Jeremy Hewlett

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

Assignment 1 – Network Detects, 35 points.

Detect #1:

IDS alert:
```
00:39:32  [T]  202.135.23.122  x.x.0.227   [IIS:UNICODE2] (tcp,dp=80,sp=4284)
```

Payload:
```
GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c: HTTP/1.0
```

1. Source of trace:
My network (work), Jan 1st 2001

2. Detect was generated by:
This detect was generated by a Dragon IDS sensor, version 4.2.1. Detect came from the web.lib ruleset, which was the most current at that date.

The fields are as follows:
| Time | direction of packet. In this case, "T" for to protected net | src host | dst host | alert name | (proto, dst port, src port) |

The second frame above is the command in the payload that was sent to the web server. This is standard http, fields are:

| Type of command (GET/PUT/POST/etc) | What to get | http proto version |

3. Probability the source address was spoofed:
It is not very likely this was spoofed. In order for this attack to succeed, the attacker would need to see the output generated from the GET command. Secondly, in order to initiate the GET command, the attacker would have to successfully complete the 3-way handshake.

4. Description of attack:
This is the extended Unicode attack for IIS versions 4 and 5. The attack allows attackers read, list or execute files outside of the web root. Information on this can be found under CVE ID CAN-2000-0884, Microsoft ID MS00-078 and BugTraq ID 1806. An example of some of the things that can be done with this can be found at http://www.securityfocus.com/templates/archive.pike?list=82&mid=144710, which was posted to the Vuln-Dev mailing list hosted at securityfocus.

5. Attack mechanism:
This attack works in much the same way web servers were vulnerable to adding a "../" (dot-dot slash) into the URL, thus allowing the attacker back out of the web root. Web server vendors applied fixes to their software, and hence requests with "../" are denied. This type of attack however replaces the aforementioned characters with the UNICODE translation of / or \.
The commands that could be run would be run under the built-in IUSR_machinename, which is

basically the equivalent of the 'nobody' user that apache typically runs under.

6. Correlation:

Stan Scalsky has had similar probes (http://www.sans.org/y2k/010301.htm) and interestingly enough, also uses dragon:

Greetings, 1) Monitoring summary from my Dragon sensors at a few of my sites show that the most popular exploit this holiday period is without a doubt the IIS UTF/UNICODE directory traversal exploit – probes like the following generally happen on a daily basis on our DMZ. Sometimes this is followed by an attempt to install some trojan program such as nc.exe via tftp or install a stealth cmd.exe.

```
 GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.1{D}{A}
 GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+copy%20c:\winnt\system32\cmd.exe%20sensepost.exeHTTP/1.0{D}{A
}
```

7. Evidence of active targeting:

At first glance this looks like active targeting because this is the only host out of approximately 1024 this person hit with an IIS exploit. This would make me a bit nervous, as this person must have done some recon first. However, after further analysis, it appears to be an automated script. I say this because the web server is not IIS, and not even windows. As to why just this one host was targeted, I can only speculate because its domain name could probably be considered "elite" in the realm of IRC.

8. Severity:

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

(4 + 1) – (5 + 2) = -2

System Criticality: 4 -> Web server running.
Attack lethality: 1 -> It's not windows or IIS. Unlikely to do any damage at all.
System countermeasures: 5 -> I give this a 5 because all of the IIS servers are fully patched, so had the attack been directed at an IIS box, it would have failed.
Network countermeasures: 2 -> Permissive firewall, no access control.

9. Defensive recommendation:

Defenses here would be to keep all the IIS servers updated and patched, and using an IDS to keep an eye on them as well as the rest of the network. Since this is a public webserver, using TCP wrappers or firewalling to certain hosts wouldn't be feasible. However, one could always deny this person's IP. ☺

10. Write a question that is based on the trace and your analysis with your answer:

Based on the packet payload as pasted earlier, what did the attacker hope to accomplish?

A. Check the existence of cmd.exe
B. Execute the command c+dir+c
C. Get a directory listing of C:

D. Get a directory listing of /winnt/system32

Correct answer, C

Detect #2:

IDS alert:

```
18:39:08  [T]  62.252.236.70   x.x.208.10  [FTP:CWD-DOS] (tcp,dp=21,sp=3253)
```

Payload #1:

```
USER anonymous
PASS guest@here.com
CWD /pub/
MKD 001228004049p
CWD /public/
CWD /pub/incoming/
PORT 216,25,117,6,1,21
CWD
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppp
```

1. Source of trace:

My network (work), Dec 27th, 2000.

2. Detect was generated by:

Dragon IDS sensor, version 4.2.1. Detect came from ftp.lib ruleset, which was the most current at that date.

The fields are as follows:
| Time | direction of packet. In this case, "T" for to protected net | src host | dst host | alert name | (proto, dst port, src port) |

In the payload, the commands issued are:
USER is the username
PASS is the password
CWD is "change working directory (cd)"
MKD is "make directory"
PORT is used to initiate a transfer (this is explained in detail later).

3. Probability the source address was spoofed:

Not very high. Since a connection was made to the ftp server and commands issued, a 3-way handshake has already been completed.

4. Description of the attack:

This detect just kept getting better. The more I investigated the more neat-o things I ran into. The initial alert that caught my eye was the CWD DoS as pasted above. After looking for more of this on other machines on the network, I found 3 more:

```
18:39:08  [T]  62.252.236.70   x.x.208.60  [FTP:CWD-DOS] (tcp,dp=21,sp=3303)
18:39:10  [T]  62.252.236.70   x.x.208.125 [FTP:CWD-DOS] (tcp,dp=21,sp=3368)
18:39:19  [T]  62.252.236.70   x.x.208.250 [FTP:CWD-DOS] (tcp,dp=21,sp=3493)
```

Looking at the CVE entries (CVE-1999-0219 , CVE-1999-0362 , CAN-2000-0131), this is an attack for windows based FTP servers such as serv-u, warftpd and ws_ftp. When commands such as cwd or mkdir are issued with long arguments, a DoS or overflow is the end result. BugTraq ID 612 says that proftpd (vers 1.2pre1, 1.2pre3 and 1.2pre3) also has a similar buffer overflow with mkdir. Lastly, BugTraq ID 961 mentions Tiny FTPd (ver .52) has a buffer overflow problem with the commands "APPE, MKD, RMD, RNFR, RNTO, SIZE, STOR, XMKD, and XRMD."

So it would seem that someone wanted to crash/exploit these servers, but wait, there's more! Doing a search for this src IP produced the following:

```
18:39:02  [T]  62.252.236.70   x.x.208.20  [FTP:USER-ANON] (tcp,dp=21,sp=3263)
18:39:02  [T]  62.252.236.70   x.x.208.21  [FTP:USER-ANON] (tcp,dp=21,sp=3264)
18:39:02  [T]  62.252.236.70   x.x.208.24  [FTP:USER-ANON] (tcp,dp=21,sp=3267)
18:39:02  [T]  62.252.236.70   x.x.208.23  [FTP:USER-ANON] (tcp,dp=21,sp=3266)
18:39:02  [T]  62.252.236.70   x.x.208.27  [FTP:USER-ANON] (tcp,dp=21,sp=3270)
18:39:02  [T]  62.252.236.70   x.x.208.10  [FTP:USER-ANON] (tcp,dp=21,sp=3253)
```

```
18:39:03 [T] 62.252.236.70 x.x.208.29 [FTP:USER-ANON] (tcp,dp=21,sp=3272)
18:39:03 [T] 62.252.236.70 x.x.208.25 [FTP:USER-ANON] (tcp,dp=21,sp=3268)
18:39:03 [T] 62.252.236.70 x.x.208.32 [FTP:USER-ANON] (tcp,dp=21,sp=3275)
18:39:03 [T] 62.252.236.70 x.x.208.30 [FTP:USER-ANON] (tcp,dp=21,sp=3273)
18:39:03 [T] 62.252.236.70 x.x.208.35 [FTP:USER-ANON] (tcp,dp=21,sp=3278)
18:39:03 [T] 62.252.236.70 x.x.208.44 [FTP:USER-ANON] (tcp,dp=21,sp=3287)
18:39:03 [T] 62.252.236.70 x.x.208.39 [FTP:USER-ANON] (tcp,dp=21,sp=3282)
18:39:03 [T] 62.252.236.70 x.x.208.41 [FTP:USER-ANON] (tcp,dp=21,sp=3284)
18:39:03 [T] 62.252.236.70 x.x.208.60 [FTP:USER-ANON] (tcp,dp=21,sp=3303)
18:39:03 [T] 62.252.236.70 x.x.208.45 [FTP:USER-ANON] (tcp,dp=21,sp=3288)
18:39:03 [T] 62.252.236.70 x.x.208.47 [FTP:USER-ANON] (tcp,dp=21,sp=3290)
18:39:03 [T] 62.252.236.70 x.x.208.46 [FTP:USER-ANON] (tcp,dp=21,sp=3289)
18:39:04 [T] 62.252.236.70 x.x.208.50 [FTP:USER-ANON] (tcp,dp=21,sp=3293)
18:39:04 [T] 62.252.236.70 x.x.208.53 [FTP:USER-ANON] (tcp,dp=21,sp=3296)
18:39:04 [T] 62.252.236.70 x.x.208.51 [FTP:USER-ANON] (tcp,dp=21,sp=3294)
18:39:04 [T] 62.252.236.70 x.x.208.54 [FTP:USER-ANON] (tcp,dp=21,sp=3297)
18:39:05 [T] 62.252.236.70 x.x.208.61 [FTP:USER-ANON] (tcp,dp=21,sp=3304)
18:39:05 [T] 62.252.236.70 x.x.208.55 [FTP:USER-ANON] (tcp,dp=21,sp=3298)
18:39:05 [T] 62.252.236.70 x.x.208.62 [FTP:USER-ANON] (tcp,dp=21,sp=3305)
18:39:05 [T] 62.252.236.70 x.x.208.58 [FTP:USER-ANON] (tcp,dp=21,sp=3301)
18:39:05 [T] 62.252.236.70 x.x.208.64 [FTP:USER-ANON] (tcp,dp=21,sp=3307)
18:39:05 [T] 62.252.236.70 x.x.208.68 [FTP:USER-ANON] (tcp,dp=21,sp=3311)
18:39:05 [T] 62.252.236.70 x.x.208.71 [FTP:USER-ANON] (tcp,dp=21,sp=3314)
18:39:05 [T] 62.252.236.70 x.x.208.72 [FTP:USER-ANON] (tcp,dp=21,sp=3315)
18:39:05 [T] 62.252.236.70 x.x.208.76 [FTP:USER-ANON] (tcp,dp=21,sp=3319)
18:39:05 [T] 62.252.236.70 x.x.208.81 [FTP:USER-ANON] (tcp,dp=21,sp=3324)
18:39:05 [T] 62.252.236.70 x.x.208.86 [FTP:USER-ANON] (tcp,dp=21,sp=3329)
18:39:05 [T] 62.252.236.70 x.x.208.125 [FTP:USER-ANON] (tcp,dp=21,sp=3368)
18:39:05 [T] 62.252.236.70 x.x.208.89 [FTP:USER-ANON] (tcp,dp=21,sp=3332)
18:39:06 [T] 62.252.236.70 x.x.208.87 [FTP:USER-ANON] (tcp,dp=21,sp=3330)
18:39:06 [T] 62.252.236.70 x.x.208.95 [FTP:USER-ANON] (tcp,dp=21,sp=3338)
18:39:06 [T] 62.252.236.70 x.x.208.96 [FTP:USER-ANON] (tcp,dp=21,sp=3339)
18:39:06 [T] 62.252.236.70 x.x.208.131 [FTP:USER-ANON] (tcp,dp=21,sp=3374)
18:39:06 [T] 62.252.236.70 x.x.208.132 [FTP:USER-ANON] (tcp,dp=21,sp=3375)
18:39:06 [T] 62.252.236.70 x.x.208.101 [FTP:USER-ANON] (tcp,dp=21,sp=3344)
18:39:06 [T] 62.252.236.70 x.x.208.104 [FTP:USER-ANON] (tcp,dp=21,sp=3347)
18:39:06 [T] 62.252.236.70 x.x.208.138 [FTP:USER-ANON] (tcp,dp=21,sp=3381)
```
[The rest is cut for brevity. You get the idea, a scan over an entire class C network for anonymous logins.]

So, too much was going on in a very short period of time from this host for this to be considered normal ftp traffic. The odd thing here is, out of an entire class C (all of those hosts offering ftp services, and about half offering anonymous) only 4 were subjected to the "CWD DoS." So, what else is in store, what are in these other payloads I ask. I found this person issued the following commands to every server that allowed anonymous ftp connects.

Payload #2:

```
230 Guest login ok, access restrictions apply.
PORT 128,178,25,62,132,212
500 Illegal PORT Command
PASV
227 Entering Passive Mode (x,x,208,10,78,148)
STOR 1MB
425 Possible PASV port theft, cannot open data connection.
QUIT
221-You have transferred 0 bytes in 0 files.
```

So, putting the two payloads together (based on illegal PORT/PASV commands, searching for incoming directories and attempted file/directory creation) now I get the picture of a warez (pirated software) pup. How it all goes together and how it works is in the next section.

5. Attack mechanism:

In packet #1, after the person logs in we see him/her first check to see if a directory can be created. I suspect that whatever program is being used here attempts to find common *incoming* directories and attempt to write here as well. I say this because the directories /public and /pub/incoming do not even exist on the target machines. What's more, a common technique among warez pups is to quickly move warez from one server to another, even when the initiator is on a slow link. The technique is referred to as "FXPing." The odd PASV and PORT commands seem to reflect this type of activity. FXP works like so:

"A" is a site that allows anonymous uploads and downloads.
"B" is a site that has warez on it.
"C" is a modem user.

"C" needs to move/copy warez from "B" to "A." This would be painfully slow over a modem. An FXP client will connect to server "B" and depending on whether the transfer is coming or going, issue a PORT command with the IP address of server "A" instead of the typical address of the requesting IP ("C" in this case) or issue a PASV. The FXP client basically does the same thing on the other server and starts a transfer. The two servers are now talking to each other. You'll notice in the PORT commands above, the IP address listed are never the same as the initiating IP.

So, this person needs a server that will do one of two things:

1. Accept PORT commands for arbitrary IP addresses. "FTP port bouncing" works in a similar way too, however one also needs to also be able to specify ports <1024 in the command.
2. Accept PASV commands and allow arbitrary IP addresses to connect.

Just incase you're confused about the weird looking PORT/PASV commands, they don't use periods to separate out the 4 octets of an IP. They use commas. As for the last two numbers, the port number can be ascertained by multiplying the first number by 256 and adding the result to the second number. For example:

PORT 216,25,117,6,1,21

Is 256x1+21=**277**

Finally, the last thing that tipped me off is the "1MB" file that was attempted to be uploaded in payload #2. I figure the 1MB file is a 1 meg file that is used to gauge the speed of the site. Since none of the ftp sites allowed the upload, I couldn't verify this. So, I did the next best thing, I asked other people. ☺

6. Correlations:

To verify my theory of warez pups, I asked cmg@edu, an intrusion analyst at a university. He provided the following snort logs and commentary (These are snort logs.
The format of this is |*date-time*|*[**]alert name[**]*|*src ip:port*|*dst ip:port*| The stuff underneath is tcpdump style hex.):

They found a writeable share from one of the isp's ips they store their 1000000 byte 1mb file

```
1/05-11:53:26.463104  [**] STOR 1MB - possible warez site [**] 62.4.245.154:1807 ->
xxx.xxx.xxxx.xxx:21

11:53:26.463104 adsl-9626.turboline.skynet.be.1807 > xxx.xxx.xxx.xxx.ftp: P
580945497:3580945511(14) ack 1499419871 win 17034 (DF)
                    4500 0036 3789 4000 7106 4ef4 3e04 f59a
                    xxxx xxxx 070f 0015 d570 e459 595f 54df
                    5018 428a 75c4 0000 5354 4f52 2031 4d42
                    7465 7374 0d0a
```

Later, they retrieve it twice (possibly another compromised host at
the same isp or DHCP adsl), presumably as a speed test / success test.

```
01/06-09:10:16.871468  [**] RETR 1MB - possible warez site [**] 62.4.244.66:1071 ->
xxx.xxx.xxx.xxx:21

09:10:16.871468 adsl-9282.turboline.skynet.be.1071 > 138.26.197.139.ftp: P
103710778:103710792(14) ack 2603923748
win 16983 (DF)
                    4500 0036 030a 4000 7106 84cb 3e04 f442
                    xxxx xxxx 042f 0015 062e 803a 9b34 b924
                    5018 4257 0386 0000 5245 5452 2031 4d42
                    7465 7374 0d0a
```

```
alert.978796860:01/06-09:28:36.941170  [**] RETR 1MB - possible warez site [**] 62.4.244.66:4446 ->
xxx.xxx.xxx.xxx:21

09:28:36.941170 adsl-9282.turboline.skynet.be.4446 > 138.26.197.139.ftp: P
541055472:541055486(14) ack 2741793853
win 16984 (DF)
                    4500 0036 3041 4000 7106 5794 3e04 f442
                    xxxx xxxx 115e 0015 203f d9f0 a36c 743d
                    5018 4258 bf3d 0000 5245 5452 2031 4d42
                    7465 7374 0d0a
```

Over the course of several months, many warez/mp3/vcd sites were created. Most of these sites had the 1mb file and often the remote machine was running IIS with a world writeable directories related to frontpage. Other times it they were unix hosts with insecure permissions.

When a site was to actually be used, often a directory containing the phrase "RoboFXP" was created. Probably a customized host to host ftp transfer program with speed testing options built in.

7. Evidence of active targeting:

This was an entire netblock scan, no specific machine was targeted. However, I still have an unknown as to what the "CWD DoS" stuff is that was apparently directed to only 4 IPs. I suspect I'm actually missing some packets, as the particular IDS that caught this has a problem with dropping packets when the network gets bursty.

8. Severity:

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

(2 + 2) – (5 + 2) = -3

System Criticality: 2 -> Not terribly important, these are servers that are left over from a network move.
Attack lethality: 2 -> I give this a 2 because if the intruder had been able to put warez on our network, all this really could do is max out network bandwidth when the machine makes it to a warez list.
System countermeasures: 5 -> This gets a 5 because none of the servers allow anonymous uploading, or arbitrary, illegal PORT/IP addresses.
Network countermeasures: 2 -> Permissive firewall, no access control

9. Defensive recommendations:

Turn off anonymous ftp access on the servers that don't need it. However, for the rest of them, due to the nature of anonymous ftp, it would be impossible to limit who can connect. In this case, keeping the ftp software current and patched is the best defense. If an anonymous server needs upload access, then the upload directory shouldn't be readable and should also restrict downloads until the administrator can review what was uploaded and move it to a publicly available area.

10. Multiple choice test question:

What should be done on the ftp servers now?
A. Nothing. They're fine.
B. Check for unauthorized files and directories. Perhaps a server allowed anonymous uploads.
C. Reinstall.
D. Move the ftp server to a different port to hide it.

Correct answer, B.

Detect #3:

IDS Alert:

```
06:28:34  [T]  193.140.46.97  x.x.65.7    [WEB:MYLOG] (tcp,dp=80,sp=2656)
06:37:06  [T]  193.140.46.97  x.x.65.7    [WEB:MYLOG] (tcp,dp=80,sp=2886)
```

Payload:

06:28:34

```
GET /mylog.phtml?screen=/etc/passwd HTTP/1.0
```

06:37:06

```
GET /mylog.phtml?screen=/etc/passwd HTTP/1.0
```

1. Source of trace:

My network (work), Jan 14th , 2001.

2. Detect was generated by:

Dragon IDS sensor, version 4.2.1. Detect came from web.lib ruleset, which was the most current at that date.

The fields are as follows:
| Time | direction of packet. In this case, "T" for to protected net | src host | dst host | alert name | (proto, dst port, src port) |

The payload format is of:
| Type of command (GET/PUT/POST/etc) | What to get | http proto version |

3. Probability the source address was spoofed:

Not very high. At this point, the attacker has already made a connection to the server and requested information. A 3-way handshake must be completed in order to do this. It is a possibility that the target machine could be a victim of sequence number guessing too. The target machine, however, is a relatively recent OS with all patches installed, so this type of attack shouldn't be very easy at all. I'm betting my quarter on this not being spoofed.

4. Description of the attack:

This is the "PHP/FI mylog/mlog" attack that will present the password file back to the attacker, if the script exists and is vulnerable. In the logs above, it looks like the attacker was doing other things while scanning for this particular exploit. I say this because of the src port the first time the attacker connected and the src port the second time there was a connection. The difference in src port numbers suggests there were other connections elsewhere. However, why the person came back again is a bit of a mystery. Perhaps a script mishap. The machine in question does not have /etc/passwd, it isn't even UNIX. CVE gives this vulnerability ID CVE-1999-0346 and BugTraq's ID is 713.

5. Attack mechanism:

Much like many other CGI type exploits, the scripts put too much trust in the (ab)user on the other end. One of the golden rules for scripts of this type is to accept only the characters needed by the script. Typical examples of things to not accept are things like slashes, dot-dots, pipes, or generally anything that is a special character to the shell or other program. In this particular case, no filtering was done on the input of the "screen" variable. So, one could define screen as "/etc/passwd" and viola, a passwd file.

6. Correlations:

A couple of people made references in their practicals of also being scanned for this type of vulnerability. Those can be found here:

http://www.sans.org/y2k/practical/EAVazquezJr.html
http://www.sans.org/y2k/practical/Todd_Garrison.html#attack1-source1

Todd Garrison mentions RFP's whisker web scanner. That does indeed look for this vulnerability. However, looking at whisker (v1.4), it does not append "?screen=/etc/passwd" as was in the payload. Also looked at Nessus (v1.0.6), cgichk (v2.42) and Retina (v2.0, evaluation) and none of these appeared to look for m[y]log.[p]html. Searches on google returned only discussions of how to use this vulnerability. Based on how easy this is, I suspect this was done either by hand or some home-brewed cgi scanner.

7. Evidence of active targeting:

This attack was only directed to this one machine. This is definitely active targeting. However, there appears to have been no recon done because, as mentioned before, this is not a UNIX box and therefor has no /etc/passwd. This could be a "fire and forget" scan in the hopes of turning up something somewhere.

8. Severity:

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

(4 + 1) – (5 + 2) = -2

System Criticality: 4 -> Web server
Attack lethality: 1 -> This isn't going to succeed at all.
System countermeasures: 5 -> Up-to-date OS, all patches.
Network countermeasures: 2 -> Permissive firewall, no access control.

9. Defensive recommendations:

Since the mylog scripts are just test scripts, and since test scripts have been known to occasionally have problems, I recommend going through all the web servers and disabling or removing any test scripts that are found. Also, I would make sure all the UNIX boxes have shadowed password files, so in the event a passwd is retrieved no password hashes are also obtained.

10. Multiple choice test question:

Why would retrieving an /etc/passwd file be beneficial?

A. Could be used as method of harvesting usernames for Spam bots.
B. Sometimes /etc/passwd will contain password hashes (machine doesn't use a method of shadowed passwords).
C. Gives an attacker usernames to brute force a login password from.
D. All of the above.

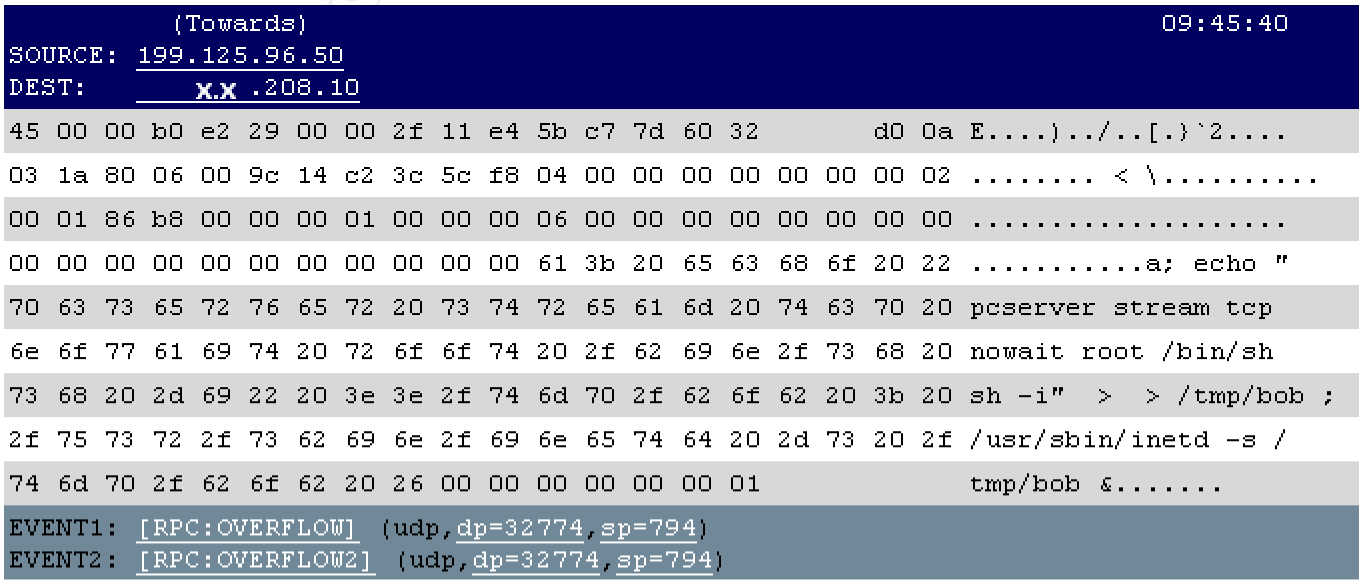
Correct answer, D

Detect #4

IDS alert:

09:45:40 [T] 199.125.96.50 x.x.208.10 [RPC:OVERFLOW] (udp,dp=32774,sp=794)

Payload:



```
          (Towards)                                            09:45:40
SOURCE: 199.125.96.50
DEST:       x.x .208.10
45 00 00 b0 e2 29 00 00 2f 11 e4 5b c7 7d 60 32       d0 0a E....)../..[.)`2....
03 1a 80 06 00 9c 14 c2 3c 5c f8 04 00 00 00 00 00 00 00 02 ........ < \..........
00 01 86 b8 00 00 00 01 00 00 00 06 00 00 00 00 00 00 00 00 ....................
00 00 00 00 00 00 00 00 00 00 00 61 3b 20 65 63 68 6f 20 22 ...........a; echo "
70 63 73 65 72 76 65 72 20 73 74 72 65 61 6d 20 74 63 70 20 pcserver stream tcp
6e 6f 77 61 69 74 20 72 6f 6f 74 20 2f 62 69 6e 2f 73 68 20 nowait root /bin/sh
73 68 20 2d 69 22 20 3e 3e 2f 74 6d 70 2f 62 6f 62 20 3b 20 sh -i"  >  > /tmp/bob ;
2f 75 73 72 2f 73 62 69 6e 2f 69 6e 65 74 64 20 2d 73 20 2f /usr/sbin/inetd -s /
74 6d 70 2f 62 6f 62 20 26 00 00 00 00 00 00 01             tmp/bob &.......
EVENT1: [RPC:OVERFLOW]  (udp,dp=32774,sp=794)
EVENT2: [RPC:OVERFLOW2]  (udp,dp=32774,sp=794)
```

1. Source of trace:

My network (work), Jan 5th 2001.

2. Detect was generated by:

Dragon IDS sensor, version 4.2.1. Detect came from the overflow.lib ruleset, which was the most current at that date.

The fields are as follows:
| Time | direction of packet. In this case, "T" for to protected net | src host | dst host | alert name | (proto, dst port, src port) |

The payload above is a snapshot of Dragon in "raw" mode. This includes the ASCII, hex, the alerts generated (the "event" stuff), protocol, and the src and dst ports/IPs. I figure I'd spice it up a bit and include everything and not just the application layer. ☺

3. Probability the source address was spoofed:

Unlike the other detects, this attack went over UDP. Remember that UDP is a connectionless protocol (in other words there is no 3-way handshake, so there is no connection reliability). Because of this UDP is an easy protocol to spoof. That being said, I don't believe this to be a spoof. My reasoning behind this is that the only other packet sent to this machine from this host was a request to portmapper requesting statd's port. The attacker would have to have seen the reply in order to immediately connect again to statd on port 32774. Also, the TTL in the above hex dump is 47, and on connecting to that machine's HTTP port returns a syn+ack with a TTL of 52 which is pretty close. The IP ID is 57897 and when compared to the IP ID on the initial portmap request (seen below in the "question" section), the ID is 57894. Again, that appears to be correct. More on TTLs can be found at http://www.switch.ch/docs/ttl_default.html

4. Description of the attack:

This attack is an older statd exploit, not like the "format string" statd exploit that's still very popular at this time. If this attack works, the inetd parseable line "*pcserver stream tcp nowait root /bin/sh sh -i*" would be echoed to "/tmp/bob." After which another copy of inetd would be started specifying /tmp/bob as a configuration file. The CERT advisory on this can be found at http://packetstorm.securify.com/advisories/cert/CA-97.26.statd

5. Attack mechanism:

This mechanics behind this are really simple. In this attack, statd does not have the proper bounds checking for user input, thus submitting a specially crafted packet an attacker can force statd to execute the above command. As we all know from using the shell, the semicolon allows us to specify multiple commands on one line. So, the attacker has added commands to be done as root (or the user statd is running as). When inetd executes this code, it will start an interactive shell on port 600 (pcserver) for the attacker to come in on and do whatever else.

6. Correlations:

While searching around and looking at who else has seen this attack, I came across a few others who have seen this problem:

http://www.health.ufl.edu/wss/mail-archives/unix-sec/1999//05//msg00008.html
http://www.securityportal.com/list-archive/firewalls/2000/Mar/0058.html

Interestingly, even though this exploit is old it still continues to be used.

7. Evidence of active targeting:

Definitely active targeting. Out of 4 other class C networks, only this host was hit. After this initial attack, there was no other evidence that this IP has been back in an attempt to exploit something else.

8. Severity:

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

(5 + 5) – (5 + 2) = 3

System Criticality: 5 -> Daily operations server for employees, also has access to other parts of the network.
Attack lethality: 5 -> This attack can provide remote root access.
System countermeasures: 5 -> Up-to-date OS, all patches.
Network countermeasures: 2 -> Permissive firewall, no access control, statd open to the world.

9. Defensive recommendations:

TCPWrap all services to only those who need them. Firewall the machine to provide the same sort of ACL. Since statd (and other RPC) services are needed, keep close watch on patches and stay current.

10. Multiple choice test question:

In the following hex dump of a portmap port request, what sequence of hex represents the program ID of statd?

```
45 00 00 54 e2 26 00 00 2f 11 e4 ba c7 7d 60 32 cc fd xx xx E..T.&../....}`2....
03 19 00 6f 00 40 f6 38 3b 5c f8 24 00 00 00 00 00 00 00 02 ...o.@.8;\.$........
00 01 86 a0 00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00 ....................
00 00 00 00 00 00 00 00 00 01 86 b8 00 00 00 01 00 00 00 11 ....................
00 00 00 00                                                 ....
```

A. 00 03 00 00
B. 00 01 86 b8
C. 3b 5c f8 24
D. c7 7d 60 32

Correct answer, B

Assignment 2 – Analyze This, 40 points.

This is an analysis of incomplete snort logs because for one reason or another, such as failed power or a full disk, logging had been disabled. The below are the results of doing said analysis for GIAC Enterprises.

Even though it was said that I would be provided with one month's worth of logs, it appears the logs I have received encompass more than just one month. There even seem to be some alert files that are duplicates. For example, SnortA14 and 19 are both of size 88368 and have no differences. These duplicates were ignored during the actual analysis. Within the archive of snort logs, there are three types of text files: Alerts, Scan logs, and Packet Dumps. After all the zips were unpacked and put in a safe place, and a little rearranging of some of the information within the text files was completed, the total amount of disk space used by the text files is 56.6M.

The snort alerts start on 09/26 and end on 11/22. Days missing are 09/29, 10/04, 10/10, 10/14, 10/17, 10/20, 10/27, 11/01, 11/06, 11/11, 11/15, 11/18 and 11/21.

First, we'll start out with a list (via snortsnarf, which can be found at www.snort.org) of alerts in ascending order. The span of the dates and times is also included:

Earliest alert at **00:00:52**.873106 *on 09/26*
Latest alert at **23:32:20**.988483 *on 11/22*

Signature (click for definition)	# Alerts	# Sources	# Destinations
Happy 99 Virus	2	2	2
site exec - Possible wu-ftpd exploit – GIAC000623	6	4	4
Tiny Fragments - Possible Hostile Activity	7	5	6
SITE EXEC - Possible wu-ftpd exploit – GIAC000623	7	1	4
External RPC call	13	8	3
Probable NMAP fingerprint attempt	15	14	13
connect to 515 from inside	56	2	3
SUNRPC highport access!	60	13	12
NMAP TCP ping!	96	21	20
Queso fingerprint	142	29	58
SMB Name Wildcard	218	33	33
Null scan!	283	204	196
SNMP public access	468	23	1
Back Orifice	1697	40	932
Broadcast Ping to subnet 70	1813	216	1
Attempted Sun RPC high port access	2542	20	33
TCP SMTP Source Port traffic	2893	4	2836
WinGate 1080 Attempt	4802	570	2655
Watchlist 000222 NET-NCFC	8166	45	26
Watchlist 000220 IL-ISDNNET-990517	30998	61	108
SYN-FIN scan!	56250	30	25751

Based on the information above, the top five source addresses are listed below and ranked by the number of alerts. I tried including all source addresses, but there were such a large number of hosts with large amounts of alerts it just wasn't feasible.

Top Talkers

# alerts	IP Address	Whois Info
7226	160.78.49.191 *ema.chim.unipr.it*	Centro di Calcolo di Ateneo (NET-PARMANET1)
6660	208.61.4.207 *adsl-61-4-207.mia.bellsouth.net*	BellSouth.net Inc. (NETBLK-BELLSNET-BLK7)
4993	209.92.40.32 *dslcv1-32.fast.net*	FASTNET Corporation (NETBLK-DSL1-FASTNET) DSL1-FASTNET
3921	63.195.56.20 *adsl-63-195-56-20.dsl.snfc21.pacbell.net*	SNFC21 RBACK11 BASIC 63.195.56.0 (NETBLK-SBCIS39515) SBCIS39515
3886	130.89.229.48 *cal032044.student.utwente.nl*	University Twente (NET-UTNET)

160.78.49.191

160.78.49.191 performed a very noisy SYN-FIN scan for DNS servers on the networks using src port 53. Out of all the scans, it seems that the only servers that were sent anything other than SF were the following:

```
Sep 30 13:10:40 160.78.49.191:1327 -> 192.168.1.3:53 UDP
Sep 30 13:19:42 160.78.49.191:1337 -> 192.168.109.38:53 UDP
Sep 30 13:19:42 160.78.49.191:1339 -> 192.168.109.41:53 UDP
Sep 30 13:19:46 160.78.49.191:1340 -> 192.168.110.16:53 UDP
Sep 30 13:21:31 160.78.49.191:2773 -> 192.168.130.122:53 SYN **S*****
Sep 30 13:21:31 160.78.49.191:1345 -> 192.168.130.122:53 UDP
Sep 30 13:25:52 160.78.49.191:1348 -> 192.168.181.131:53 UDP
Sep 30 13:27:47 160.78.49.191:1350 -> 192.168.204.30:53 UDP
Sep 30 13:27:50 160.78.49.191:1351 -> 192.168.204.134:53 UDP
Sep 30 13:29:02 160.78.49.191:1352 -> 192.168.218.162:53 UDP
```

These servers should be checked to see what versions/type of DNS software they are running and whether the results are vulnerable or not. 122 looks particularly interesting, I'd start there.

208.61.4.207, 209.92.40.32

The next two hosts in our list, 208.61.4.207, and 209.92.40.32, were likely to be trolling for already compromised machines. Like the previous machine, this was a noisy SF scan going all over the network. In other words, the third octet was incremented by one after rotating through all the ips in the last octet. Src and dst port was 9704. Looking around for a bit of correlation, most of what I found were just other people asking what this is, or making note that the scan had happened. It is interesting to note that the SEQ/ACK numbers do not change very often and the window size is always 0x404, and the ID number is always 39426. The tool that scans with this signature is Psychoid's SynScan. This also was formerly known on the snort mailing list as "Mystery scan tool 11."

http://www.sans.org/y2k/082000-1100.htm
http://www.sans.org/y2k/081600-1500.htm
http://www.sans.org/y2k/practical/Dale_Ross_GCIA.htm
http://www.sans.org/y2k/practical/william_stearns_gcia.html

As I've seen this port before being used on compromised machines, my guess is that (as stated above) this is a commonly used backdoor port. Some searches from the Incidents list at securityfocus confirm this.

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D75%26mid%3D78737

The following link suggests a statd worm uses this port:

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D75%26mid%3D142003

Grepping over the snort files showed no other information about this host other than the scan itself. Following the directions in the link above about grepping inetd for this port would shed some light on whether or not a particular machine running statd had been infected or not. Telneting to this port would also work.

63.195.56.20

The next host on our list (surprise surprise) performed another massive SF scan. However, this time it was a little different – a scan for FTP from src port 21. Everything else, scanner type and all, is the same… how unimaginative. Grepping through the snort files, there weren't any other alerts other than scans for this IP though that doesn't mean nothing has happened. Snort might not have had a signature in the rulebase for a particular attack by this IP. As above, looking over the versions and type of FTP servers running would be an excellent idea. Some recent CVE entries that may pertain to what this person was looking for are CAN-1999-0911, and CAN-2000-0573.

130.89.229.48

The last host on our top five list here is more of the same. Like the first host, this is a SF scan for DNS servers from port 53 to port 53 and using SynScan. Possible CVE entries that may relate to what the scanner is looking for are CVE-2000-0887, CVE-2000-0888 and CVE-2000-1169. JD Baldwin lists this type of scan in one of the detects at http://www.sans.org/y2k/practical/JD_Baldwin.html

Alerts

- **Happy99 Virus**

  This alert was triggered two times to (presumably) two of your mail servers on port 25. This makes sense as one of the methods of propagation for this virus is through email (the other being through news groups). Information on what this virus does can be found at http://www.geocities.com/SiliconValley/Heights/3652/SKA.HTM. Looking at a recent snort signature for happy 99 shows that it matches on the e-mail header of "X-Spanska: Yes." This isn't likely to false alarm much, if at all. It seems pretty specific. Mail logs should be checked for who the recipient of these viruses was. Andy Siske and Joanne Treurniet also make note of this in their practicals:

  http://www.sans.org/y2k/practical/Andy_Siske_GCIA.htm
  http://www.sans.org/y2k/practical/JoanneTreurniet.html

- **Site exec – Possible wu-ftpd exploit – GIAC000623**
- **SITE EXEC – Possible wu-ftpd exploit – GIAC000623** (There was a second signature)

  This is a particularly familiar alert for me. I've seen this type of alert everywhere. There have been quite a few wu-ftpd exploits lately. Dale Ross records in his practical (http://www.sans.org/y2k/practical/Dale_Ross_GCIA.htm) this particular alert, from the same number of hosts (4) in our alert.

  The following hosts were sources of this particular attack:

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total))
208.61.44.215	9	9	2	5
24.31.88.99	2	2	1	1
202.9.188.89	1	1	1	1
63.202.13.20	1	6	1	6

The destinations (lower case site-exec):

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total))
192.168.221.82	2	3	1	2
192.168.205.94	2	11	2	8
192.168.97.206	1	9	1	8
192.168.100.209	1	2	1	2

The destinations (upper case SITE-EXEC)

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total))
192.168.130.242	3	5	1	3
192.168.205.94	2	11	1	8
192.168.99.130	1	2	1	2
192.168.130.81	1	3	1	3

The first three src IPs didn't seem to have any other traffic recorded. However the last one seems to have done a bit of recon work. Perhaps a step above the average script kiddie:

```
10/04-11:56:00.850049  [**] Queso fingerprint [**] 63.202.13.20:1187 ->
192.168.100.127:21
10/04-11:56:14.289566  [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**]
63.202.13.20:1188 -> 192.168.100.209:21
10/04-11:56:27.511836  [**] Queso fingerprint [**] 63.202.13.20:1190 ->
192.168.130.98:21
10/04-11:56:46.630183  [**] Queso fingerprint [**] 63.202.13.20:1192 ->
192.168.163.17:21
10/04-11:57:24.186779  [**] Queso fingerprint [**] 63.202.13.20:1196 ->
192.168.205.94:21
10/04-11:57:46.592127  [**] Queso fingerprint [**] 63.202.13.20:1198 ->
192.168.214.186:21
```

CERT advisory CA-2000-13 discusses this problem. In fact, the release date is July 7, and the date above is only about 3 months after. Chances of these servers being vulnerable (if the admins were not diligent in keeping up to date) are high.

- **Tiny Fragments – Possible Hostile Activity**

  It is possible that this is related to some sort of small DoS, however I think that is unlikely, based on the small amount of alerts (7). Eric Hacker's practical has quite a few more of these alerts, of which I tend to lean towards his opinion of this being an attack in his case: http://www.sans.org/y2k/practical/Eric_Hacker.html#_Toc490920406

  However, still it could be an indication of some malicious intent to crash a machine with fragmentation re-assembly bugs. One should block and log these strange fragments at the firewall. In fact I've never actually seen a fragment less than 512 bytes in the wild. Fragments of 512 or less sounds like a good number to block.

- **External RPC call**

  All of these are requests to portmapper. Whether or not they were dumping a list of services or just requesting the port for a particular services (i.e.; statd) is unknown. This would require a packet dump. There doesn't seem to be one with this type of information. Regardless, this isn't a good thing to allow into one's network. There are too many exploits and too much information to be gleaned from this. This type of traffic needs to be blocked at the firewall, and if possible bind these services to internal IPs. This idea is mentioned in http://www.sans.org/infosecFAQ/firewall/blocking_cisco.htm. AKA *Top Ten Blocking Recommendations Using Cisco ACLs.*

  The following hosts should be checked that they are running RPC services and if they've been patched against all the recent vulnerabilities published lately:

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total))
192.168.6.15	9	63	7	9
192.168.100.130	3	8	3	5
192.168.15.127	1	4	1	4

It looks like especially bad news for 192.168.6.15 however, as it was then connected to on an RPC port:

```
11/11-11:08:56.576798 [**] SUNRPC highport access! [**]
211.46.110.81:690-> 192.168.6.15:32771
```

I would start looking for compromised machines on this host. The other two machines didn't have any record of SUNRPC highport access. Joanne Treurniet mentions in her practical (http://www.sans.org/y2k/practical/JoanneTreurniet.html) this type of traffic to the same IP (192.168.6.15) that had the highport access alert above. This practical is dated September 22, 2000, so it seems as though perhaps no one has turned off portmapper/SUNRPC to the world yet.

This brings us straight to another alert of interest.

- **SUNRPC highport access!**
- **Attempted Sun RPC high port access (This alert chart not shown, but talked about)**

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total))
192.168.206.222	21	323	2	11
192.168.202.242	20	38	3	5
192.168.212.186	4	7	1	4
192.168.228.62	3	5	1	3
192.168.97.59	3	9	1	3
192.168.53.23	2	5	1	4
192.168.253.114	2	9	1	5
192.168.179.78	1	2	1	2
192.168.140.51	1	1	1	1
192.168.6.15	1	63	1	9
192.168.206.218	1	3	1	3
192.168.53.14	1	3	1	3

Well, this doesn't look so good for a few of those. Let's take a closer look.

192.168.206.222:

Most of the connections regarding this alert (and the "Attempted" one) come from *.icq.aol.com:4000. These can likely be ignored as being ICQ. However, for the truly paranoid (like myself), Teri Bidwell wrote in his practical, http://www.sans.org/y2k/practical/Teri_Bidwell_GCIA.doc, while looking over RPC highport access alerts he came across links such as http://archives.neohapsis.com/archives/bugtraq/1999-q3/1514.html that detail ICQ worms.

Other connections that set off this alert a lot are from head.rwc.rhns.redhat.com:443 (which does run https, so can be ignored. This alert just matches on TCP to port 32771) and 216.10.12.30:2078, which is apparently a squid proxy if one were to telnet to that port.

192.168.202.242:

This IP is the same story as above. Only difference is a src 216.10.12.2:4600, which doesn't accept a connection. In fact, it's mostly the same (after looking a little closer) for all of the IPs **except** for the following alerts, which are more suspicious:

```
10/14-12:29:16.379139 [**] SUNRPC highport access! [**]
195.34.28.117:3191-> 192.168.97.59:32771 - This is a Russian dialup.
There are 3 connections in about 4 minutes.

09/28-13:34:31.092814 [**] SUNRPC highport access! [**]
24.18.90.197:2668-> 192.168.253.114:32771
09/28-13:38:23.094713 [**] SUNRPC highport access! [**]
24.18.90.197:2061-> 192.168.253.114:32771
09/28-13:28:03.304676 [**] SUNRPC highport access! [**]
24.18.90.197:4795-> 192.168.179.78:32771 - A md.home.com user.

10/03-19:01:24.124407 [**] SUNRPC highport access! [**]
129.123.6.14:50000-> 192.168.140.51:32771 - multi.ece.usu.edu. That
src port is bit high as well.

11/11-11:08:56.576798 [**] SUNRPC highport access! [**]
211.46.110.81:690-> 192.168.6.15:32771 - this one was noted above.

10/24-00:28:55.778708 [**] SUNRPC highport access! [**]
24.40.46.225:3486-> 192.168.206.218:32771 - A de.home.com user.
```

So it's not as bad as first thought, though the IPs listed above have the greatest chance of being owned via statd (assuming 32771 is the most common port for statd to use).

- **Connect to 515 from inside**

Here is what we have to go on:

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total))
192.168.101.142	54	54	1	1
192.168.179.78	2	2	2	2

Destinations receiving this attack signature

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total))

192.168.100.3	54	58	1	5
64.244.202.110	1	1	1	1
64.244.202.66	1	1	1	1

First, if the destination hosts are valid print servers, then we can settle down a bit. I suspect however this may not be the case. 192.168.100.3 could very well be a legitimate print spooler, and 192.168.101.142 is using it to print. This is probably true. I'd bet my quarter on 192.168.179.78 being compromised though. We saw above 192.168.179.78 was visited by the RPC highport fairy, now we see this very same IP connected to the 64.244.202.x addresses' printer port. These are, by the way, mail.healthcite.com and igw.healthcite.com:

```
11/22-11:24:06.406682 [**] connect to 515 from inside [**]
192.168.179.78:2274-> 64.244.202.110:515
11/22-11:33:56.296324 [**] connect to 515 from inside [**]
192.168.179.78:2707-> 64.244.202.66:515
```

192.168.179.78 should be looked into immediately. Should the machine prove to be compromised, getting in touch with healthcite.com would be wise. They may need to have a look at their own machines now.

- **NMAP TCP Ping**
- **Null scan!**

Nmap being so popular, it's not surprising there are so many alerts such as "TCP Ping" (96 total) and "NULL scan" (283). These pings are all mostly standard technique. Port 80 -> 80, 80 -> 53, 53 -> 53, or some variant thereof. Most places allow src port 80/53 into their networks for obvious reasons, so these are good src ports to chose from.
There are significantly more Null scans. According to the ruleset I have, a null scan is triggered when the SEQ/ACK/FLAGS are NULL. Fair enough… After looking at the scans, it could be caused by Napster users. The use of src ports 6699 and 6688 and IPs that are cable, dsl, dorm rooms, or otherwise fast connections seems to confirm this. http://www.sans.org/y2k/100200.htm has similar examples, as well as http://www.sans.org/y2k/060400.htm. The policy on MP3 sharing of course is up the people who make the site policies. According to "wc" and "sort" there are 124 unique 192.168.x.x addresses that match this pattern. If this type of activity is not permitted then a policy should be drawn up stating specifically how people are allowed to use company resources. Firewalling connections to those ports could also be effective as well.

As for the rest of the scan, there really is no method to the madness, it appears mostly random. However, right off I recognize some packet mangling (mangling, not crafting). A couple examples:

I've seen this one quite a bit.

```
11/10-17:46:25.867116  [**] Null scan! [**] 62.252.4.220:18245 -> 192.168.204.38:21504
```

As stated on various lists, some Nortel boxes mangling TCP headers are causing this oddball packet. To wit, 18245 is 4745 in HEX, which is GE in ASCII. 21504 is 5400, which is "T" in ASCII. Put it together and we have "GET." What is happening is the TCP data is getting translated as header and leading to the goofy stuff shown above. The actual list mail can be found at http://www.securityfocus.com/templates/archive.pike?list=75&mid=161200 I've also seen similar oddities on my own network, where normal web traffic will be doing its thing, then a munged up packet (i.e.; src and dst port 0) and then normal traffic again. Which I also see src and dst port 0 traffic in some of these NULL alerts.

- **Probable NMAP fingerprint attempt**
  (The top five)

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total))
24.95.192.51	2	2	1	1
24.180.134.156	1	3	1	1
193.231.207.72	1	1	1	1
128.54.203.218	1	1	1	1
24.9.64.57	1	1	1	1

- **Queso fingerprint**
  (The top five)

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total))
24.3.161.193	45	45	2	2
195.115.7.2	22	22	1	1
129.242.219.27	19	24	18	22
64.80.63.121	15	15	9	9
24.163.42.82	8	8	1	1

- **SYN-FIN scan!**
  (The top five)

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total))
160.78.49.191	7199	7199	7199	7199
208.61.4.207	6635	6635	6635	6635
209.92.40.32	4967	4967	4967	4967
63.195.56.20	3897	3897	3897	3897
130.89.229.48	3860	3860	3860	3860

In the last three bullets, we see the top five scanning hosts for each type of recon that was performed on the network. Fortunately, these were all incoming, and no scans of these types going out.

Again in there are some false positives. As in the NMAP fingerprint alerts, we have several alerts probably caused by Napster. If one follows the Snort mailing list, there are occasionally many alerts of this sort that are set off by Napster users. For example:

```
11/05-13:41:42.659984 [**] Probable NMAP fingerprint attempt [**]
128.54.203.218:6699-> 192.168.218.162:2803 - t8kim.resnet.ucsd.edu
```

Then there's the Queso false alarms:

```
09/26-07:07:58.389135 [**] Queso fingerprint [**] 24.3.161.193:32866-
> 192.168.145.9:110
09/26-07:17:58.400199 [**] Queso fingerprint [**] 24.3.161.193:32867-
> 192.168.145.9:110
09/26-07:27:58.275507 [**] Queso fingerprint [**] 24.3.161.193:32868-
> 192.168.145.9:110
```

This type of traffic happens like the above on a regular basis. In the snippet we see every 10 minutes someone checking his or her mail. The source ports also increments by one each time. This tells me that this machine is doing nothing else that would cause it to 'expend' another ephemeral port, so it is quiet, except for the occasional mail check.

There are however, a very large amount of real scans. Most of them aimed at ports 21, 23, 53, 9704, 27374, and 109.

21 and 53 have had numerous exploits published. 23 has had a couple problems, however this may also just be a check to see if telnet is open for a banner grab later, or perhaps just a check to see if the admin(s) are security conscious (i.e.; don't think about running ssh and turning off telnet). 9704 is as stated earlier looking for an already compromised machine from a statd attack. 27374 is again, another open port that indicates a machine that is compromised. For a reference of this trojan (and others) http://www.simovits.com/nyheter9902.html is a good start. That site lists port 27374 as Bad Blood, SubSeven, SubSeven 2.1 Gold, or Subseven 2.1.4 DefCon 8. It just also happens to be the distribution port of the Ramen worm that is now popular and affecting RedHat boxes. I'm not quite sure what scanning for pop2 (109) will show the attacker. Pop2 is a mostly abandoned protocol. My guess at first was they might be searching for older machines/installations. The existence of pop2 would confirm this. After a bit of searching I found a pop2 exploit referenced at http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0920, that's probably what they're after.

After these types of scans and recon has been performed, one should check the network to see if any of these ports answer back, this is especially true for new and unusual scans for ports that haven't been seen before and one isn't sure if that port could be listening. Adding snort rules that match on outgoing SYN+ACK or ACK (for TCP) on these well-known trojan ports would also shed some light on whether or not they are being used.

- **SMB Name Wildcard**

This is common traffic for Windows machines when trying to determine NetBIOS names when all that is available to go on is an IP address. Information that could be obtained from this is users logged in, domain names, workstation names, etc. This alert alerts quite a bit on my network and others too when the traffic is benign. As an example, a colleague of mine was once sent an email about his web server "hacking" a government machine via NetBIOS. In actuality, this web server was trying to resolve host names, and one of the ways in which it would attempt this is with name-query probes. According to http://www.whitehats.com/info/IDS177 this type of traffic is "background noise" and should only be considered when there is other forensic evidence that points to a problem. This port (and 138,139) should be filtered both in and out of the network. After looking at these alerts, I then proceeded to look to see what was talking on 139. IPs 192.168.221.82 and 192.168.224.150 seem to be acting unusual.

192.168.221.82:

On `Oct 16 18:34:04` to `Oct 16 19:17:13` this host scanned 2668 hosts within 192.168.x.x on port 139. Chances of this host having a NetBIOS worm are high. Looking through the snort logs for events that happened near this date are:

From the scan log…

```
Oct 16 11:01:44 212.34.32.91:1131 -> 192.168.221.82:22 SYN **S*****
Oct 16 11:01:44 212.34.32.91:1132 -> 192.168.221.82:23 SYN **S*****
Oct 16 11:01:44 212.34.32.91:1135 -> 192.168.221.82:79 SYN **S*****
Oct 16 11:01:45 212.34.32.91:1145 -> 192.168.221.82:443 SYN **S*****
Oct 16 11:01:45 212.34.32.91:1144 -> 192.168.221.82:224 SYN **S*****
Oct 16 11:01:45 212.34.32.91:1155 -> 192.168.221.82:8080 SYN **S*****
Oct 16 11:01:46 212.34.32.91:1150 -> 192.168.221.82:1524 SYN **S*****
Oct 16 11:01:48 212.34.32.91:1164 -> 192.168.221.82:60665 SYN **S*****
Oct 16 11:01:47 212.34.32.91:1151 -> 192.168.221.82:3128 SYN **S*****
Oct 16 11:01:47 212.34.32.91:1160 -> 192.168.221.82:26405 SYN **S*****
Oct 16 11:01:48 212.34.32.91:1165 -> 192.168.221.82:60666 SYN **S*****
Oct 16 18:34:04 192.168.221.82:3057 -> 192.168.0.3:139 SYN **S*****
Oct 16 19:17:13 192.168.221.82:4925 -> 192.168.255.238:139 SYN **S*****
```

And from the alert log…

```
10/16-16:55:26.342617  [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**]
24.31.88.99:62275 -> 192.168.221.82:21
10/16-16:57:49.491247  [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**]
24.31.88.99:62281 -> 192.168.221.82:21
```

Chances of 192.168.221.82 being owned just got significantly higher. This machine should be investigated immediately. If the machine is a Windows box, a recent virus/worm scanner is a must.

192.168.224.150:

This IP is behaving mostly the same. Massive scans of port 139 on 192.168.x.x from `Nov 2 16:13:52` to `Nov 2 16:18:57`, scanning 2980 hosts.

Events occurring around this time are:

```
Oct 29 10:39:01 63.88.175.201:3901 -> 192.168.224.150:21 SYN **S*****
Nov  1 05:39:26 24.115.67.96:1653 -> 192.168.224.150:27374 SYN **S*****
```

Similar actions should be taken as above. An exploit for 21 that snort didn't pick up could have compromised this box, or this box may be commandeered with SubSeven.

- **SNMP public access**

This signature generated 468 alerts, and all came from and to the 192.168.x.x address space (23 srcs and 1 dst of 192.168.101.192). The SNMP string of "public" should be changed to some hard-to-guess word. If anyone should connect to any SNMP enabled machine and use the public string, then they would be able to get a great amount of recon info with very little effort. Likewise, this port should be blocked in and out at the firewall. If 192.168.101.192 shouldn't be doing SNMP things, then this warrants some investigation. http://www.sans.org/topten.htm gives some information about SNMP and how to secure it. There is a good article at http://www.sans.org/newlook/resources/IDFAQ/SNMP.htm called "Using SNMP for Reconnaissance" that details the many ways an attacker can utilize SNMP public/private strings. Also, Lenny Zeltser makes note of the same 192.168.101.192 IP doing SNMP traffic in his practical at http://www.sans.org/y2k/practical/Lenny_Zeltser.htm. His practical is dated 15 August 2000, so apparently the advice of changing the community string from "public" to something else has fallen on deaf ears.

- **Back Orifice**

Information on this can be found at http://www.networkice.com/advice/Phauna/RATs/Back_Orifice/default.htm and at http://www.sans.org/infosecFAQ/malicious/back_orifice.htm. The large number of hosts scanned makes it impossible to paste them all here; it would make this document insanely huge. An overview of the scanning src hosts show that these scans were sequential increments of each octet. Thus, it doesn't appear that anyone came back later. Still, it would be wise to scan the network on port 31337 looking for open ports. A look at the current ruleset I have for snort shows that all rules related to back orifice are content based. However, I do not have a rule named "Back Orifice" as in your ruleset, so what your particular signature alerts on is unknown. If rules with content searching are also within your rule base, then there hasn't been any successful SubSeven action going on, as none of those rules alerted. An example of a content-based rule for back orifice can be found at http://www.sans.org/y2k/practical/Marc_Bayerkohler_GCIA.html.

- **Broadcast Ping to subnet 70**

  There are several things associated with this alert that immediately come to mind. First, this could have been an attacker gathering network information such as which hosts are alive. Sending one ping to the broadcast network, all (Unix) machines will respond with an echo reply. Secondly, this subnet could have been used in a denial of service attack against others on the internet. Similarly to how hosts will respond to a broadcast ping, a ping that is a spoofed via an attacker to your subnet makes the machines on the 192.168.70.x subnet reply to the spoofed person (victim). This is commonly called a Smurf. Information on that can be found at www.cert.org/advisories/CA-1998-01.html. If the network is responding to pings directed at broadcasts, this should be immediately disabled by blocking ICMP to broadcast and network addresses, or disabling "IP-directed broadcasts" at your router.

- **TCP SMTP Source Port traffic**

  As mentioned earlier, many people scan from well known low ports like 80 and 53 because those are typically allowed into one's network. This is more of the same, except the source port is 25. Many scanners/people typically scan from the same src port as the port they're scanning. One example would be a scan from port 25 to port 25. This isn't naturally occurring in the wild, so this is a dead give away that screams scan. These scans alerted here are all of src port 25 to dst port 25. Any mail servers active on the network need to be checked for allowing anyone to relay mail through them. Getting put on SPAM lists isn't a Good Thing. Information on preventing SPAM/relaying can be found at www.sendmail.org and http://mail-abuse.org.There are also a number of SMTP related problems. Going to http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= and dropping keywords like "sendmail" or "smtp" reveals a number of results.

- **WinGate 1080 Attempt**

  Without actually seeing the snort ruleset used, it's impossible to determine at this point whether these attempts yielded anything useful to the prober and how many are false positives.

  Looking at the current rule set that I have, I see the following:

  ```
  alert tcp  !53 ->  1080 (msg:"MISC-WinGate-1080-Attempt";flags:S;)
  ```

  Which would probably false alert a good many times. Think of a passive ftp session where 1080 is picked, or another type of transaction that would pick this ephemeral port.

  Based on the sheer number of alerts, the majority of these alerts are likely to be real scans.

The top 5 are listed below.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total))
63.193.210.208	1883	1883	1837	1837
208.194.161.155	222	222	104	104
198.63.2.192	179	179	9	9
204.117.70.5	157	157	36	36
64.86.5.250	137	137	68	68

63.193.210.208 - adsl-63-193-210-208.dsl.snfc21.pacbell.net:

This host was rather noisy, scanning a large portion of the network. So many IPs/ports in such a short time is unlikely to be a false positive. This scan started at `10/05-18:58:22` and ended at `10/05-19:03:42.` Sending a note to pacbell about this large scan is a good idea. In my dealings with pacbell on such things they've been quick to respond, communicate the problem with their user, and give them time to "straighten up" before canceling the account.

208.194.161.155 - proxy.monitor.twisted.ma.us.dal.net:
This is a real positive, but also false. This network has IRC users (again, this is a site policy issue). The particular dal.net server is checking if the connecting user has an open socks port. Abusers of IRC will use such proxies to hide behind and flood users, take over channels, etc. Checking for the existence of such an open proxy at the user's connection attempt to the IRC network is a quick way to stop this type of abuse before it happens.

198.63.2.192 – UNKNOWN (Verio, Inc. (NET-VRIO-198-063))

Verio looks like a "business to business" company, perhaps 198.63.2.192 is a customer of theirs that is misbehaving or has been compromised. What's interesting is that this IP connects to 9 different hosts, then focuses on 192.168.206.118. It makes approximately 170 connects from `11/19-06:57:36` to `11/19-15:38:03.` That ought to be looked into quickly for what they were doing. If it turns out to be abusive/unauthorized, dropping a note to Verio letting them know of a potential cracked into or abusive client would be a good idea.

204.117.70.5 - security.enterthegame.com :

This is pretty much the same as the dal.net alert. Their web page reads :

When you connect to ETG, our Security server will check its database of known clean connections. If you¹ve connected on a clean (secure) connect before, you will have a "lease" in the database. These are good for a specified period of time from the initial connect. If Security detects a connect that it either has not scanned before or a potentially unsecure connect, it will attempt to connect directly to you, in order to make sure you¹re not operating off an unlimited wingate or an unsecure proxy or bounce. Security will check two ports on your machine - port 23 (telnet port, checks for a wingate telnet bounce) and port 1080 (socks/wingate port) for an unsecured SOCKS4 and SOCKS5 proxy. If it finds a wingate telnet bounce on *port 23 or if it finds an unsecured SOCKS4 or SOCKS5 Proxy, you will be killed for an unsecure connect.*

This IP can also be found in a "detects analyzed" post at http://www.sans.org/y2k/072100.htm, although there's really nothing much said about it other than it being included in a list of other IPs.

64.86.5.250 - proxy3.monitor.dal.net:

And once again, a dal.net check is performed on connecting IRC users.

As with all of the alerts, since real scans have been done it would be wise to re-evaluate what machines have SOCKS proxies and verify they are not open to the world. Having "bad guys" parading around on the net from 192.168.x.x is a Bad Thing. Others, such as Eric Hacker (http://www.sans.org/y2k/practical/Eric_Hacker.html), have noticed similar scans and also came to my conclusion that since there is no content analysis it is impossible to determine if actual WinGate/SOCKS activity has occurred.

- **Watchlist 000222 NET-NCFC**

This alert group has 45 sources and 26 destinations and none of the sources are from the 192.168.x.x network. Since I don't have access to the snort rulebase that was used, I can only assume that this rule matches on any connection from 159.226.x.x, as that's what appears in the list of alerts produced by snortsnarf.

Registry information for this netblock is

```
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
```

A few statistics for the highest number of connections **TO** 192.168.x.x:

# connects	HOST:PORT	# connects	PORT	# connects	HOST

5723	192.168.6.7:25	7842	25	5801	192.168.6.7
1251	192.168.100.230:25	108	113	1299	192.168.100.230
457	192.168.253.43:25	70	40627	461	192.168.253.43
183	192.168.253.41:25	53	43879	186	192.168.253.41
152	192.168.253.42:25	14	23	155	192.168.253.42
70	192.168.99.51:40627	11	443		
57	192.168.6.7:113	10	25459		
53	192.168.100.81:43879				
37	192.168.100.230:113				
32	192.168.145.9:25				
14	192.168.6.7:23				
13	192.168.6.34:25				
11	192.168.6.47:25				

It would seem that port 25 is a popular target. The mail servers should be checked for open relays (and software versions as well, as noted earlier about SMTP exploits). Interestingly, 113 has been connected to 108 times. That's a bit scary in and of itself. The majority of these connects are going to 192.168.6.7 with 57. Based on how ident works, this suggests that 192.168.6.7 is connecting out to various 159.226.x.x addresses. The ports that it is connecting to are requesting who the connection is being made as, thus sending that request to 192.168.6.7's identd daemon. Other IPs that have activity on identd are 192.168.100.230 at 37, 192.168.253.53 at 8, 192.168.253.52 at 2 and 192.168.253.51 at 4. The activities of these machines should be watched, as it is probable they are participating in activities that they shouldn't be. It is also possible that the attacker is using the ident protocol to figure out what processes are running as who.

Disturbingly, there are also 103 connects from port 23 from 159.226.x.x, as well as 6 from 21. Ick, if my idea of how the snort signature looks is correct, then those incoming packets from those ports probably have the SYN+ACK bit set so 192.168.x.x has been connecting out to places it probably shouldn't be. There is unfortunately no data in the snort dumps to verify what TCP flags are set. So, what we have here is:

159.226.22.55 was ftp'd to three times by 192.168.130.185.
159.226.113.1 was ftp'd to three times by 192.168.154.27.
159.226.41.166 was telneted to seventy times by 192.168.99.51.
159.226.41.166 was telneted to fifty-three times by 192.168.100.81.

It's also interesting that 192.167.6.7 was telneted to 14 times. I'd bet my quarter on these machines being compromised. There are also a large number of connects to a couple IPs on some high numbered ports (from the charts above). This should be investigated to see what is listening, if anything.

- **Watchlist 000220 IL-ISDNNET-990517**

This alert group has significantly more destinations than the previous one. Here we have 61 sources and 108 destinations, according to snortsnarf. Also, I'm assuming the same sort of snort rule as defined above, so it looks like this rule will be alerting on traffic from 212.179.x.x. This is registered to

```
inetnum:    212.179.0.0 - 212.179.1.255
netname:    AREL-NET
descr:      arel-net
country:    IL
admin-c:    TP1233-RIPE
tech-c:     TP1233-RIPE
status:     ASSIGNED PA
notify:     hostmaster@isdn.net.il
changed:    hostmaster@isdn.net.il 19990624
source:     RIPE
```

IL is Israel. The list below is again, the higher numbers of connections **TO** 192.168.x.x.

# connects	HOST:PORT	# connects	PORT	# connects	HOST
4810	192.168.211.146:4922	9692	6699	4810	192.168.211.146
3938	192.168.223.98:6699	5733	4619	3938	192.168.223.98
3913	192.168.206.90:4619	4811	4922	3916	192.168.206.90
1638	192.168.203.142:4619	3255	6688	1638	192.168.203.142
1459	192.168.218.142:4990	1459	4990	1459	192.168.218.142
1353	192.168.214.170:6699	648	1069		
950	192.168.202.22:6699	625	1255		
796	192.168.201.174:6688	579	1476		
667	192.168.214.74:6688	437	6346		
648	192.168.209.106:1069	418	4968		
625	192.168.223.254:1255	366	6700		
609	192.168.211.178:6699	308	25		
589	192.168.221.146:6699				
579	192.168.15.215:1476				
564	192.168.227.190:6699				
505	192.168.203.206:6688				

From the looks of things, the majority of traffic appears to be Napster related (port 6688 and 6699) which is again, a site policy thing. There seem to be a good deal of unknown ports such as 4619, 4922, etc. Checking with http://www.simovits.com/nyheter9902.html, only one port, 1255, is used for a trojan (Scarab). Without further packet data, there really is no good way to determine what this is. Since there are large amounts of Napster traffic going on, I tried a little experiment. Using Napster and starting some downloads I noticed traffic like:

Remote:6688 -> me:3184
Remote:7847 -> me:3193

So, it looks like this odd port traffic above could very well be related to Napster. Still, further packet analysis is the best option to be sure. There isn't any indication of what this might be in the packet dumps provided with these snort logs unfortunately.

The high volume of SMTP traffic is a concern as well

141 connections to 192.168.253.41:25
124 connections to 192.168.253.43:25
25 connections to 192.168.6.47:25
11 connections to 192.168.253.42:25
7 connections to 192.168.110.150:25

These servers should be checked in the same way that the SMTP servers were checked above. Outgoing traffic to this 212.179.x.x network seems to be like the incoming, i.e.: seemingly odd ports. Since there seems to be large amounts of incoming traffic to a large number of odd ports, I recommend installing a decent (preferably stateful) firewall and to be on the safe side, scanning the network for trojans.

Summary:

Based on the aggregate data, there were many things that should be looked into immediately. There are a couple machines that are compromised for sure, and several more that have a higher than average probability. A firewall should be put up and configured with a default deny statement, as well as configuring the router/firewall to reject odd fragments and broadcast traffic. Machines that are compromised should be taken off-line, reinstalled, patched and brought back on line. This should be done at the same time, no sense in putting a newly installed machine back on the network when other cracked machines are still lying around. Machines that are compromised are:

192.168.179.78. This was attempting to connect to lpd on remote sites after an alert was triggered via an RPC highport rule earlier.

192.168.221.82, 192.168.224.150. These were doing massive NetBIOS scans. There is something definitely fishy with that.

192.168.130.185, 192.168.154.27, 192.168.99.51, 192.168.100.81. These machines were communication via ftp and telnet to the Watchlist 000222 addresses. If there is no reason for this, these machines have been compromised. Also remember that 192.168.6.7 was communicating a lot with something on the remote hosts that was requesting identd information from it. This IP was also telneted to many times from the remote hosts.

Machines that could be compromised:
192.168.221.82, 192.168.205.94, 192.168.97.206, 192.168.100.209, 192.168.130.242, 192.168.205.94, 192.168.99.130, 192.168.130.81. In other words, those machines subjected to the wu-ftpd site exec exploit.

192.168.6.15, after the RPC call, it was connected to on an RPC port. 192.168.100.130, 192.168.15.127 were also probed with an RPC call, but didn't appear to have any other connections afterwards. They should be investigated anyway to be safe.
Other RPC high port access machines that should be checked include 192.168.87.59, 192.168.253.114, 192.168.179.78, 192.168.140.51, and 192.168.206.218. Keep in mind the signature for these alerts was (according to my snort rules here) match on connections to 32771, so there may be false positives here.

There are also the recipients of the Happy99 virus that should be looked at. If network usage should not be consumed by chat protocols and Mp3 trading protocols, this needs to be addressed via a policy and firewalling. Other various high port connections should be looked into via a packet sniffer or something similar to better accurately determine what it is.

I would also in the future suggest using a tightly secured, central syslogd server for log verification and the use of a host based ID, such as Tripwire (www.tripwire.org) Access lists on such publicly available services (like SOCKS, mail relay) should be checked for sanity.

Assignment 3 - Analysis Process, (30 Points)

After downloading the three sets of snort files, snorta.zip, snorts.zip and OOS.zip I created three directories for each set, "snorta" for alerts, "snorts" for scans and "OOS" for the packet dumps. This left me with a parent directory of "snort_data" with the aforementioned directories contained within. This parent directory is where I used most of my command line tools. I figured as I moved along looking through snort logs I was going to forget what I had done. So, my solution to this was to use a tool called "script." Script will record to a file all input and output that traverses a terminal. Now at assignment #3, all I really have to do is look over this "log file" to know what I've done, what command line tools were used, and what the output was.

At first glance of the files I was a bit confused on the file naming conventions (such as snorta54.txt), I needed something a little easier on the eye. In order to rectify this, I decided that using the dates contained inside the logs would be a good way to categorize the files. So, the following command line would accomplish this (use color=no for GNU ls, otherwise you'll get crazy ANSI characters and nothing will work).

```
for i in `ls --color=no *.txt`; do mv -v $i `tail -n 2 $i|head -1|cut -f 1 -d\-|sed s\|/\|\|`-$i; done
```

The preceding line lists the .txt files and cuts out a line of each file. I used "2" instead of "1" so that tail will get the last two lines of text and not risk get a blank line or new line out of the logs. The line returned would be something like

```
11/20-01:16:14.001989  [**] SMB Name Wildcard [**] 141.157.99.21:137 ->
MY.NET.6.15:137
```

It focuses on "11/20" (see line above) because I used field 1 for cut, cutting on the – character. The / is replaced with nothing and then renames the file with the date of 1120 prefixed in front. Hence, files are now named like 1120-snorta54.txt. Now tail a file to be sure I have the right dates corresponding with the file name:

```
11/20-22:37:00.425112  [**] spp_portscan: portscan status from 24.200.168.131: 1
connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
11/20-22:37:01.896124  [**] spp_portscan: End of portscan from 24.200.168.131 (TOTAL
HOSTS:1 TCP:1 UDP:0) [**]
11/20-23:03:12.739585  [**] SUNRPC highport access! [**] 216.10.12.30:2078 ->
MY.NET.206.222:32771
```

MY.NET needs to be digits so that snortsnarf won't be confused:

```
for i in `ls --color=no *.txt`; do cat $i | sed 's/MY.NET/192.168/g' >$i.ips; done
tail -n 6 1104-snorta43.txt.ips
```

This is mostly the same as above. The only real difference is that each file, one by one, is cat'd and piped through sed, which replaces MY.NET with 192.168 and writes out to a new file of the same name with a ".ips" extension.

Now files named like 1120-snorta54.txt.ips have 192.168.x.x addresses. Now I did another tail to verify the operation worked, and removed the old .txt files.

```
11/04-23:27:55.495339  [**] WinGate 1080 Attempt [**] 207.114.4.46:1753 ->
192.168.217.42:1080
11/04-23:40:38.961739  [**] Attempted Sun RPC high port access [**] 205.188.153.108:4000 -
> 192.168.221.246:32771
```

Before starting any analysis, I just thumbed through the data by hand to get a feel of what is to come. After I felt comfortable with what I was looking at, and had an idea of who was doing the most talking, I identified possibly problematic ports where intruders could have come in on. It was now time to look at something a little more tangible. Since I'm already familiar with snortsnarf I figured that'd be a good place to start. Unfortunately, running snortsnarf on all of these files caused my box to run out of memory. So, I did another "for i in" loop like above, but substituting the command to be run as snortsnarf on each file individually. This proved semi useful to me as it broke out the data for each day into its own directory structure. I also wanted something that could aggregate the data so I could have a bird's eye view, which is easier for me to see what's going on "from a distance." So, I copied the ".ips" files to a Sun Ultra10 and ran snortsnarf from there. After a long time crunching, and no “memory exhausted” errors, I got my results without any problems. After looking over the snortsnarf info, I began the analysis.

Since snortsnarf had already made a pretty listing of what alerts occurred, I didn’t need to do that. What I did need to do is make a list of what IPs were most frequent in both src and dst networks, and what ports were connected to most often. This would serve as an overview of potentially suspicious activity and something to keep an eye out for. As used above, similar command line tool methods were applied here. All that is really required to do this is to break out the snort alert logs (or any snort log for that matter) into easily parseable fields. As an example, to get a list of dst IPs, the line,

```
09/26-00:00:52.873106  [**] WinGate 1080 Attempt [**] 208.194.161.155:3718 ->
192.168.97.206:1080
```

can be broken into two parts with the > character (and again with the : char to break out hosts and ports):

```
find ./snorta -type f -exec grep "\[\*\*\]" {} \;|grep -v portscan|cut -d \> -f 2|cut -d : -f
1|sort|uniq -c|sort -r|less
```

Now I have looked at the alerts and uniq will count the number of occurrences.
Sort -r will list them in descending order. Typical output would be something like

```
5808  192.168.6.7
4814  192.168.211.146
3940  192.168.223.98
3918  192.168.206.90
```

I then proceeded to get a list of ports in much the same way, but using the second field in the second “cut.” I continued on this type of trend listing the most connects to a certain port, src hosts, dst hosts, etc until I had all the statistics I needed before going into analysis with snortsnarf. After this point everything else was mostly a more refined method based on the alert that I was currently looking at with snortsnarf**.** For example, while looking into the “SMB Name Wildcard” alerts, snortsnarf gave a good representation broken down into src and dst IPs. However, there are too many IPs to click on every one. So I used “find” and grepped for the specific alert, breaking up the fields based on hosts to see who the specific hosts were talking to. Then I swapped it up a bit and grepped for hosts that were the chattiest for this alert. This showed everything that IP had done, and since the parent directory I was working from contained all three snort log types, the find+grep looked through everything. If I found something interesting, I could thumb through the specific file by hand. However, in order to know what file the match was found in, I had to add an extra command to “find” after the grep. The command which was used is “-exec ls {} \;” This lists the filename after the match(es). Using “find” helped out a good deal for situations where there were events that didn’t get fed into snortsnarf, such as SYN scan alerts. Knowing that connections to port 137 are often followed by connections to port 139, doing another find+grep and looking for 192\.168\. and :139 helped me to see an odd amount and quickly moving number of connections to NetBIOS ports from a couple machines.

This pattern of systematically going through each alert group and armed with the statistics of "hosts/ports that have large amounts of traffic" from when I first started gathering statistics helped to create what eventually became a breadcrumb trail of IPs. I often found correlation with one IP from an earlier event when looking at the current alert in my list of alerts. There were also at times some activity or packet dumps that I was unclear about. This is where having a few browsers open to places like www.google.com, www.securityfocus.com, and http://cve.mitre.org were a big help answering those lingering questions of "what is this I'm looking at?" Old mailing list archives that are searchable such as Incidents and Vuln-Dev (at securityfocus) were also helpful.