57:19.299453 [**] WinGate 1080 Attempt [**] 212.72.75.236:3094-> MY.NET.206.118:1080
11/19-05:57:19.690656 [**] WinGate 1080 Attempt [**] 212.72.75.236:3101-> MY.NET.206.118:1080
11/19-05:57:20.191309 [**] WinGate 1080 Attempt [**] 212.72.75.236:3111-> MY.NET.206.118:1080
11/19-05:57:20.627951 [**] WinGate 1080 Attempt [**] 212.72.75.236:3114-> MY.NET.206.118:1080
11/19-05:57:20.790840 [**] WinGate 1080 Attempt [**] 212.72.75.236:3118-> MY.NET.206.118:1080
```

From the SUNRPC highport access! alerts:

These hosts have had successful connections to port 32771. It is likely they are Sun Solaris workstations running rpcbind on that port. This requires further investigation.

MY.NET.206.222
MY.NET.202.242
MY.NET.212.186
MY.NET.228.62
MY.NET.97.59
MY.NET.53.23
MY.NET.253.114
MY.NET.53.14
MY.NET.140.51
MY.NET.6.15
MY.NET.179.78
MY.NET.206.218

From the Broadcast Ping to subnet 70 alerts:

216 separate external hosts have sent a ping to the broadcast address on the MY.NET.70.0 subnet, MY.NET.70.255. This is likely an attempt at network mapping. Only hosts that are alive on the network will respond, giving the attacker an inventory of target hosts.

From the Back Orifice alerts:

MY.NET.97.208 may be infected with the BackOrifice trojan. There is a lot of activity from different hosts directed at port 31337 on this machine. This requires further investigation.

From the SNMP public access alerts:

23 hosts on your internal network are accessing SNMP information on MY.NET.101.92. If this machine is running network-monitoring software then this traffic may be legitimate. This requires further investigation.

Correlation:
Alan Powell, from SANS Security DC 2000, also believes this traffic is legitimate network monitoring activity:

```
"[**] SNMP public access [**]

The source address for all of the events triggering this is MY.NET.97 and the
destination address is always on  MY.NET.101. Maybe someone setting up ucd-snmp on a
linux box?"
```

From the Connect to 515 from inside alerts:

This indicates a possible compromise. Why is a host on your network using print services of a machine on the Internet? Red Hat Linux 6.x (lpr) and Red Hat Linux 7.0 (LPRng) both have vulnerable services that run on port 515 for which there are publicly available exploits.

It seems more likely that the MY.NET.179.78 host has been compromised and is being used to scan other hosts or that its owner is involved in nefarious activity.

```
11/22-11:24:06.406682 [**] connect to 515 from inside [**]
MY.NET.179.78:2274-> 64.244.202.110:515
11/22-11:33:56.296324 [**] connect to 515 from inside [**]
MY.NET.179.78:2707-> 64.244.202.66:515
```

Correlation:
Ken Wellmaker, from SANS Security DC 2000, also reported that MY.NET.179.78 has been compromised:

```
"Compromised hosts

This table of data shows the attackers that have successfully connected to port 32771
(SUNRPC).  This indicates these hosts have been compromised.  In addition,
MY.NET.253.12 is listed as a source indicating that he has been compromised as well.
MY.NET.253.12 has triggered 4,225 SUNRPC highport alerts indicating he may have
compromised other hosts on the 101, 102, 16 and 19 networks.  In addition to this
activity, there are 6,849 "attempted" alerts in the logs from various attackers.

Alert Date    Message      Origin SP     Destination   DP
[...]
```

```
06/12-21:42:56.272373      [**] SUNRPC highport access! [**] 24.13.123.8   3708
      MY.NET.179.78 32771
06/13-18:22:46.343232      [**] SUNRPC highport access! [**] 24.18.90.197  2468
      MY.NET.179.78 32771
[...]"
```

From the OOSCheck logs:

99% of the network traces in the OOS files indicate routine ports scans by attacker(s) using the *synscan* tool created by psychoid. We know this because the packet ID from every trace is 39426, a signature of that particular tool. Also, with a few exceptions, the source port is the same as the destination port.

```
11/11-04:58:35.455546 211.46.110.81:4 -> 10.0.232.165:23
TCP TTL:23 TOS:0x0 ID:39426
**SF**** Seq: 0x503F864   Ack: 0x5CDD9853   Win: 0x404
00 00 00 00 00 00                                ......
```

This summary table of source and destination ports from the *synscan* portscans illustrate what the attacker was looking for in each case.

Src		Dst	Service
27374	→	27374	Sub Seven trojan
9704	→	9704	back door root shell left in wake of rpc.statd compromise
109	→	109	POP2 post-office protocol 2
21	→	21	FTP file transfer protocol
53	→	53	DNS domain name service
4	→	23	Telnet

Defensive recommendations

Your network has been the target of many thousands of port scans. It is obvious that your organization has not employed a firewall or even packet filtering on your border router since these scans have penetrated into your internal network.

Therefore, our primary recommendation is that you implement a firewall or configure restrictive access control lists on your border router as soon as possible.

We suggest a "deny by default" policy. That is, you should deny all network traffic by default and then selectively allow certain types of necessary traffic, such as web, mail and DNS services.

To provide the most flexibility to your users, you can take advantage of the stateful aspect of most firewalls and the "established" keyword in router access lists. This will allow your users to use most internet services by initiating the connection from inside your network. Inbound traffic that is part of these established connections can be safely allowed back into your network.

If this is done correctly then nearly 100% of the future port scans directed at your organization will be dropped at the edge of your network.

We recommend that you perform proactive port scans of the hosts on your network. This will allow you to promptly discover if a host has been trojaned or is running unauthorized software so you can address it

fast. Generally, it is a poor strategy to wait to investigate until you detect successful connections to well-known trojan ports with your intrusion detection sensor.

Your network has also received a large amount of network traffic orginating from two known bad netblocks. In addition, you now have a concise list of "top talkers" responsible for much of the malicious traffic.

At the very least, your administrators should block the IP addresses in the "top talkers" list at your border router with access control lists or at your firewall through policies. In many cases, these IP addresses represent **compromised hosts that will continue to be used to probe and attack your network**.

However, we recommend a more aggressive measure: block access to your network from ANY hosts on these networks. Since these networks are the hangouts for known troublemakers, you gain very little by allowing communication between their network and yours.

Even if the SNMP activity detected in the Snort alert logs is legitimate network monitoring activity, your administrators are using the default SNMP community string – PUBLIC – for authentication. This string can easily be changed to something more complex and harder to guess. Follow "strong password" guidelines and set a new community string.

The W32/Ska worm was detected twice moving through your mail gateway. Consider installing antivirus filter software on your mail gateway or configure sendmail to block dangerous attachments. Of course, you should also install antivirus software on Windows desktop and laptop machines and caution users against opening e-mail attachments. Since viruses outbreaks can be so costly to quash, it is advisable to employ layers of protection to ensure that they don't occur.

Closing

Most of the malicious activity targeting your internal network is from the outside. And much of that traffic is from addresses in China and Israel already known for questionable activities.

You have a large number of hosts on your network that do not seem to be adhering to any restrictions on network services (evidence of external connections to services running on odd ports are numerous). You also have many users using applications that may not be essential for business, i.e., internet relay chat.

By implementing firewall and/or packet filtering, and by proactively auditing your internal networks for unauthorized services and software, you can address most of these problems.

Your network will be more secure and more efficient as a result.

Assignment 3 - Analysis Process (30 Points)

Requirement:
A list of detects, prioritized either by severity or number of occurrences, and a brief description of these.

Like many previous students, I chose to use SnortSnarf (http://www.silicondefense.com/snortsnarf) to organize and present the data contained in the Snort alert and Snort scan files. Working with the files manually, even using UNIX tools such as *sort* and *grep*, would have been overwhelming.

I normally work with Windows NT or Windows 2000. However, for data manipulation, I find that UNIX has more flexible tools. For this assignment, I used *sed*, *awk*, *grep*, *sort* and *wc*.

On Windows 2000 Professional, I used ActivePerl (http://www.activestate.com), Internet Explorer 5, Microsoft Word 2000 and *nslookup*.

In UNIX, I downloaded the files and used *unzip* to decompress them.

```
unzip snortA.zip
unzip snortS.zip
unzip OOS.zip
```

I concatenated all the extracted files with *cat* into one big file for processing by SnortSnarf.

```
cat SnortA* > /usr/snortALERT
cat SnortS* > /usr/snortSCAN
cat OOSc* > /usr/OOSCHECK
```

Then I ran SnortSnarf:

```
./snortsnarf.pl snortALERT2 snortSCAN2
```

I can attest to the author's caution about SnortSnarf and RAM. On a 200Mhz processor with 32 MB of RAM, the snortsnarf script failed and produced "out of memory" errors.

I had better success on a machine with a 700Mhz PIII processor and 256 MB of RAM.

The SnortSnarf reports eventually finished but some of the links were broken. I realized that this was likely due to the MY.NET obfuscation of the internal addresses.

I used the UNIX tool *sed* to replace every instance of MY.NET with 10.0.

```
sed -e 's/MY.NET/10.0/' snortALERT > snortALERT1
sed -e 's/MY.NET/10.0/' snortSCAN > snortSCAN1
sed -e 's/MY.NET/10.0/' OOSCHECK > OOSCHECK1
```

I wasn't going to process the OOSCHECK file with SnortSnarf but I decided I would replace the MY.NET references anyway for consistency.

I had to run these files through *sed* twice because it didn't catch every instance of MY.NET on the first run. I suspect this is because MY.NET occurred twice on some lines where it prefixed both the source and destination addresses and *sed* only substituted the first instance. I'll make it a point to learn more about *sed* since I'll likely use it again in the future.

```
sed -e 's/MY.NET/10.0/' snortALERT1 > snortALERT2
sed -e 's/MY.NET/10.0/' snortSCAN1 > snortSCAN2
sed -e 's/MY.NET/10.0/' OOSCHECK1 > OOSCHECK2
```

I now had two files properly prepped for SnortSnarf -- snortALERT2 and snortSCAN2.

I ran SnortSnarf again:

```
perl snortsnarf.pl snortALERT2 snortSCAN2
```

After a few hours, the script finished.

The index.html file created by SnortSnarf contains a table with this heading:

Signature	# Alerts	# Sources	# Destinations

This is perfect for the "list of detects" requirement. I split the alerts into one table and the scans into another by cutting and pasting them into Word 2000.

One minor problem remained. SnortSnarf automatically sorts by number of alerts in ascending order. I wanted my summary lists to be sorted in descending order, starting from most alerts, since large numbers of detects usually (but not always) demand attention first.

To accomplish this, I used Word's Table → Sort feature. Highlight the second column (#Alerts) and choose Sort By: Column 2 and Type: number and click the Descending radio button.

Analyzing the OOSCheck files

The assignment instructions cautioned against ignoring the OOSCheck files. As I began to browse through the concatenated OOSHCHECK2 file I created, I noticed that the packet ID for many of the traces was 39426.

I knew from research that a packet ID of 39426 was a signature of the *synscan* port scanner created by psychoid.

I used the UNIX tools *grep* and *wc* to find out how much of the OOSCHECK file were examples of port scans conducted by the *synscan* tool.

```
bigeye# grep ID: OOSCHECK2 | wc -l
   63398
bigeye# grep ID:39426 OOSCHECK2 | wc -l
   62590
```

62590 / 63398 = .987 or 99% of the OOSCheck files are *synscan* port scans.

I also used *grep* to get an idea of the source and destination ports of the traces contained in the OOSCHECK2 file. This gave me at least a superficial view of what services the attacker(s) were looking for.

I typed `grep "ID:39426" -B 1 OOSCHECK2` and watched the source and destination ports as the output scrolled on my screen. grep's B argument allows you to specify a certain number of lines to be printed before the string you specify.

```
--
10/14-21:00:22.936337 130.89.229.48:53 -> 10.0.68.56:53
TCP TTL:32 TOS:0x0 ID:39426
--
10/14-21:00:23.022289 130.89.229.48:53 -> 10.0.68.60:53
TCP TTL:32 TOS:0x0 ID:39426
[…]
```

Requirement:
A "top talkers" list

For the "top talkers" list, I arbitrarily extracted source addresses responsible for 1000 alerts or more from the SnortSnarf report using Edit → Cut in Internet Explorer. I chose 1000 since this amount of activity seemed excessive and I had to set the "top talkers" level somewhere.

For these addresses, I used my lookup list to resolve hostnames and used the Arin, Ripe and Apnic whois databases to find registration information.

http://www.ripe.net/cgi-bin/whois
http://www.arin.net/cgi-bin/whois.pl
http://www.apnic.net/apnic-bin/whois.pl

Requirement:
A list of source addresses and registration information about these.

I tried to use the UNIX *awk* utility to extract the source addresses from the snortALERT and snortSCAN files.

I planned to pipe the output through *uniq* to get a list of all source addresses.

One use of *awk* allows you to treat a line of data as fields or columns, using spaces as the delimiter.

I tried this command:

```
awk '{ print $6$7 }' snortALERT2
```

But it produced mixed results:

```
203.155.129.50:31338->
203.155.129.50:31338->
from4.40.0.91:
from4.40.0.91:
from4.40.0.91:
from4.40.0.91:
portscanfrom
from4.40.0.91
from4.40.0.91:
from4.40.0.91:
from4.40.0.91:
```

Although this printed out most of the source IP addresses, the field they inhabited in the alert file was not consistent from line to line. In many cases, the seventh field was "from", "port" or "->". I was not confident this would give me an accurate accounting of the source IP addresses.

Finally, I decided to cut and paste the source addresses from the SnortSnarf reports in Internet Explorer. It took longer but it was more accurate.

I ended up with a list like this, which included extraneous info from the SnortSnarf report:

```
210.101.101.110 2582 5160 2582 2582
212.0.107.107 2338 4678 2338 2344
143.89.13.3 1584 3171 1584 1587
128.2.81.133 1569 1569 1569 1569
63.167.58.13 1531 3077 1531 1546
163.10.19.34 1105 3315 1105 1105
212.187.21.156 1085 2167 1085 1085
```

Now I could use *awk* to print only the first field, which is the source IP address.

```
awk '{ print $1 }' sourceaddr > sourceaddr1
```

I wrote a quick and dirty script in Perl to use *nslookup* to resolve host names to IP addresses.

```
@hosts = (
[list of source IP addresses, surrounded by quotes and comma-delimited]
);
foreach $host (@hosts) {
$x = `nslookup $host`;
print $x;
}
```

But before I could populate the @hosts array with the list of source IP addresses from the Snort alert and scan data, I had to treat the IP addresses to surround them with quotation marks and separate them by commas, or the script wouldn't work.

I pasted the source IP addresses into a separate file called external.

I used *awk* to add the quotes and commas

```
awk '{print "\""$0"\""\,"}' external > external1
```

This command turned this:

```
24.66.231.148
24.26.55.231
24.65.145.247
24.214.18.65
24.72.12.211
24.129.82.105
24.3.52.236
24.229.67.16
24.129.82.132
24.40.46.225
24.200.80.101
24.214.77.118
24.27.222.61
24.218.41.5
[…]
```

into this:

```
"24.66.231.148", "24.26.55.231", "24.65.145.247", "24.214.18.65",
"24.72.12.211", "24.129.82.105", "24.3.52.236", "24.229.67.16",
"24.129.82.132", "24.40.46.225", "24.200.80.101", "24.214.77.118",
"24.27.222.61", "24.218.41.5"
[…]
```

And now I could paste it into the array in the lookup.pl script and run it.

```
C:\WINNT\Profiles\Administrator\Desktop>perl lookup.pl > lookuplist.txt
```

It produced mixed results because there were many addresses that couldn't be resolved:

```
*** se1.res.dns.psi.net can't find 212.0.107.107: Non-existent domain
*** se1.res.dns.psi.net can't find 211.46.110.81: Non-existent domain
```

```
*** se1.res.dns.psi.net can't find 211.46.110.81: Non-existent domain
*** se1.res.dns.psi.net can't find 211.46.110.81: Non-existent domain
*** se1.res.dns.psi.net can't find 211.46.110.81: Non-existent domain
[…]
```

But it did create a nice list of hostnames for the IP addresses that could be resolved:

```
Name:    net104-226.jal.cc.il.us
Address:  4.19.104.226

Name:    lsanca1-ar8-184-141.dsl.gtei.net
Address:  4.35.184.141
[…]
```

I saw little value in including the entire list in my practical exercise but I kept it on hand to lookup hostnames when necessary using Word's Edit → Find utility.

Requirement:
Correlations from previous students' practicals (numbers 0209 and above).

This was fairly easy. I downloaded the student practicals numbered 0209 and above into a folder on my hard drive. When I needed to find previously reported activity for a particular IP address, I used Explorer's Find feature. Start → Search → For Files and Folders → and entered the IP address in question into the Containing Text field.

Requirement:
At least one link graph

I used Microsoft Visio to draw a link graph to describe network detects that I otherwise could make no sense out of. The visualization of the traffic helped me make more educated guesses, especially with the "Watchlist" detects, since the source and destination ports were so odd I couldn't decipher what the scanners were looking for.

Requirement:
Any insights into internal machines, such as compromise or possible dangerous or anomalous activity.

For this requirement I used many sources of information. My main source was the Google search engine (http://www.google.com). I typically tried this first when I encountered something I didn't know. For information on vulnerabilities and fixes, I used the vulnerability database hosted by SecurityFocus (http://www.securityfocus.com). It allows searches by Bugtraq ID or CVE ID. I also used Max Vision's arachNIDS database for Snort rule descriptions and quick one-paragraph descriptions of exploits. Finally, the port search database at http://www.snort.org is the quickest way I know to look up a port number, especially since links to it have been incorporated into the SnortSnarf reports.

Requirement:
Defensive recommendations.

For this requirement, I used my own experience and the vulnerability database at http://www.securityfocus.com

Requirement:

A summary analysis of the aggregate data.

For this requirement, I just tried to condense the major findings into a couple short paragraphs.