Global Information Assurance Certification Paper

SANS GIAC Training Practical assignment version 2.8

Dean White

SANS Darling Harbor 2000 GCIA practical assignment
Featuring an introductory paper on IDS Flexible response Systems

Table of Contents:

Assignment #1 - Analysis of 5 detects:

Detect #1

Smurf Scan
1) Source of detect
The source of this detect is a perimeter intrusion detection system running on the targeted network.

2) Detect generated by
This detect was generated by a Snort version 1.7 OpenBSD box running a modified snort ruleset from http://www.snort.org and appeared in the SnortSnarf output generated every 10 minutes on this site.

3) Probability of spoofed source address
The probably of the attacker spoofing the source address in this case is low. This appears to be a network recon looking for sub-nets which can be used as smurf amplifiers. If the attacker were to spoof their source address they would get no information back from the recon.

4) Description of attack
The attacker is scanning both broadcast and network addresses looking for networks which can be used as Smurf amplifiers for a DoS attack.

What is interesting about these ICMP packets are they are all crafted. Notice that in

each ICMP packet the ID remains the same. Normally this should have incremented by one with each ICMP packet sent. Static fields like this are generally the keys to being able to write signatures to detect this kind of traffic.

Snort Alert data:

```
[**] <ts0> ICMP Broadscan Smurf Scanner [**]
05/07-17:32:38.110000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x2E
62.98.184.147 -> a.b.c.0 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20 DgmLen:32
Type:8 Code:0 ID:0 Seq:0 ECHO


[**] <ts0> ICMP Broadscan Smurf Scanner [**]
05/07-17:32:38.140000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x2E
62.98.184.147 -> a.b.c.8 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20 DgmLen:32
Type:8 Code:0 ID:0 Seq:0 ECHO

[**] <ts0> ICMP Broadscan Smurf Scanner [**]
05/07-17:32:38.140000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x2E
62.98.184.147 -> a.b.c.63 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20 DgmLen:32
Type:8 Code:0 ID:0 Seq:0 ECHO

[**] <ts0> ICMP Broadscan Smurf Scanner [**]
05/07-17:32:38.160000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x3C
62.98.184.147 -> a.b.c.64 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20 DgmLen:32
Type:8 Code:0 ID:0 Seq:0 ECHO

[**] <ts0> ICMP Broadscan Smurf Scanner [**]
05/07-17:32:38.180000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x3C
62.98.184.147 -> a.b.c.128 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20
DgmLen:32Type:8 Code:0 ID:0 Seq:0 ECHO

[**] <ts0> ICMP Broadscan Smurf Scanner [**]
05/07-17:32:38.190000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x2E
62.98.184.147 -> a.b.c.127 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20
DgmLen:32Type:8 Code:0 ID:0 Seq:0 ECHO

[**] <ts0> ICMP Broadscan Smurf Scan ner [**]
05/07-17:32:38.200000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x3C
62.98.184.147 -> a.b.c.191 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20
DgmLen:32Type:8 Code:0 ID:0 Seq:0 ECHO

[**] <ts0> ICMP Broadscan Smurf Scanner [**]
05/07-17:32:38.250000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x3C
62.98.184.147 -> a.b.c.192 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20
DgmLen:32Type:8 Code:0 ID:0 Seq:0 ECHO

[**] <ts0> ICMP Broadscan Smurf Scanner [**]
05/07-17:32:38.250000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type :0x800 len:0x2E
62.98.184.147 -> a.b.c.254 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20
DgmLen:32Type:8 Code:0 ID:0 Seq:0 ECHO
```

[**] <ts0> ICMP Broadscan Smurf Scanner [**]
05/07-17:32:38.290000 0:0:0:0:0:0 -> 0:0:0:0:0:0 type:0x800 len:0x2E
62.98.184.147 -> a.b.c.255 ICMP TTL:236 TOS:0x0 ID:2149 IpLen:20
DgmLen:32Type:8 Code:0 ID:0 Seq:0 ECHO

TCPDUMP data:
17:32:38.110000 62.98.184.147 > a.b.c.0: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 1514 3e62 b893 E....e.....>b..
0x0010 cb6d ef00 0800 f7ff 0000 0000 cb6d ef00 .m.........m..

17:32:38.140000 62.98.184.147 > a.b.c.8: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 150c 3e62 b893 E....e .....>b..
0x0010 cb6d ef08 0800 f7ff 0000 0000 cb6d ef08 .m.........m..

17:32:38.140000 62.98.184.147 > a.b.c.63: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 14d5 3e62 b893 E....e.....>b..
0x0010 cb6d ef3f 0800 f7ff 0000 0000 cb6d ef3f .m.?.........m.?

17:32:38.160000 62.98.184.147 > a.b.c.64: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 14d4 3e62 b893 E....e.....>b..
0x0010 cb 6d ef40 0800 f7ff 0000 0000 cb6d ef40 .m.@.........m.@
0x0020 0000 0000 0000 0000 0000 0000 0000 ....

17:32:38.180000 62.98.184.147 > a.b.c.128: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 1494 3e62 b893 E....e.....>b..
0x0010 cb6d ef80 0800 f7ff 0000 0000 cb6d ef80 .m.........m..
0x0020 0000 0000 0000 0000 0000 0000 0000 ....

17:32:38.190000 62.98.184.147 > a.b.c.127: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 1495 3e62 b893 E....e.....>b..
0x0010 cb6d ef7f 0800 f7ff 0000 0000 cb6d ef7f .m.........m..

17:32:38.200000 62.98.184.147 > a.b.c.191: icmp: echo request (wrong ic mp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 1455 3e62 b893 E....e.....U>b..
0x0010 cb6d efbf 0800 f7ff 0000 0000 cb6d efbf .m.........m..
0x0020 0000 0000 0000 0000 0000 0000 0000 ....

17:32:38.250000 62.98.184.147 > a.b.c.192: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 1454 3e62 b893 E....e.....T>b..
0x0010 cb6d efc0 0800 f7ff 0000 0000 cb6d efc0 .m.........m..
0x0020 0000 0000 00 00 0000 0000 0000 0000 ....

17:32:38.250000 62.98.184.147 > a.b.c.254: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 1416 3e62 b893 E....e.....>b..
0x0010 cb6d effe 0800 f7ff 0000 0000 cb6d effe .m.....m..

17:32:38.290000 62.98.184.147 > a.b.c.255: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 1415 3e62 b893 E....e.....>b..
0x0010 cb6d efff 0800 f7ff 0000 0000 cb6d efff .m.....m..

Packets with wrong icmp checksums normally get thrown away by the hosts they are destined for. What we are seeing is the traffic of a tool that doesn't craft packet checksums correctly.

This traffic has been created with the broadscan smurf scanner tool, an application which scans for networks that can be used as smurf amplifiers. When it finds a network that responds, it records it in a text file.

5) Attack Mechanism
The aim is to find networks with machines that use the BSD TCP/IP stack, which respond to this type of broadcast. When an ICMP echo is sent to a network or broadcast address the router for the sub-net will expand the ICMP echo and send it to all hosts on the sub-net. They the respond with an ICMP echo-reply to the source. In this way, one incoming ICMP echo can generate many ICMP echo-responds to the spoofed source address.

6) Correlation's
I would have believed this detect would have featured regularly on some of the incident correlation sites. I have been unable to find any reference to the broadscan smurf scanner. I have seen this event happen at least 2-3 a week for the past few months.

7) Evidence of active targeting
There is no evidence to suggest active targeting in this case. This is a recon probe, searching sub-net network and broadcast addresses, watching for responses.

8) Severity
Criticality = 5 (This scan is launched against all systems on this network)
Lethality = 4 (This has the potential to allow this network to DoS others)
System Countermeasures = 5 (This machine has been patched and does not answer broadcast pings)
Network Countermeasures = 5 (This firewall completely blocks ICMP ping)

(5+4) – (5+5) = -1

9) Defensive recommendation
This network because of correct filtering at the firewall did not return any responses to this stimulus.

The defensive recommendation to prevent your site from being used as a smurf amplifier is to either completely filter ICMP ping, or, if this is not possible due to monitoring software you use (there are much better way s to monitor availability than using ping) then filter out ICMP ping bound for your network and broadcast address of each sub-net.

To prevent machines in your network from being used to launch smurf attacks, place in filters that only allow your correct source addresses to pass out from your network.

10)Multiple choice question
17:32:38.110000 62.98.184.147 > a.b.c.0: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 1514 3e62 b893 E....e.....>b..
0x0010 cb6d ef00 0800 f7ff 0000 0000 cb6d ef00 .m.........m..

17:32:38.140000 62.98.184.147 > a.b.c.8: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 150c 3e62 b893 E....e.....>b..
0x0010 cb6d ef08 0800 f7ff 0000 0000 cb6d ef08 .m.........m..

17:32:38.140000 62.98.184.147 > a.b.c.63: icmp: echo request (wrong icmp csum) (ttl 236, id 2149)
0x0000 4500 0020 0865 0000 ec01 14d5 3e62 b893 E....e.....>b..
0x0010 cb6d ef3f 0800 f7ff 0000 0 000 cb6d ef3f .m.?.....m.?

What is the most likely cause of the traffic above ?
a) The Broadscan smurf tool scanning for networks that can be used as smurf amplifiers, one indicator of this is the static ICMP id numbers
b) A hardware fault in a router at 62.98.184.147 responding with an ICMP message telling us that the packet we sent has an incorrect ICMP checksum
c) A user on host 62.98.184.147 using the ping application to ping hosts in the a.b.c network to see if they are alive
d) A user on host 62.98.184.147 attempting to DoS our network by sending packets with incorrect ICMP checksums

Correct Answer is: A
Because of the static ICMP id's these packets are crafted. By hitting the network boundaries the source is hoping to receive multiple r esponses back to there single ping.
The ICMP checksums being incorrect is a bug in the broadscan software.

Detect 2

RPC portmap scan and RPC statd service query

1) Source of detect

The source of this detect is a perimeter intrusion detection system r unning on the targeted network.

2) Detect was generated by
This detect was generated by a Snort version 1.7 OpenBSD box running a modified snort ruleset from http://www.snort.org

3) Probability that source address was spoofed
As this scan requires an ini tial response before it version checks rpc.statd the probability of the source address being spoofed is extremely low.

4) Description of Attack
The attacker is scanning for hosts with portmapper enabled. When a host is found listening, a request is made t o the portmapper asking if statd is listening. This can be used to gather information about the machine or to see if a vulnerable statd application is running. This probe is recorded as CAN -1999-0632

Tcpdump data:

```
10:00:28.981829 210.178.180.226.688 > a.b .c.4.111: udp 56 (ttl 47, id 51649)
0x0000   4500 0054 c9c1 0000 2f11 7fd0 d2b2 b4e2   E..T..../.......
0x0010   cb6d ef04 02b0 006f 0040 485f 3841 2c33   .m.....o.@H_8A,3
0x0020   0000 0000 0000 0002 0001 86a0 0000 0002   ................
0x0030   0000 0003 0000 0000 0000 0000 0000 0000   ................
0x0040   0000 0000 0001 86b8 0000 0001 0000 0011   ................
0x0050   0000 0000                                 ....
```

This is clearly a statd query to portmapper. As you can see from the above tcpdump output bytes 0x0029 - 0x002b contain the byte pattern of " 01 86 a0". This is the byte pattern for a statd query.

```
10:00:39.751829 210.178.180.226.689 > a.b.c.6.111: udp 56 (ttl 47, id 53663)
0x0000   4500 0054 d19f 0000 2f11 77f0 d2b2 b4e2   E..T..../.w.....
0x0010   cb6d ef06 02b1 006f 0040 d8e2 546a 7f83   .m.....o.@..Tj..
0x0020   0000 0000 0000 0002 0001 86a0 0000 0002   ................
0x0030   0000 0003 0000 0000 0000 0000 0000 0000   ................
0x0040   0000 0000 0001 86b8 0000 0001 0000 0011   ................
0x0050   0000 0000                                 ....
```

Snort Portscan log:

```
May 11 10:00:28 210.178.180.226:3628  -> a.b.c.2:111 SYN ******S*
May 11 10:00:28 210.178.180.226:3626  -> a.b.c.0:111 SYN ******S*
May 11 10:00:28 210.178.180.226:3630  -> a.b.c.4:111 SYN ******S*
May 11 10:00:28 210.178.180.226:688  -> a.b.c.4:111 UDP

--- These have been cut for sanity reasons  ----------------------------------------------

May 11 10:00:28 210.178.180.226:3731  -> a.b.c.105:111 SYN ******S*
May 11 10:00:28 210.178.1 80.226:3732 -> a.b.c.106:111 SYN ******S*
May 11 10:00:39 210.178.180.226:689  -> a.b.c.6:111 UDP
```

May 11 10:00:40 210.178.180.226:3733 -> a.b.c.107:111 SYN ******S*

--- These have been cut for sanity reasons ---------- ------

May 11 10:00:40 210.178.180.226:3880 -> a.b.c.254:111 SYN ******S*
May 11 10:00:40 210.178.180.226:3878 -> a.b.c.252:111 SYN ******S*
May 11 10:00:40 210.178.180.226:3740 -> a.b.c.114:111 SYN ******S*
May 11 10:00:41 210.178.180.226:3741 -> a.b.c.115:111 SYN ******S*

It is interesting to note above that the portscan source ports are incremental, suggesting that the source machines only network activity is to scan for port 111. Notice however, that when the two RPC statd queries are issued they have a starting source port of 688 ending with 689. This shows that a new function has been called to query the portmapper for RPC statd.

Snort Alert:

05/11-10:00:17.780171
[**] <ts0> spp_portscan: PORTSCAN DETECTED on ts0 from 210.178.180.226 (THRESHOLD 4 connections exceeded in 0 seconds) [**]

05/11-10:00:28.413502
[**] <ts0> RPC portmap request rstatd [**]
05/11-10:00:28.981829 0:90:27:F:22:A2 -> 0:50:BA:C3:EA:27 type:0x800 len:0x62
210.178.180.226:688 -> a.b.c.4:111 UDP TTL:47 TOS:0x0 ID:51649 Ip Len:20 DgmLen:84
Len: 64

[**] <ts0> spp_portscan: portscan status from 210.178.180.226: 107 connections across 106 hosts: TCP(106), UDP(1) [**]

05/11-10:00:32.100106
[**] <ts0> spp_portscan: portscan status from 210.178.180.226: 1 connections across 1 hosts: TCP(0), UDP(1) [**]

05/11-10:00:39.753802
[**] <ts0> RPC portmap request rstatd [**]
05/11-10:00:39.751829 0:90:27:F:22:A2 -> 0:A0:24:CB:70:62 type:0x800 len:0x62
210.178.180.226:689 -> a.b.c.6:111 UDP TTL:47 TOS:0x0 ID:53663 IpLen:20 DgmLen:84
Len: 64

[**] <ts0> spp_portscan: portscan status from 210.178.180.226: 139 connections across 139 hosts: TCP(139), UDP(0) [**]

05/11-10:00:43.677558
[**] <ts0> spp_portscan: End of portscan from 210.178.180.226: TOTAL time(13s) hosts(244) TCP(245) UDP(2) [**]

05/11-10:00:47.685902

5) Attack Mechanism
Scanning for portmapper listening on port 111. The attacker when a listening host is found queries the portmapper for the details regarding statd. If a vulnerable version of statd were to be found, a buffer overflow attack would no doubt be attempted.

6) Correlation's
This host is reported at http://www.mynetwatchman.com probing numerous networks for port 111

7) Evidence of active targeting
There is no evidence to suggest active targeting; this was a probe across the entire sub-net looking for hosts listening on port 111.

8) Severity
Criticality = 5 (This machine is the authoritative DNS server)
Lethality = 1 (This machine is not vulnerable to this attack)
System Countermeasures = 5 (This machine has the l atest security patches applied)
Network countermeasures = 2 (Network contains a firewall but packet was passed)

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

9) Defensive recommendation

The RPC scans managed to cross this network perimeter due to human error. There was a rule change made to this device but the rules were not correctly applied. The total time of exposure was less than one hour. With the amount of hostile activity present today that kind of exposure opens your network to numerous abuses from multiple attackers. Fortunately, the probed statd was not vulnerable to the attack so no attack was ever issued.

The defensive recommendation is to firstly set up your perimeter devices to block the portmapper port and the port which rstat is listening on. If you are not using NFS then there is no need to run statd, in this case disable the executable in inetd.conf. If you are running NFS statd should be patched with the recommended security patches and restarted.

10)Multiple choice question
How do attackers attempt to find RPC services on a Solaris machine if RPC portmapper is blocked?
(Choose the most correct answer)

a) This is impossible on Solaris, you must request the RPC application number and port from RPC portmapper before connecting to an RPC service.
b) RPC services run on ports above 32774 on Solaris. Knowing this, an attacker can connect to a RPC service without using portmapper
c) By sending through a lone ACK which will by pass the firewall you can still communicate with RPC portmapper and request the RPC service p ort
d) The best way to find what socket a RPC service is running on is to portscan the entire port ranges from 1 -65535.

Correct answer is: B
RPC services run on static ports above 32774 on a Solaris box. These are well known and can be used to directly comm unicate with a RPC service.

Detect 3

DNS server scan and iquery request:

1) Source of attack detect
The source of these detects are firewall syslogs on the perimeter firewall and an intrusion detection device running at the perimeter.

2) Detect was generated by
This detect data was generated by Ipfilter syslogs and Snort 1.7

3) Probability that source address was spoofed
As this scan requires an initial response before it issues an inverse query the probability of the source address being spoofed is ex tremely low.

4) Description of Attack
The attacker is scanning IP addresses sequentially, scanning for hosts that are listening on port 53 TCP. When a listening host is found the attacker has sent an inverse query to the nameserver in question.

IPfilter syslogs:
Apr 30 05:42:31 machine ipmon[4275]: 05:42:31.431374 de0 @0:15 b
211.104.253.185,4888 -> a.b.c.143,53 PR tcp len 20 60 -S IN
Apr 30 05:42:31 machine ipmon[4275]: 05:42:31.441656 de0 @0:15 b
211.104.253.185,4889 -> a.b.c.144 ,53 PR tcp len 20 60 -S IN
Apr 30 05:42:31 machine ipmon[4275]: 05:42:31.459696 de0 @0:15 b
211.104.253.185,4890 -> a.b.c.145,53 PR tcp len 20 60 -S IN

--- These have been cut for sanity reasons ---------- -----

Apr 30 05:42:35 machine ipmon[4275]: 05:42:34.915617 de0 @0:15 b
211.104.253.185,4932 -> a.b.c.186,53 PR tcp len 20 60 -S IN
Apr 30 05:42:35 machine ipmon[4275]: 05:42:34.918486 de0 @0:15 b
211.104.253.185,4933 -> a.b.c.187,53 PR tcp len 20 60 -S IN
Apr 30 05:42:35 machine ipmon[4275]: 05:42:34.923920 de0 @0:15 b
211.104.253.185,4934 -> a.b.c.188,53 PR tcp len 20 60 -S IN
Apr 30 05:42:35 machine ipmon[4275]: 05:42:34.929267 de0 @0:15 b
211.104.253.185,4935 -> a.b.c.189,53 PR tcp len 20 60 -S IN

Again we find a machine that is only scanning for hosts listening on certain sockets. This can be seen by the fact that the source port is incrementing by one. Notice that when the inverse query is issued a new range of source ports is used.

TCPDUMP output:
05:42:34.462896 211.104.253.185.3646 > a.b.c.94.53: [udp sum ok] 43981 inv_q+

```
[b2&3=0x980] A? . (23) [tos 0x10] (ttl 48, id 29255)
  0000: 4510 0033 7247 0000 3011 3cda d368 fdb9  E..3rG..0.<..h..
  0010: d208 385e 0e3e 0035 001f bd22 abcd 0980  ..8^.>.5..".....
  0020: 0000 0001 0000 0000 0000 0100 0120 2020  ..........
  0030: 2002 61                                  .a
```

5) Attack Mechanism
By locating systems running bind, the attacker will t hen attempt an inverse query to determine if the nameserver in question supports inverse queries. There is a buffer overflow in BIND 4.9 and BIND 8 which could be used to effect access as the user running the nameserver (usually root) This vulnerability is recorded as: Cert CA - 98.05 and CVE-1999-0009

6) Correlation's
Scans for listening port 53 (DNS) services are quite common, on incidents.org the day this event was recorded there were 18 such events recorded. None originated from the source address on this detect however.

7) Evidence of active targeting
There is no evidence to suggest active targeting; this was a probe across the entire sub-net looking for hosts listening on port 53.

8) Severity
Criticality = 5 (Attack is against an authoritative dns ser ver)
Lethality = 5 (If attack was successful the attacker could execute commands as root)
System countermeasures = 5 (This system is patched with the latest security patches)
Network Countermeasures = 2 (Network contains a firewall but packet was passed)

(5+5) – (5+2) = 3

9) Defensive recommendation

Inverse queries are normally off by default. Check in your named.conf file for a line containing the words "fake iquery", if they exist then you have enabled iquery and are vulnerable.

The defensive recommendat ion is to upgrade to a latter version of bind and in the meantime disable iqueries, block TCP port 53 access to defeat the attackers original probe and only allow TCP port 53 from hosts you zone transfer with.

10)Multiple choice question
What is an inverse q uery ?
a) An inverse query is when a nameserver queries a client asking for its hostname rather than a client querying a nameserver
b) An inverse query looks up an IP address based on its hostname using a non - standard query type
c) An inverse query looks u p a hostname based on its IP address using a non -standard query type
d) An inverse query is a query to the inverse DNS server, which returns the inverse of your IP address

Correct answer is: B
An inverse query is an obsolete non -standard method of transla ting hostnames into IP addresses

Detect 4

Queso fingerprint attempt

1) Source of attack detect
The source of this detect is from both firewall syslogs on the perimeter firewall and an intrusion detection device running at the perimeter.

2) Detect was generated by
This detect was generated by IPfilter syslogs and Snort 1.7 output

3) Probability that source address was spoofed
As this appears to be a QueSo fingerprint packet, the attacker would require a response, therefore the probability that the so urce address is spoofed is low.

4) Description of Attack
Now it could be argued that this is an ECN echo rather than a QueSo packet.
I believe this is QueSo. This is a single packet directed toward a critical machine. There are no further attempts from th is host to contact this machine on port 111. This source host is also recorded in the mynetwatchman incidents as a host scanning for port 111.

TCPDUMP output:
16:24:01.417949 202.99.23.180.37054 > a.b.c.37.111: S [ECN -Echo, CWR} 1450658282:1450658282(0) w in 5840 <mss 1460,sackOK,timestamp 6938599 0,nop,wscale 0> (DF) (ttl 52, id 38413, len 60)

```
0x0000    4500 003c 960d 4000 3406 ea69 ca63 17b4    E..<..@.4..i.c..
0x0010    d208 1225 90be 006f 5677 49ea 0000 0000    ...%...oVwI.....
0x0020    a0c2 16d0 5852 0000 0204 05 b4 0402 080a   ....XR.....
0x0030    0069 dfe7 0000 0000 0103 0300              .i.....
```

IPfilter syslogs:
May 20 16:24:0 machine ipmon[4322]: 16:24:01.418082 dc0 @0:99 b 202.99.23.180,37054 -> a.b.c.37,111 PR tcp len 20 60 -S IN

5) Attack Mechanism
By sending a packet with the 1 and 2 reserved bits set and a TCP SYN, the attacker hopes to fingerprint a machine. If the remote host is listening, a SYN and Ack will be returned to the attacker, if the host is not listening a RST will be sent back. Using these responses the attacker can identify the TCP/IP stack in question.

6) Correlation's
This attacker is also recorded at http://www.mynetwatchman.com as atacking host 210.14.x.x on port 111

7) Evidence of active targeting
This was an active target, there was no portscan detected and this was the only machine to be hit.

8) Severity
Criticality = 5 (This machine is a primary webserver for a web based company)
Lethality = 2 (This is an Operating System fingerprint)
System Countermeasures = 5 (This machine has all recommended security patches)
Network Countermeasures = 5 (The firewall prevented this from reaching its destination)

(5+2) – (5+5) = -3

9) Defensive recommendation
This probe was stopped at the firewall perimeter, as portmapper requests are not allowed from the outside. Stopping QueSo packets is difficult especially now considering legitimate ECHO traffic also uses the two reserved bits.

10) Multiple choice question

How can you tell if a packet with the 1, 2 and SYN flags set is QueSo and not ECN ?
a) Web providers only use ECN so the destination port should be port 80
b) With a QueSo packet the source will not respond with a SYN ACK after receiving the responding ACK from the destination
c) With a QueSo packet, the source will respond with a SYN ACK after receiving the responding ACK from the destination
d) ECN isn't used much through the network, so it must be a QueSo fingerprint

Correct answer is: B
You can only really be sure that you are witnessing a QueSo fingerprint by the lack of response from the source to your ACK.

Detect 5

DNS version.bind query

1) Source of attack detect
The source of this detect is from an intrusion detection device running at the perimeter of this network.

2) Detect was generated by
This detect was generated by Snort 1.7 using a standard snort rulesfile from http://www.snort.org

3) Probability that source address was spoofed
As the attacker is expecting a response to this query, the likelyhood of this being spoofed is low.

4) Description of Attack

The attacker is using a multi -threaded application to scan for hosts listening on TCP port 53 (DNS). When a host is found to be listening a version.bind query is issued. This is a recon probe used to ascertain the version of BIND being run.

Jun 4 10:51 :17 211.233.25.147:3668 -> a.b.c.109:53 SYN **S*****
Jun 4 10:51:17 211.233.25.147:3672 -> a.b.c.113:53 SYN **S*****
Jun 4 10:51:17 211.233.25.147:3676 -> a.b.c.117:53 SYN **S*****
Jun 4 10:51:17 211.233.25.147:3621 -> a.b.c.62:53 SYN **S*****
Jun 4 10:51:17 2 11.233.25.147:3684 -> a.b.c.125:53 SYN **S*****

--- This has been cut for sanity ---

[**] IDS278 - SCAN -named Version probe [**]
06/04-10:51:17.486318 0:2:B9:A1:20:60 -> 0:80:C8:CA:A1:8D type:0x800 len:0x48
211.233.25.147:1090 -> a.b.c.94:53 UDP TTL:49 T OS:0x0 ID:30508
Len: 38

--- This has been cut for sanity ---

Jun 4 10:51:20 211.233.25.147:3741 -> a.b.c.182:53 SYN **S*****
Jun 4 10:51:20 211.233.25.147:3600 -> a.b.c.41:53 SYN **S*****
Jun 4 10:51:20 211.233.25.147:3623 -> a.b.c.64:53 SYN **S*****
Jun 4 10:51:20 211.233.25.147:3629 -> a.b.c.70:53 SYN **S*****

5) Attack Mechanism
The attacker scans for hosts listening on TCP port 53. When a host is found, a packet containing the version.bind command is sent to the host in question.
Using this data the att acker can match the BIND version against a list of known vulnerable BIND releases.

Version.bind is a command that returns the application version of DNS servers. Setting the DNS class to CHAOS and the DNS Query to TXT activates it.

The CHAOS class was o riginally included in BIND for the Chaosnet network that used the Hesiod software. MIT was historically one of the users of the Hesiod Class.

6) Correlation's
Mynetwatchman records scans to BIND TCP port 53 on June 4 2001 to 2 networks, 210.14.x.x and 24.1 .x.x. It lists the attacker as coming from kidc.net

7) Evidence of active targeting
There is no evidence of active targeting, the attacker has scanned this network for listening DNS servers and has issued the query when a listening server is found.

8) Severity
Criticality = 5 (This host is an authoritative nameserver)
Lethality = 5 (If successful, this attack will give the attacker root access)
System Countermeasures = 5 (This hosts security patches are up to date)
Network Countermeasures = 2 (There is a fir ewall in the perimeter but it passed the

packet)

(5+5) – (5+2) = 3

9) Defensive recommendation
In your named.conf file, which is the config file for BIND, an option can be specified which returns bogus version.bind responses. This is set in your named.co nf file with the statement version (text you wish to respond to version.bind queries with); for example:

version you_must_be_joking;
This will return you_must_be_joking to any version.bind query.

Use filtering routers and firewalls to block TCP connectio ns to port 53. Only allow hosts you zone transfer with to conduct TCP connections to port 53.

10) Multiple choice question
What are the steps to retrieve the application version of BIND from the DNS server you are currently using? Using nslookup (There a re two correct answers to this)
a) nslookup –q=txt –class=CH version.bind
b) nslookup –class=txt –q=CH version.bind
c) nslookup
   class=CHAOS
   query=TXT
   version.bind
d) nslookup
   query=CHAOS
   class=TXT
   version.bind

The correct answers are: A and C
CHAOS is a class originally included for the Chaosnet protocols and QUERY is the type of DNS query. Such as TXT for text, A for address etc.

Assignment II - Describe the state of Intrusion Detection

IDS Flexible Response Systems
Dean White

Flexible Response
" The capability of military forces for effective reaction to any enemy threat or attack with actions appropriate and adaptable to the circumstances existing. "
Defense Technical Information Center

With the advent of Network I ntrusion Detection Systems (ID systems), an ageless problem returns. This problem is the ability to flexibly and sensibly respond to new threats and to adapt to the changing environment.

So what is Flexible Response? Flexible Response is, as quoted above, " The capability of [a system] for effective reaction to any enemy threat or attack with actions appropriate and adaptable to the circumstances existing." So what does that mean for the field of network security?

For network and host security, Flexible R esponse is a technology which allows a system, such as an ID system, to interact with other security devices. These include devices such as firewalls and routers in the same environment and to respond quickly to attacks in an attempt to prevent further dam age. In this sense, it is able to adapt to changing attacks.

To understand how this works in the ID field, lets quickly review traditional ID system thinking. Traditionally an ID system passively listened to the network around it; all the while collectin g data and matching signatures. The administrator was alerted to an event as close to the time of the event as technology allowed (for example an alphanumeric page sent to a pager). This was analysis preformed in retrospect to tthe event, by some other me ans, such as storing the event in a database or system log for later review.

No direct action was taken to prevent the attacker continuing with their attack. The ID system was primarily used as a tool to determine what had occurred at a later date. The ID system provided one of the many avenues for event correlation, as an example, in combination with perimeter device logs.

Flexible response, however, takes the ID system the next step forward. This is done by allowing the ID system to not only detect att acks such as buffer overflows, but to respond directly to events by sending connection blocks or warning messages. Naturally there are some concerns with taking what was a passive device and turning it into an active engine; One that is capable of acting i n a hostile way to hostile events.

A well-documented example of an ID system containing a flexible response agent is Snort, a free ID system package. With snort, it is possible to terminate active connections using the following response codes.

- RST_SND, this response generates a TCP reset directed at the source of the threat, effectively causing the source to terminate the current connection.

- RST_RCV, this response generates a TCP reset directed at the destination of the threat, preventing the destinatio n from responding to the event.

- RST_ALL, this response generates a TCP reset in both directions causing the source and destination to close the connection.

- The ICMP_NET response generates an ICMP net unreachable (ICMP Code 3 Type 0) message to the sende r, advising the sender that the host it has attempted to connect to is unreachable.

- The ICMP_HOST response generates an ICMP host unreachable ICMP Code 3 type 1) message to the sender of the packet, informing the sender that the host they wish to commun icate with is not reachable.

- The ICMP_PORT response generates an ICMP port unreachable (ICMP Code 3 type 3) message informing the sender that the UDP port they are trying to connect to at the destination is not in listening mode (i.e. No service is bound to that port)

- The ICMP_ALL sends all of the above ICMP messages. I believe this is an incorrect response in all circumstances and should not be used. It is never seen in normal traffic and clearly shows an attacker that a flexible response engine exists in your network.

Finally react. React is the act of closing the connection (blocking) and/or sending a visible notice to the sender, informing them (via http) that there connection has been closed or denied.

Naturally, the correct response needs to be selected. There is no point generating an ICMP_NET unreachable when a user is trying to connect to a port on a machine he can ping and get a response to. The very fact he can ping the device tells him that the network is reachable. The correct response in this circumstance, dependant on protocol would be, for TCP connections, generate a reset and for UDP connections generate an ICMP_PORT unreachable (ICMP Code 3 Type 3) message.

Checkpoints OPSEC, which takes this idea one step further, provides interaction between ID systems such as ISS Realsecure and Checkpoints FW/1 firewall package.

OPSEC not only provides all the above Flexible Responses, but allows the ID system to issue commands to the Checkpoint FW/1 firewall to insert rules to filter the offender at the perimeter.

As you can well imagine, this becomes a very powerful tool. For example, if an attacker issues a buffer overflow against a BIND DNS server. Using the most appropriate response, this connection can be torn down and a rule added into your FW/1 firewall to prevent this person from re-connecting to the DNS server. Thus, preventing the buffer overflow from ever being executed and preventing a re-try from the attacker.

By now you may well be seeing some limitations with the OPSEC type of flexible response. Firstly, what if I could spoof that I was an ID system and deliver OPSEC messages to the Checkpoint FW/1 Firewall, informing the firewall to block and deny connections from legitimate users. This very thing was in fact a vulnerability in FW/1 which was addressed in some recent service packs.

What if the attackers aim was not to buffer overflow the DNS server at all, but to block you, a legitimate user from being able to use the DNS server? What would an attacker need to do to prevent you from accessing the DNS server?

If the attacker could spoof your source IP address, they could issue a command they know would illicit a response from the ID system that would cause an OPSEC block message to be sent to the Checkpoint FW/1 Firewall. If the OPSEC response was enabled on an incorrect signature, it could lock you out from being able to use your

DNS server. (denial-of-service)

What do I mean by an incorrect signature? There are some activities that generating an OPSEC response for makes a lot of sense, then there are signatures which open you up for huge denial of service opportunities from attackers. This has been the most hotly debated topic with regards to Flexible Response. You need to choose wisely which rules your apply OPSEC responses to .

Guardian by Anthony Stevens is a similar package to OPSEC, it was developed for Snort and Linux, and allows Snort and a Linux IPChains firewall to basically interact the same way as OPSEC on a Checkpoint FW/1 Firewall and ISS Realsecure IDS' work.

Guardian generates IP Chains rules based on Snort alerts to deny and any further packets from the attacker getting to the system

Naturally, it has the same advantages and disadvantages as OPSEC does.

In conclusion, Flexible Response provides a very powerful way of rapidly responding to attacks against your critical infrastructure. When used correctly, it can stop an attacker dead in his tracks, offering a superior level of protection than just a simple firewall and ID system combination. When used incorrectly it can be a conduit for denial-of-service attacks, thus leaving your critical services unreachable by your employees, yourself and your customers.

In critical systems the integrity and security of data is paramount. Loss of data is more costly than temporary service outages. OPSEC and Flexible Response could be a robust solution to the problem of protecting your classified documents from walking out through your network front door.

Assignment III - Analyze This!

The following is an analysis based on partial data. Therefore in some instances it is impossible to completely ascertain what has happened.

During the period there were power outages and hardware faults with the deployed device that left gaps in the data sampling. However, over 100 megabytes of data was still collected during the period.

As a consequence of these interruptions to service, this report can only be a survey and summary of the major events that occurred during the period.

A number of these attacks show the compromise of your systems, many however, are false positives, events which match the IDS signature but upon further analysis are discovered to be legitimate traffic.

In this report, I wish to concentrate on the actual compromised machines and whom they were compromised by and what steps the attacker took to compromise these machines. (Where there is enough data present to correlate.)

During the analysis of the data five machines were identified as being compromised by attacks which suggest the Ramen worm. Some of them are being used to scan and attack other machines, thus spreading the worm.

Other machines also display traffic patterns suggesting they have services listening on port 27374, the Ramen and SubSeven ports. These machines are to be regarded as compromised till otherwise ascertained.

Below is a listing of hosts affected by the Ramen Worm of whom actual data exists showing the method of initial intrusion.

IP Address	Exploited Vulnerability	Attacker	Date/Time
MY.NET.219.22	WU-FTPD	128.61.136.233	3/6/01 16:44
MY.NET.130.81	RPC.STATD	171.65.61.201	2/20/01 19:42
MY.NET.105.169	RPC.STATD	171.65.61.201	2/20/01 19:42
MY.NET.105.91	RPC.STATD	171.65.61.201	2/20/01 19:42
MY.NET.181.127	RPC.STATD	171.65.61.201	2/20/01 19:42

A summary of the events regarding the abov e machines follows:

MY.NET.179.78:
There is no data that allows the exploit date of MY.NET.179.78 to be determined, however the portscan file SnortS34.txt shows that MY.NET.179.78 is scanning host 162.33.212.88 on Feb 6. This is indicative of a compromise d machine.

Further, attack source 24.67.186.244 whilst scanning for Ramen/SubSeven on tcp 27374 receives a response packet from MY.NET.179.78, indicating that a service is listening.

02/11-23:15:41.729932 [**] Possible RAMEN server activity [**]
MY.NET.179.78:27374 -> 24.48.226.183:3108
02/23-23:11:44.278445 [**] Possible RAMEN server activity [**]
24.67.186.244:1244 -> MY.NET.179.78:27374
02/23-23:11:44.278565 [**] Possible RAMEN server activity [**]
MY.NET.179.78:27374 -> 24.67.186.244:1244

MY. NET.219.22:
This host was compromised by host 128.61.136.233 using the WU -FTPD vulnerability, this occurred at 16:44 on March 6. There is no scan data to indicate after the wu-ftpd attack was launched that MY.NET.219.22 began scanning.

However, whilst attack source 24.67.186.244 is scanning for Ramen/SubSeven on 27374, MY.NET.219.22 responds indicating there is a service listening. This is the first solid lead this machine has been successfully compromised.

02/23-23:14:32.606383 [**] Possible RAMEN server a ctivity [**]
MY.NET.219.22:27374 -> 24.67.186.244:3426
02/23-23:14:33.233285 [**] Possible RAMEN server activity [**]
MY.NET.219.22:27374 -> 24.67.186.244:3426

03/06-16:44:02.658052 [**] SITE EXEC - Possible wu-ftpd exploit - GIAC000623 [**] 128.61.136.233:4705 -> MY.NET.219.22:21

MY.NET.130.81:
Host 171.65.61.201 compromised this host on Feb 20 at 19:42 using the RPC statd attack. There is no record of an original scan from the attacker that would have discovered this system was running RPC. However, because of the data from the hosts below it can be assumed that an eternal RPC call was issued at 19:42

There is also no scan data to indicate that after the attack, MY.NET.130.81 began to scan the Internet looking for new systems to attack.

Further, attack source 24.67.186.244 on Feb 23 whilst scanning for Ramen/SubSeven on TCP 27374 receives a response packet from MY.NET.130.81, indicating that a service is listening. This is the first solid lead this machine has been successfully compromised.

02/23-23:08:12.884910 [**] Possible RAMEN server activity [**] 24.67.186.244:4607 -> MY.NET.130.81:27374
02/23-23:08:12.884956 [**] Possible RAMEN server activity [**] MY.NET.130.81:27374 -> 24.67.186.244:4607
02/20-19:42:51.102298 [**] STATDX UDP attack [**] 171.65.61.201:891 -> MY.NET.130.81:941

MY.NET.105.169
Host 171.65.61.201 compromised this host on Feb 20 at 19:42 using the RPC statd attack. An external RPC call is recorded as coming from the attacker at 19:42.

Attacker 24.67.186.244 on Feb 23 at 23:06 whilst scanning for Ramen/SubSeven on TCP 27374 receives a response packet from MY.NET.105.169, indicating that a service is listening. This is the first solid lead this machine has been successfully compromised.

Earlier in the month on Feb 11, MY.NET.105.169 is scanned by 24.48.226.183, looking for listening Ramen or SubSeven. There is no response indicating the service is not listening. Based on this I am surmising that this host was indeed compromised on Feb 20, Ramen was inserted and this is why on Feb 23 when 24.67.186.244 is scanning for TCP port 27374 there is a response.

02/11-23:10:35.128255 [**] Possible RAMEN server activity [**] 24.48.226.183:4278 -> MY.NET.105.169:27374
02/23-23:06:29.797797 [**] Possible RAMEN server activity [**] 24.67.186.244:2217 -> MY.NET.105.169:27374
:02/23-23:06:30.386511 [**] Possible RAMEN server activity [**] MY.NET.105.169:27374 -> 24.67.186.244:2217
02/20-19:42:33.683596 [**] STATDX UDP attack [**] 171.65.61.201:873 -> MY.NET.105.169:32774
02/20-19:42:34.124788 [**] External RPC call [**] 171.65.61.201:4894 -> MY.NET.105.169:111

MY.NET.105.91
Host 171.65.61.201 compromised this host on Feb 20 at 19:42 using the RPC statd attack. An external RPC call is recorded as coming from the attacker at 19:42

Attacker 24.67.186.244 on Feb 23 at 23:06 whilst scanning for Ramen/SubSeven on TCP 27374 receives a response packet from MY.NET.105.91, indicating that a service is listening. This is the first solid lead this machine has been successfully compromised.

02/23-23:06:28.589604 [**] Possible RAMEN server activity [**]
24.67.186.244:2139 -> MY.NET.105.91:27374
02/23-23:06:28.590951 [**] Possible RAMEN server activity [**]
MY.NET.105.91:27374 -> 24.67.186.244:2139
02/20-19:42:32.945358 [**] External RPC call [**] 1 71.65.61.201:4792 ->
MY.NET.105.91:111
02/20-19:42:33.320412 [**] STATDX UDP attack [**] 171.65.61.201:871 ->
MY.NET.105.91:798

MY.NET.181.127
Host 171.65.61.201 compromised this host on Feb 20 at 19:42 using the RPC statd attack. In this case no externa l RPC call was recorded.

Also, no traffic is recorded to suggest that 24.67.186.244 ever scanned or received a response from MY.NET.181.127. However it is probably safe to assume this data was dropped and that the traffic coincides with the previous data recorded for other hosts.

However, at 19:51 on Feb 20 after being compromised, host MY.NET.181.127 begins scanning the Internet for hosts listening on port 111. This is indicative of a compromised host.

02/20-19:45:33.132877 [**] STATDX UDP attack [**] 171.65.61.201:936 ->
MY.NET.181.127:910

Other related events:

Attack source 24.67.186.244 scanned all of the MY.NET class B address at 10:00 looking for Ramen and or SubSeven Trojans. During this scan over one thousand devices responded as listening on port 27374 TCP. All of these hosts are most likely compromised.

Detailed Description of Attack types involved with the compromise.
The following is a detailed description of the type of attacks and probes that led to the compromise of these machines. Rec ons and fingerprinting not involved with the actual exploits have been left out, except in the case where there are network issues that need to be brought to your attention.

These descriptions of the probes, attacks and fingerprints critical to the explo its include the following sub -headings with explanations:

➓ The classification of the type of event detected

- The severity of the event
- A brief description of what the event is
- A recommendation to limit exposures to these attacks in the future.

Event Site exec buffer overflow WuFTPD
Event Classification: Attack
Event Severity: Critical

Description:
Washington University ftp daemon (wu -ftpd) is vulnerable to a format string vulnerability that can be used to gain a shell and execute commands as the user running the ftpd server (usually root). This attack can be issued by any user, both authenticated users and anonymous users making this a very serious vulnerability. Redhat 6.2 and Redhat 7.0 Operating systems are common targeted systems.

Recommended action:
The machines listed above have been compromised using this vulnerability. Unfortunately, these machines have probably had root kits installed that can be very hard to fully remove. It is recommended that these machines are re -installed or restored from backup.

No ftp server is completely immune from problems however, due to the numerous security holes in wu -ftpd over the lifetime of the product, the recommendation is to run another ftp daemon, such as BSD ftp.

Event StatDX UDP
Event Classification: Attack
Event Severity: Critical

Description: RPC.statd is vulnerable to buffer overflow. This attack can be issued locally by an authenticated user or remotely by an attacker with no valid authentication.
It allows the attacker to execute commands as root making this another serious vulnerability. Redhat 6.2 is a common target for this attack.

Recommended action:
Ensure that routers and firewalls filter unwanted access to this service.
RPC statd not only has buffer overflows, but also can leak unwanted information about your hosts. The recommendation is to seriously consider shutting down this service.

Event Connect to 515 from Inside
Event Classification: Attack and Recon
Event Severity: Low to Critical (depending on event)

Description:
LP the line printer daemon is the well -known service running on this socket. There have been recent vulnerabilities in LPRng found on Linux distributions.

Most of these detects are probably false positives, people attempting to print on

remote printers or mis-configuration of printcaps.

It is also possible that machines have been infected with the Ramen worm and are being used to scan for Redhat 7.0 Linux boxes with a vulnerable version of LRPng.

Ramen is a linux worm that uses the LRPng vulnerability to acc ess the remote machine and configures the new host system to scan for more vulnerable systems.

Recommended action:
Ensure that your machines are not victims of the Ramen worm.
Use filtering routers and firewalls to prevent inside access to port 515 unles s remote printers are used. In the case of remote printers, install pass rules only for the hosts you require access to.

There is a script written by William Stearns available from SANS at http://www.sans.org/y2k/ramen.html which can be used to detect the presence of Ramen.

External References:
More information regarding Ramen can be found at
http://www.sans.org/infosecFAQ/malicious/ramen3.htm

Event External RPC call
Event Classification: Recon and Attack Pre-Cursor
Event Severity: Low

Description:
RPC portmapper running on socket 111 is a service that is used to map RPC services running on a system. It contains a list of available RPC services and the sockets each of these services is listening on. RPC services have had numerous vulnerabilities and it appears that attacker is trolling for listening portmapper services. When a portmapper is found listening on a vulnerable machine, portmapper will be requested to list the services it has to offer and the sockets these services are listening on.

This is listed on the SANS Top ten common vulnerability list. RPC services rpc.ttdbserverd (Tool Talk), rpc.statd and rpc.cmsd (Calendar Manager) all have known vulnerabilities.

Note: Early versions of NMAP were capable of Denial-Of-Service attacks against Portmapper.

Recommended action:
Ensure systems running RPC services are patched when new vulnerabilities arise.
Shut down RPC services you are not using. Vendor web pages are a good source for required patches and information regarding RPC service requiremen ts.
Ensure that routers and firewalls filter unwanted access to this service.

Other things of interest that should be investigated:

There is a large report of traffic inside this network where the source and destination IP address lie outside the local network. Some of these (IP address range 224.0.0.0 - 224.255.255.255) are multicast traffic from video conferencing etc.

Others appear to come from dial up providers such as AOL. One possible explanation for this is multi-homing. I am suspecting users wit hin the local network also have modems connected to their machines and they are dialing into AOL. As the default gateway listed on their machines is through the LAN, packets with their dial in addresses as source are being routed across the local LAN inste ad of through the modems.

There is some SNMP Public Community String traffic being recorded. This is used to gather information from remote devices. It is a UDP protocol using socket 161 and 162. Information about your systems can be leaked out into the I nternet if community strings are not configured.

Configuring filtering routers and firewalls to block access to these ports will elevate some of this problem. Fixing the default community strings of public and private should be done immediately.

Tiny fragments are also being detected. The aim of these is to avoid some signature based ID systems and also circumvent filtering devices such as filtering routers. It could also lead to DoS potential. Because the full header is not included in the packet there is a chance it will pass through a filtering router because its signature does not match a block rule. Dropping all fragmented TCP packets that have a length of less than 8 octets in length can defeat a tiny fragment attack.

In conclusion:

5 hosts within this network are running the Ramen worm; it is recommended these machines be restored from a backup or preferably re -installed as rootkits may exist also on the backups.

Hundreds of IP addresses in this network appear to be listening on 27374, the port usually associated with Ramen or SubSeven. These machines should be considered as compromised until auditing can be done to establish if they have been attacked or not.

Users appear to have dial in modems within the network; this explains much of the source and destination address activity detected.

Packets with destinations of 224.x.x.x are multicast packets that contain requested video and audio streams.

I recommend a firewall be put in place, there appears to be no perimeter protection within this network, naturally with the size eof this network this is not always practicable. In these circumstances, multiple firewalls at sub -networks are a much more feasible option.
Finally, due to the numerous outages with the ID device itself, much of the data

required to provide full analysis was lost. I recommend this machine be placed on an UPS and new hardware be purchased and burnt in.

A secondary ID device in case of failure is always a good idea and can aid with correlation of detected events.

List of References:

Butler, Max "Arachnids Database" http://www.whitehats.com/ids/index.html (23 -May-2001)

Internet working Group, "RFC 1157: A simple Network Management protocol" Url:http://andrew2.andrew.cmu.edu/rfc/rfc1157.html (1 -June-2001)

Internet Working Group "RFC 1858 Security Considerations for IP Fragment Filtering"
http://andrew2.andrew.cmu.edu/rfc/rfc1858.html (1 -June-2001)

Defense Technical Information Center, "flexible response",
URL: http://www.dtic.mil/doctrine/jel/doddict/data/f/02482.html (24 -May-2001)

Roesch, Martin "Writing Snort rules - URL: http://www.snort.org/ (24 -May-2001)

Stephens, W. Richard. "TCP/IP Illustrated, Vol. 1" Reading: Addison Wesley Longman, Inc, 1994, 70-71

Internet Security Systems, "ISS Realsecure Console users guide ", About OPSEC responses, Internet Security Systems, 2000, 31 -35

Checkpoint, "Potential Security Issues recently identified in VPN -1/FireWall-1",
URL: http://www.checkpoint.com/techsupport/alerts/list_vun.html (25 -May-2001)

Stevens, A "Guardian Respons e Application"
URL: http://www.snort.org/Files/Guardian.tar.gz (24 -May-2001)

"mynetwatchman.com - Intrusion reporting and response"
URL: http://www.mynetwatchman.com (1 -June-2001)

"CVE-1999-0009",
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=1999-0009 (1-June-2001)

"CVE-2000-0666"
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0666 (23-May-2001)

"CVE-1999-0080"
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=1999-0080 (23-May-2001)

Collins, Jack R. "RAMEN, a Linux worm"
URL: http://www.sans.org/infosecFAQ/malicious/ramen3.htm (1-June-2001)

Sterns, William "SANS Global Incident Analysis Center > Ramen Worm"
URL: http://www.sans.org/y2k/ramen.html (1-June-2001)

Albitz, Paul & Liu, Cricket, "DNS and BIND" O'Reilly & Associates, Inc, 19 98, 17