Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)

Practical Assignment
Version 3.4 (revised September 24, 2003)
Thomas Harbour
March 11, 2004

Table of Contents

List of Figures

List of Tables

Part 1 - msdm.exe Trojan Horse and Spamming

1.1 Introduction

This part of the GCIA Practical Assignment will examine how the msdm.exe Trojan horse can appear on a network and what it can be used for. The incident that this part is based on occurred in December 2003. The Trojan was seen again in February 2004 so it's still a current exploit.

1.2 How the Trojan was Detected

On the network being monitored by a variety of sensors, a Cisco IDS sensor started reporting a large number of signature ID 3050 events with a destination of port tcp/25. These events were similar to the following example:

> CSIDS: 4 0 12/10/2003 14:52:42 12/10/2003 9:52:42 10008 102 100 IN OUT 3 **3050** 25 TCP/IP 10.10.64.114 **209.58.237.10** 12041 25 0.0.0.0 0 0 NO DATA NO DATA TCP 1

The reporting sensor was monitoring traffic going out to the Internet from the private network. The source IP address was one of two private IP addresses while the destination IP addresses were invariably legitimate Internet mail exchangers. In the example event shown above, the destination is hawaii.smtp-in.load.com (209.58.237.10), a mail exchanger for load.com.

Signature ID 3050 is a half-open SYN attack signature triggered when multiple TCP sessions have been improperly initiated on any of several well known service ports. In this case the well known service port was smtp (tcp/25).

Since the source IP address was not an authorized mail exchanger, two possibilities were that either the source had some malware or else someone was conducting a denial of service (DoS) attack against Internet mail exchangers. The latter possibility seemed unlikely since there were so many destinations such that the effect of a DoS would have been very diluted.

The desktop support personnel were asked to check the two source IP addresses for Trojans or other malware. They ran Symantec's Norton anti-virus software but found nothing, as well they ran Lavasoft's Ad-Aware software. Ad-aware is a detection and removal utility that scans memory, registry and drives for known Datamining, aggressive advertising, Parasites, Scumware, Keyloggers, selected traditional Trojans, Dialers, Malware, Browser hijackers, and tracking components. Ad-Aware found and removed many items, however signature ID 3050 events continued to be triggered by the same two source IP addresses. The desktop support personnel did not want to spend anymore time examining the two hosts unless it could be shown that the traffic triggering the signature originated from these hosts, e.g. IP address spoofing was not occurring.

To examine the nature of the traffic triggering the signature, iplogging was turned on the IDS sensor for the two source IP addresses, i.e. 10.10.64.114 and 10.10.80.187. The iplogging output files are in the standard tcpdump format and are named in the form of "iplog.<ip_address>.<date-time stamp>.

Running the "*windump -ne -r iplog.10.10.64.114.200312040301*", we see the MAC addresses of the traffic as seen by the IDS.

```
03:06:29.000000 0:2:17:fc:40:0 0:6:d7:3:17:81 0800 62: IP 10.10.64.114.4310 > 61.172.244.198.1131
```

Both MAC address prefixes, i.e. 000217 and 0006D7 are assigned to Cisco Systems[2]. This agrees with the understanding that the IDS is located on the port of a switch between a Cisco router and a firewall with the traffic between them spanned to it (see Figure 1). Hence the IDS is not on the subnet of any hosts in seen in the capture file.

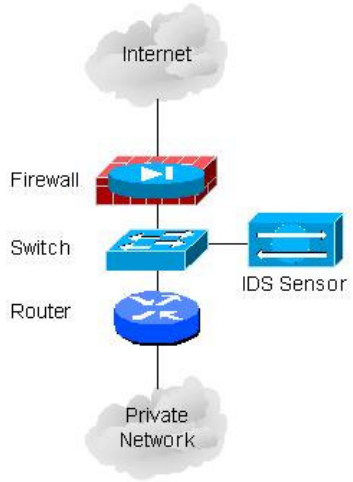

Figure 1. Location of IDS Sensor on Network

Running the "*windump -n -r iplog.10.10.64.114.200312040301*", we can see the following outline of the transaction between the internal host 10.10.64.114 and the external host 61.172.244.198:

1. The host 10.10.64.114 establishes a tcp connection to 61.172.244.198 on tcp/1131:

```
03:06:29.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: S 3499192079:3499192079(0) win 64240 <mss
1460,nop,nop,sackOK>  (DF)
03:06:29.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: S 1814580424:1814580424(0) ack 3499192080 win 65535
<mss  1380,nop,nop,sackOK> (DF)
03:06:29.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: . ack 1 win 64860 (DF)
```

2. Then 7,941 bytes of data are transfer from the destination IP to the internal host and 23 bytes of data in the other direction (in the next section we'll see what this data consists of):

```
03:06:29.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: P 1:10(9) ack 1 win 65535 (DF)
03:06:29.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: . ack 10 win 64851 (DF)
03:06:29.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: P 1:24(23) ack 10 win 64851 (DF)
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: P 10:24(14) ack 24 win 65512 (DF)
```

```
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: P 24:1404(1380) ack 24 win 65512 (DF)
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: P 1404:1414(10) ack 24 win 65512 (DF)
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: P 1414:2794(1380) ack 24 win 65512 (DF)
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: P 2794:2808(14) ack 24 win 65512 (DF)
03:06:30.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: . ack 1414 win 64860 (DF)
03:06:30.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: . ack 2808 win 64860 (DF)
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: P 2808:4150(1342) ack 24 win 65512 (DF)
03:06:30.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: . ack 4150 win 63518 (DF)
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: P 4150:5530(1380) ack 24 win 65512 (DF)
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: P 5530:6910(1380) ack 24 win 65512 (DF)
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: . 6910:6934(24) ack 24 win 65512 (DF)
```

3. The host 61.172.244.198 finishes its data transfer with 10.10.64.114 and then gracefully tears down the tcp connection:

```
03:06:30.000000 IP 61.172.244.198.1131 > 10.10.64.114.3504: FP 6934:7941(1007) ack 24 win 65512 (DF)
03:06:30.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: . ack 6934 win 64860 (DF)
03:06:30.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: . ack 7942 win 63853 (DF)
03:06:30.000000 IP 10.10.64.114.3504 > 61.172.244.198.1131: F 24:24(0) ack 7942 win 63853 (DF)
```

4. Immediately after the tcp connection is torn down, the internal host starts trying to connect to port tcp/25 of various Internet mail exchangers.

```
03:06:30.000000 IP 10.10.64.114.3505 > 209.58.237.10.25: S 3499509159:3499509159(0) win 64240 <mss
1460,nop,nop,sackOK> (DF)
03:06:30.000000 IP 209.58.237.10.25 > 10.10.64.114.3505: R 0:0(0) ack 3499509160 win 64240 <mss
1460,nop,nop,sackOK> (DF)
03:06:30.000000 IP 10.10.64.114.3506 > 209.202.220.99.25: S 3499564050:3499564050(0) win 64240 <mss
1460,nop,nop,sackOK> (DF)
03:06:30.000000 IP 10.10.64.114.3518 > 62.253.162.40.25: S 3499944417:3499944417(0) win 64240 <mss
1460,nop,nop,sackOK> (DF)
03:06:30.000000 IP 62.253.162.40.25 > 10.10.64.114.3518: R 0:0(0) ack 3499944418 win 64240 <mss
1460,nop,nop,sackOK> (DF)
03:06:30.000000 IP 10.10.64.114.3520 > 205.188.158.25.25: S 3500029642:3500029642(0) win 64240 <mss
1460,nop,nop,sackOK> (DF)
03:06:30.000000 IP 205.188.158.25.25 > 10.10.64.114.3520: R 0:0(0) ack 3500029643 win 64240 <mss
1460,nop,nop,sackOK> (DF)
```

Based on these types of traces it was very likely that the traffic triggering the signature originated from reported source IP addresses, i.e. IP address spoofing was not occurring unless very sophisticated hacking was occurring. The desktop support personnel were reengaged but again the anti-virus software found nothing. This time the desktop support person manually examined the locations in the registry and file system where Trojans are typically found and he noticed a folder named *c:\winnt\msdm* that contained a file named *msdm.exe*. Once the *msdm.exe* file and registry references to it were manually removed, the two hosts stopped sending out the previously observed types of traffic.

There is a description of a user having a similar Trojan on his system[3]. He states that "This little thing sent out at least 200 emails every time I logged on. I just hope it wasnt sending any worms or viruses..hopefully , just Spam. Over the past week or so, whenever I log onto the web, hundreds of Emails are being sent from my PC. I purchased Norton 2004 and it found over 100 spybot files. I removed all of them and checked the registry as per the instructions at Symantec's web page. I re-scanned my hard drive and Norton said it was clean. Not so. Once again, the moment I logged onto

web my PC started sending out the mass Emails again. I checked numerous Worms and so far nothing has come up."

Searching for information about msdm.exe on Symantec's site yielded descriptions of a couple a Trojans based on msdm.exe:

1. Backdoor.Armageddon.20 was discovered on January 09, 2003. It is a Backdoor Trojan Horse that opens some randomly changed TCP/UDP ports to connect to the hacker and allows him to remotely control an infected computer.

2. Backdoor.DMSpammer was discovered on October 28, 2003. It is a Backdoor Trojan Horse that relays spam email messages. When Backdoor.DMSpammer is executed, it listens on a (configurable) port for spammers, who can send it a list of addresses as well as what to send.

Based on these descriptions it would appear that the Trojan found on the two hosts was one of these two identified by Symantec. The method by which the hosts became infected is unknown but probably via an e-mail bearing the Trojan as an attachment.

1.3 How the msdm.exe Trojan Horse Works

Trojans do not self-replicate. They are spread manually, often under the premise that the executable is something beneficial. Distribution channels include IRC, peer-to-peer networks, newsgroup postings, email, etc.

The following are the activities that the host with the msdm.exe Trojan Horse was performing on behalf of the individual who planted this Trojan (a verbose trace is available but was not included for reasons of space):

1. The trojaned host contacted its master, IP address 61.172.244.198, on port tcp/1131. The Master and the client then exchange the following login and initial command sequences:

   <from master> 110.DMM
   <from client> LOGIN.test.buffy.1.83
   <from master> 200.SEND.303

2. The master sent the trojaned host a list of e-mail addresses. The following is a sample of part of the list of e-mail addresses downloaded during one session:

   b-falk@t-online.de..b-farias@eudoramail.com..b-faulstich@web.de..b-favelle@shaw.ca..b-fc@kiss.com..b-fd@kiss.com..b-fe@kiss.com..b-fetzer1@ti.com..b-ff@kiss.com..b-fh@kiss.com..b-fhoto@kcn.ne.jp..b-fi@kiss.com..b-file@mailcity.com..b-film-owner@onelist.com

3. The master sent the trojaned host the html-formatted e-mail to send to the list of addresses. The following is one sample e-mail:

<html>....<body.style="font:.normal.12pt.arial,.sans;">....STOP.SPAM.FOREVER!!<p>....Hello,....This.program.worked.for.me..If.you.hate.Spam.like.I.do,.you.owe.it.to.your.self.to...try.this.program,.and.forward.this.email.to.all.of.your.friends.which.also.hate.Spam.or.as.many.people.possible..Together.lets.help.clear.the.Internet.of.Spam!....<p.align=center.style="background:.lightyellow;.border:.1px.solid.yellow;.padding:.3px;"><a.href="http://www.quickeasysolution.com/10th.htm">STOP.SPAM.IN.ITS.TRACKS!..<p>..Ask.yourself.these.3.questions:..

..1)Do.you.get.junk,.scams.and.worse.in.your.inbox...every.day?...
....2)Are.you.sick.of.spending.valuable.time...removing.the.trash?...
..3)Is.your.child.receiving.inappropriate...adult.material?...

..If.so.you.should.know.that.no.other.solution.works...better.then.our.software.to.return.control.of.your...email.back.where.it.belongs!...
..Imagine.being.able.to.read.your.important.email...without.looking.through.all.that.spam....<p>..
....<p.align=center.style="background:.lightyellow;.border:.1px.solid.yellow;.padding:.3px;"><a.href="http://www.quickeasysolution.com/10th.htm">Click.here.to.vist.our.website.......</td></tr></table>....</body>....</html>

4. Immediately following the receipt of the list and e-mail, the trojaned host tried to send out the e-mail to the recipients using Outlook Express to connect to a number of external mail exchangers.

Figure 2 shows what the e-mail message would look to the recipient. The two hyperlinks shown in the image are to http://www.quickeasysolution.com/10th.htm. The purpose of this e-mail is to get the recipient to go to http://www.quickeasysolution.com/10th.htm. This is a web site that sells a product called Email Filter whose purpose is to "STOP SPAM IN ITS TRACKS!" It is ironic that a Trojan horse spammer is use to sell anti-spam software. Of course it would not be surprising if the Email Filter also contains a Trojan.

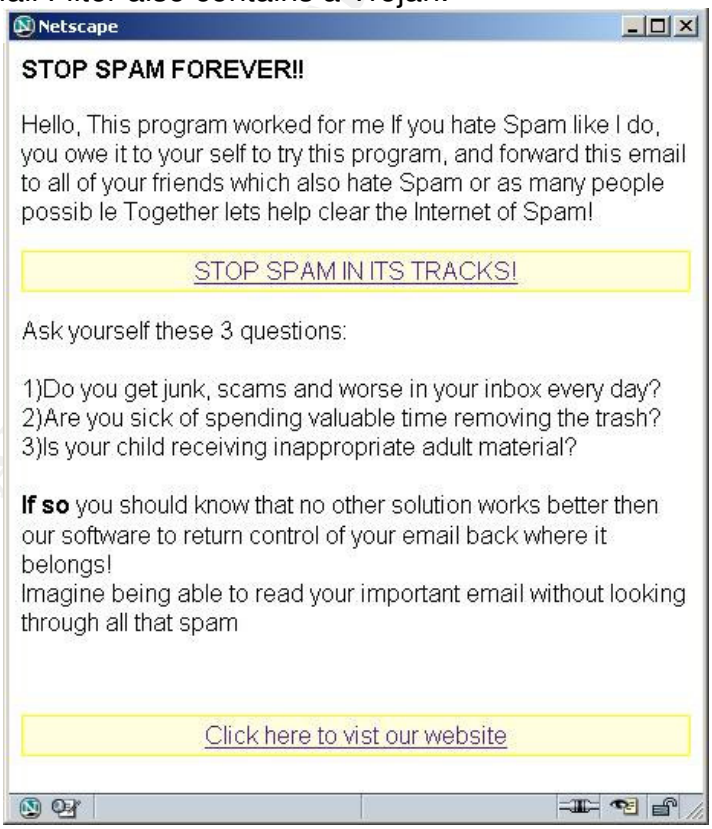

Figure 2. Spam e-mail that Trojan horse tries to send

1.4 About the Master of this Trojaned Host

APNIC[5.] offers the following information on the address block containing the Master of the trojaned host (61.172.244.198). It turns out that both the Master and the web site touted in the e-mail (http://www.quickeasysolution.com/10th.htm) are both in China.

```
inetnum:      61.172.244.0 - 61.172.244.255
netname:      GAMANIA-DIGITAL
descr:        GAMANIA DIGITAL ENTERTAINMENT CO.,LTD
country:      CN
admin-c:      WQ58-AP
tech-c:       WL371-AP
mnt-by:       MAINT-CHINANET-SH
changed:      wanglin@shaidc.com 20030115
status:       ASSIGNED NON-PORTABLE
source:       APNIC

person:       Wang Qing
address:      6F,380 Fushan Road,Shanghai   200122
country:      CN
phone:        +86-21-68761255-807
fax-no:       +86-21-68761255-805
e-mail:       wanglin@shaidc.com
nic-hdl:      WQ58-AP
mnt-by:       MAINT-CN-SHTELE-XINCHAN
changed:      wanglin@shaidc.com 20021007
source:       APNIC
```

1.5 How to Protect against this Exploit

To defend against a hacker trying to implant a Trojan to due his bidding, an organization needs to adopt a Defence in Depth strategy to network security. A defence in depth strategy is the traditional one adopted to afford the defended area the strongest and most resilient protection. In the case of the organization the defended area is the organization's data.

As shown in Figure 3, defense in depth for the organization consists of defensive measures adopted in four layers, namely: network access; the operating system; user applications; and data. At the center of the defended area is the most prized component of the defended area – the organization's data.

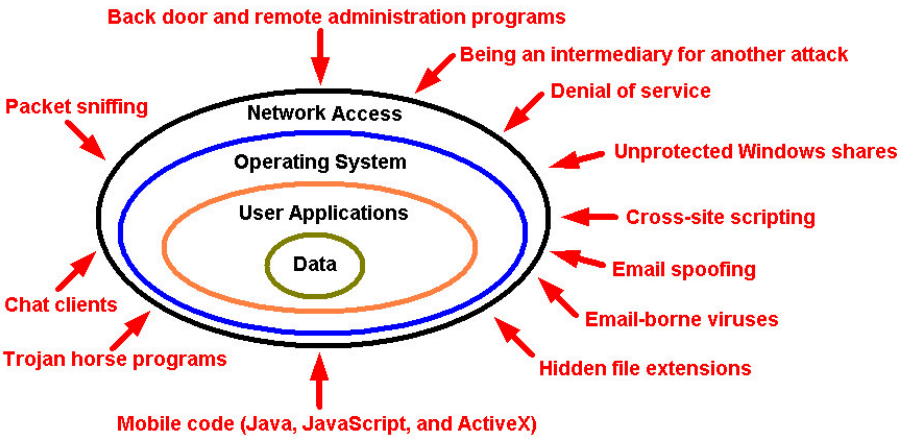

Figure 3. Common Intruder methods used against an organization[4].

This layered approach is required since even the most expensive firewall controlling network access cannot effectively control traffic content. For example, most firewalls will allow in e-mail attachments containing malware. Malware may be cleaned at the operation system layer by anti-virus software if it is recognized. However, if it is of an unknown type, then the final defence is at the data layer where the user opens the e-mail attachment with care.

In this specific case of a Trojan running on hosts inside the network, the following layers of defence are important:

1. Firewall – Ensure that the firewall is designed to control the traffic of both the inside and outside hosts. In particular the traffic from inside users needs to be restricted to that required to support the business. There was no justification for allowing inside users to initiate connections to an outside server on port tcp/1131.

2. Use of IDS – In this case without the alerting from the IDS, it would have been very difficult to detect that inside hosts were running a Trojan. This case serves to highlight the importance using IDS to help understand traffic on a network.

3. Use and keep up to date anti-virus software – A user must prevent intentional intrusions into the computer that take the form of viruses, worms and Trojan horses. An effective approach to defend against this malware is the use of a virus-detection program that is updated regularly and can run in a real-time virus scanning mode. Although in this case it appears that the anti-virus software was up to date but did not detect the Trojan.

4. End user education to open E-mail Attachments with Care – Users should be educated that before opening any email attachments, they must check if they recognize the sender of the attachment and have a good idea of why the attachment is being sent. However, recognizing the sender is not enough since some viruses such as Melissa, sent copies of themselves out as attachment to all addressees found in the Microsoft Outlook address book on the infected system.

A good approach to educate users to follow when opening an attachment is as follows:

1. Check if you recognize the sender of the attachment and know why the attachment is being sent.
2. Be very suspicious of amusing or enticing programs since this type of social engineering is sometimes used by malicious code for its propagation.
3. If you decide open the attachment then ensure that the anti-virus software's virus definitions are up-to-date and then proceed as follows:

    - save the file to your hard disk
    - scan the file using the anti-virus software
    - finally open the file

1.6 References

1. McAfee: Trojan Name: BackDoor-BAM - Method Of Infection, http://vil.nai.com/vil/content/v_100747.htm

2. [Ntop-dev] mac address prefixes in use, URL: http://lists.ntop.org/pipermail/ntop-dev/2003-March/002310.html, Fri, 7 Mar 2003

3. Posting by bogart69, URL:http://forums.spywareinfo.com/index.php?s=f5bcdae5bf7c83f7bec2b57ee600782c&showforum=27, Posted: Sep 19 2003, 07:45 AM

4. GSEC Practical Assignment, Defence in Depth on the Home Front by Thomas Harbour, April 3, 2003

5. APNIC Whois Database, http://www.apnic.net/apnic-bin/whois.pl

Part 2 - Network Detects

2.1 Network Detect #1 – Various Web Server Exploits

2.1.1 Snort Alerts

Running new versions of Snort, e.g. v2.1.0, against the source file named 2002.8.28 generated no alerts. However, using Snort 1.9.1 the 12 alerts shown Table 1 were generated. These alerts were only of the following types:

```
[**] [1:620:2] SCAN Proxy (8080) attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/28-09:17:16.346507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E
12.83.110.10:2790 -> 115.74.34.242:8080 TCP TTL:111 TOS:0x0 ID:40608 IpLen:20 DgmLen:48 DF******S* Seq:
0x379862B0  Ack: 0x0  Win: 0x16D0  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

[**] [1:618:2] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/28-10:35:01.766507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E
65.169.47.30:4522 -> 115.74.172.117:3128 TCP TTL:111 TOS:0x0 ID:9644 IpLen:20 DgmLen:48 DF******S* Seq:
0x45B326E9  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

Group	Extracts of "SCAN Proxy (8080) attempt" Alerts	Time Dif
1a	09/28-09:17:16.346507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 12.83.110.10:2790 -> 115.74.34.242:80 TTL:111 DF******S* Seq: 0x379862B0	0 sec
1b	09/28-09:17:19.286507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 12.83.110.10:2790 -> 115.74.34.242:80 TTL:111 DF******S* Seq: 0x379862B0	3 sec
1c	09/28-09:17:25.256507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 12.83.110.10:2790 -> 115.74.34.242:80 TTL:111 DF******S* Seq: 0x379862B0	6 sec
2a	09/28-10:21:51.096507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 12.83.110.10:3075 -> 115.74.165.99:8080 TTL:110 DF******S* Seq: 0xBAF2BD8	0 sec
2b	09/28-10:21:54.126507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 12.83.110.10:3075 -> 115.74.165.99:8080 TTL:110 DF******S* Seq: 0xBAF2BD88	3 sec
2c	09/28-10:22:00.156507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 12.83.110.10:3075 -> 115.74.165.99:8080 TTL:110 DF******S* Seq: 0xBAF2BD88	6 sec
3a	09/28-10:34:40.746507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 65.169.47.30:4485 -> 115.74.172.117:8080 TTL:111 DF******S* Seq: 0x4547AEAE	0 sec
3b	09/28-10:34:43.746507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 65.169.47.30:4485 -> 115.74.172.117:8080 TTL:111 DF******S* Seq: 0x4547AEAE	3 sec
3c	09/28-10:34:49.746507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 65.169.47.30:4485 -> 115.74.172.117:8080 TTL:111 DF******S* Seq: 0x4547AEAE	6 sec
#	**Extracts of "SCAN Squid Proxy attempt" Alerts**	
4a	09/28-10:35:01.766507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 65.169.47.30:4522 -> 115.74.172.117:3128 TTL:111 DF******S* Seq: 0x45B326E9	0 sec
4b	09/28-10:35:04.806507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 65.169.47.30:4522 -> 115.74.172.117:3128 TTL:111 DF******S* Seq: 0x45B326E9	3 sec
4c	09/28-10:35:10.756507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3E 65.169.47.30:4522 -> 115.74.172.117:3128 TTL:111 DF******S* Seq: 0x45B326E9	6 sec

Table 1. SCAN Proxy (8080) and SCAN Squid Proxy attempt Alerts

Extracts from these 12 alerts are found in Table 1. They were divided into groups based on source/destination IP address pairs and sequence numbers. The "Time Dif" column shows that the connection attempts are waiting the standard delay intervals

between TCP connection attempts. It appears that the connection attempts were unsuccessful.

The two apparent source addresses involved are 12.83.110.10 and 65.169.47.30, while the destination addresses are 115.74.34.242, 115.74.165.99 and 115.74.172.117.

Both the SCAN Proxy (8080)[27.] and SCAN Squid Proxy[28.] attempts can be a prelude to an attack. Hence we need to look for other events concerning the attacking IP addresses. The two apparent source addresses involved SCAN Proxy (8080) and SCAN Squid Proxy, i.e. 12.83.110.10 and 65.169.47.30, are not seen in the tcpdump files associated with any other events. Furthermore, there is no additional traffic seen to the destination IP addresses of 115.74.34.242, 115.74.165.99 or 115.74.172.117. Therefore there was no immediate follow up of this apparent reconnaissance activity and so we'll look for other suspicious activity. In fact we'll look at the activity directed against the web server with an IP address of 115.74.249.202 for reasons that are mentioned later.

2.1.2 Source of Trace

The source of this detect is a file named 2002.8.28 that is found on http://www.incidents.org/logs/Raw/. It is 105,408 bytes and dated Wed Oct 9 12:22:12 2002. The details from the associated README file are found in Section 2.1.3.

The network architecture associated with this detect is unknown but running the "*windump -ne -r 2002.8.28.detect*" command, we see the MAC addresses of the traffic as seen by the IDS:

```
19:45:16.696507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 0800 570: IP 115.74.249.65.62347 >
216.239.51.101.80: P 2743256814:2743257330(516) ack 3180748922 win 64240 (DF)
```

Both MAC address prefixes, i.e. 00000C and 0003E3 are assigned to Cisco Systems according to the IEEE Organizationally Unique Identifier (OUI) listing[25]. Looking at the flow of the traffic, we see that:

1. The Cisco device with a MAC address of 0:3:e3:d9:26:c0 is upstream from that with a MAC address of 0:0:c:4:b2:33

2. Host 115.74.249.65 is running a Microsoft Internet Explorer web browser:

```
19:45:16.696507 IP 115.74.249.65.62347 > 216.239.51.101.80 .. User-
Agent:.Mozilla/4.0.(compatible;.MSIE.6.0;.Windows.NT.5.1)
```

3. Host 115.74.249.202 likely has a web server running on port tcp/80:

```
05:49:22.206507 IP (tos 0x0, ttl 63, id 19439, len 576) 115.74.249.202.80 >
195.29.132.167.1425
```

4. Hosts 115.74.34.242, 115.74.165.99 and 115.74.172.117 may have a web server running on port tcp/8080:

09:17:16.346507 IP 12.83.110.10.2790 > 115.74.34.242.8080
10:21:54.126507 IP 12.83.110.10.3075 > 115.74.165.99.8080
10:34:40.746507 IP 65.169.47.30.4485 > 115.74.172.117.8080

Taking all this into consideration, it is likely that the Snort IDS is located either on a tap or on a port of a switch between two Cisco router/firewall devices with the traffic between them spanned to it as shown in Figure 4. Hence the IDS is not on the subnet of any hosts in seen in the detect file.

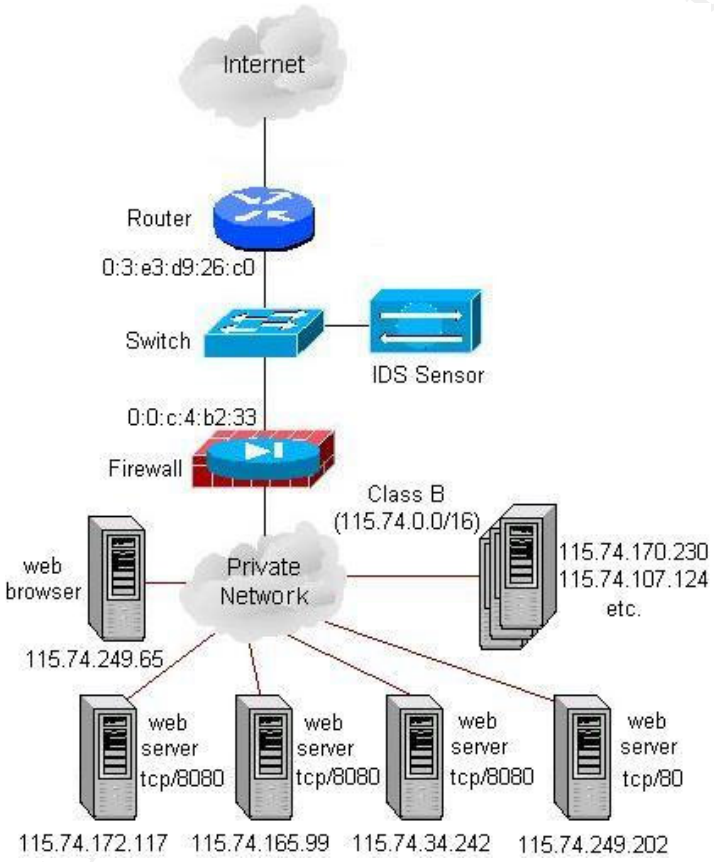

Figure 4. Representative Locations of hosts and IDS Sensor on Network #1

2.1.3 Detect was generated by?

The README file on the www.incidents.org site states that:

> "The log files are the result of a Snort instance running in binary logging mode. This means that only the packets that violate the ruleset will appear in the log. The logs themselves have been sanitized. All of the IP addresses of the protected network space have been "munged". Additionally, the checksums have been modified to prevent clever people from discovering the original IP addresses. You will find that certain keywords within the packets have been replaced with "X"s. All ICMP, DNS, SMTP and Web traffic has also been removed.

A common question is, "Are the addresses changed in the same way across all of the files?" The answer is both yes and no. If you look at the timestamp associated with the files on the website, you will see that groups of files have been posted on the same day. Files posted on the same day will have the IP addresses of the protected network modified consistently. IP addresses belonging to non-local hosts are the actual IP addresses and will be consistent across all log files regardless of date."

The detect file containing the logged packets is in the standard tcpdump binary format. So any program capable of reading this format can be used against the detect file, e.g. tcpdump/windump, snort or ethereal.

Running "*windump -nXvv -r 2002.8.28*" we see that

```
11:29:09.566507 IP 24.189.224.108.2956 > 115.74.249.202.80: GET./cgi/FormMail.cgi?
```

As stated earlier, we'll look at the activity directed against the web server with an IP address of 115.74.249.202.

2.1.3.1 Detect of Traffic to and from Inside IP Addresses

According to the README file, all of the IP addresses of the protected network space have been "munged". In this detect file all the inside IP addresses have been munged into the Class B network of 115.74.0.0/16.

In the following sub-sections, we will examine the detects by the IP addresses in the Class B network. The basic information on the inbound packets found in the detect file are shown in Table 2.

Source IP	Destination IP	Destination Port	TTL	Packet Count	Suspicious?
12.83.110.10	115.74.34.242	tcp/8080	111	3	recon
	115.74.165.99		110	3	recon
65.169.47.30	115.74.172.117	tcp/8080	111	3	recon
		tcp/3128	111	3	recon
255.255.255.255	115.74.170.230	tcp/515	15	1	Yes
255.255.255.255	115.74.107.124	tcp/515	15	1	Yes
219.165.155.85	115.74.249.202	tcp/80	39,40	12	No
213.73.200.122 (qn-213-73-200-122.quicknet.nl)			112	7	"WEB-MISC WebDAV propfind access"
4.63.173.119 (tamqfl1-ar2-4-63-173-119.tamqfl1.dsl-verizon.net)			109	9	"WEB-CGI formmail"
218.145.25.59 (Korea Telecom)			49	1	"WEB-IIS _vti_inf access"

Source IP	Destination IP	Destination Port	TTL	Packet Count	Suspicious?
218.145.25.52 (Korea Telecom)			49	1	"WEB-FRONTPAGE shtml.exe access"
217.39.87.11			46	1	No
213.44.187.50 (l06m-4-50.d2.club-internet.fr)			105	2	"WEB-FRONTPAGE shtml.exe access" and "WEB-IIS _vti_inf access"
200.249.46.195 (Comite Gestor da Internet no Brasil)			43	4	"WEB-CGI formmail"
194.230.222.228			113	1	No
193.188.94.2 (NIC Jordan)			233	1	"WEB-CGI formmail"
24.189.224.108 (Optimum Online)			120	4	"WEB-CGI formmail"
61.193.164.211 (nissan-con.co.jp)			40,231	14	Yes
133.145.228.12 (px3.hitachi.co.jp)			231/2	2	"WEB-IIS _vti_inf access"
63.16.15.140 (UUNET)			115	1	"WEB-CGI formmail"

Table 2. Inbound Packets found in Detect #1 File

The basic information on the outbound packets found in the detect file are shown in Table 3. The fact that 115.74.249.202 generates outbound traffic from a source port of tcp/80 indicated that it is running a publicly accessible web server, apparently an Apache/1.3.12 server on Red.Hat Linux with FrontPage server extensions v4.0.4.3.

```
05:49:22.206507 IP (tos 0x0, ttl 63, id 19439, len 576) 115.74.249.202.80 >
195.29.132.167.1425
HTTP/1.1.403.Forbidden..Date:.Sat,.28.Sep.2002.14:39:20.GMT..Server:.Apache/1.3.12.(Unix)..
(Red.Hat/Linux).FrontPage/4.0.4.3
```

Source IP	Destination IP	Source Port	TTL	Packet Count	Suspicious?
115.74.249.202	195.29.132.167	tcp/80	63	1	No
115.74.249.202	212.62.35.40	tcp/80	63	1	No
Source IP	**Destination IP**	**Destination Port**	**TTL**	**Packet Count**	**Suspicious?**
115.74.249.65	various web sites	tcp/80	122,240		Yes – varying TTL

Table 3. Outbound Packets found in Detect #1 File

Note that the TTL of packets coming from host 115.74.249.65 have two very different values, i.e. 122 or 240 even though they are apparently coming from the same source IP/source port combination and the same destination IP/destination port combination and occur almost simultaneously. This may be due to the way that the protected network space has been "munged" but if not then further investigation for crafted packets is warranted.

```
20:33:44.346507 IP (tos 0x0, ttl 122, id 164, len 646) 115.74.249.65.63212 >
216.239.51.101.80
20:33:44.376507 IP (tos 0x10, ttl 240, id 0, len 1162) 115.74.249.65.63212 >
216.239.51.101.80
```

2.1.3.1.1 Traffic to port tcp/80 on 115.74.249.202

In Table 2 we see that there were 60 packets destined to port tcp/80 on 115.74.249.202. In Section 2.1.3.1 it was established that there is a publicly accessible web server on 115.74.249.202. The following version of snort is available:

```
snort -V

-*> Snort! <*-
Version 2.0.2-ODBC-MySQL-WIN32 (Build 92)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
1.7-WIN32 Port By Michael Davis (mike@datanerds.net, www.datanerds.net/~mik
1.8 - 2.0 WIN32 Port By Chris Reid (chris.reid@codecraftconsultants.com)
```

Since traffic was seen to both ports tcp/80 and tcp/8080 the following preprocessor statement was set in the snort.conf file along with the HOME_NET variable:

```
1. preprocessor http_decode: 80 8080 unicode iis_alt_unicode double_encode
   iis_flip_slash full_whitespace

2. var HOME_NET 115.74.0.0/16
```

Running snort in the IDS mode using the "*snort -c snort.conf -r 2002.8.28 -l detect1 –Xde*" command, we get the following output:

```
Snort processed 167 packets.
Breakdown by protocol:                Action Stats:

    TCP: 167      (100.000%)          ALERTS: 0
    UDP: 0        (0.000%)            LOGGED: 0
   ICMP: 0        (0.000%)            PASSED: 0
    ARP: 0        (0.000%)
  EAPOL: 0        (0.000%)
   IPv6: 0        (0.000%)
    IPX: 0        (0.000%)
  OTHER: 0        (0.000%)

===================================================

TCP Stream Reassembly Stats:
   TCP Packets Used:      0          (0.000%)
   Reconstructed Packets: 0          (0.000%)
   Streams Reconstructed: 0
```

In short, the output of Snort tells us nothing about the detect since Snort finds no tcp streams.

2.1.3.1.1.1 "WEB-CGI formmail" Exploit

Running "*windump -nXvv -r 2002.8.28.detect host 115.74.249.202*" we see the following types of interesting packets involving POSTs to the formmail.pl cgi script:

```
07:19:32.496507 IP (tos 0x0, ttl 43, id 44468, len 552) 200.249.46.195.4985 >
115.74.249.202.80: . [bad tcp cksum f41f (->716)!] 3567267248:3567267760(512) ack 4148661565
win 16384bad cksum bdb (->7b4a)!
E..(....+.......sJ...y.P..-..G.=P.@.....POST./cgi-
bin/formmail.pl.HTTP/1.0..Host:.www.XXXXXXXX..Accept:.image/gif,.image/x-
xbitmap,.image/jpeg,.application/msword,.*/*..Content-Type:.application/x-www-form-
urlencoded..User-Agent:.Mozilla/4.06.(Win95;.I)..Content-
Length:.378....email=pvd39@tct46.com&recipient=afgman@aol.com&subject=www.XXXXXXXX%2Fcgi-
bin%2Fformmail.pl%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20lbbtlvj&=%0D%0A%0D%0At
ime%2Fdate%3A%2011%3A18%3A10am%20%2F%2009%2F28%2F2002%0D%0A<A%20HREF%3D%22www.XXXXXXXX%2Fcgi-
bin%2Fformmail
```

Note: According to the README file on the www.incidents.org site, the "X"s in the data are due to replacement of certain keywords.

Replacing the Unicode characters in the data payload, we get the following:

```
POST /cgi-bin/formmail.pl HTTP/1.0..Host: www.XXXXXXXX..Accept: image/gif, image/x-xbitmap,
image/jpeg, application/msword, */*..Content-Type: application/x-www-form-urlencoded..User-
Agent: Mozilla/4.06 (Win95; I)..Content-Length:
378....email=pvd39@tct46.com&recipient=afgman@aol.com&subject=www.XXXXXXXX/cgi-
bin/formmail.pl                  lbbtlvj&=....time/date: 11:18:10am / 09/28/2002..<A
HREF="www.XXXXXXXX/cgi-bin/formmail
```

The Formmail IDS signatures are found in the web-cgi.rules file. This file contains the following two signatures relevant to Formmail vulnerabilities:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI formmail arbitrary
command execution attempt"; flow:to_server,established; uricontent:"/formmail"; nocase;
content:"%0a"; nocase; reference:nessus,10782; reference:nessus,10076;
reference:bugtraq,1187; reference:cve,CVE-1999-0172; reference:arachnids,226; classtype:web-
application-attack; sid:1610; rev:5;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI formmail access";
flow:to_server,established; uricontent:"/formmail"; nocase; reference:nessus,10782;
reference:nessus,10076; reference:bugtraq,1187; reference:cve,CVE-1999-0172;
reference:arachnids,226; classtype:web-application-activity; sid:884; rev:8;)
```

Looking at the snort\etc\sid-msg.map file, we see the following two signature IDs corresponding to the Formmail IDS signatures:

```
884 || WEB-CGI formmail access || arachnids,226 || cve,CVE-1999-0172 || bugtraq,1187 ||
nessus,10076 || nessus,10782

1610 || WEB-CGI formmail arbitrary command execution attempt || arachnids,226 || cve,CVE-
1999-0172 || bugtraq,1187 || nessus,10076 || nessus,10782
```

There is no message file for the 884 signature but there is one for the 1610 signature. In the 1610.txt file, the attack scenario is that "Formmail receives information from a form via an HTTP POST. This includes the email addresses to which the form data is

sent. A URI in the form of a POST to the formmail script could be crafted to send environment variables to a specified email address."

The additional references about the "WEB-CGI formmail" exploit are:

http://www.whitehats.com/info/IDS/226
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0172
http://www.securityfocus.com/bid/1187
http://cgi.nessus.org/plugins/dump.php3?id=10076
http://cgi.nessus.org/plugins/dump.php3?id=10782

2.1.3.1.1.2 "WEB-FRONTPAGE shtml.exe access" Exploit

As shown in Section 2.1.3.1 the web server has FrontPage Server Extensions v4.0.4.3 so it may be vulnerable to Frontpage exploits such as the following "WEB-FRONTPAGE shtml.exe access" found in the web-cgi.rules file (there is no 962.txt message file):

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-FRONTPAGE shtml.exe
access"; flow:to_server,established; uricontent:"/_vti_bin/shtml.exe"; nocase;
reference:nessus,10405; reference:cve,CAN-2000-0413; reference:cve,CAN-2000-0709;
reference:bugtraq,1608; reference:bugtraq,1174; classtype:web-application-activity; sid:962;
rev:6;)
```

The following is one of eight packets that would be expected to trigger the "WEB-FRONTPAGE shtml.exe access" signature. There are six of these from 61.193.164.211 over the period of 16:30:40.236507 - 16:37:00.126507 on 28 Sep 02:

```
09/28-16:34:17.366507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x213
61.193.164.211:39021 -> 115.74.249.202:80 TCP TTL:231 TOS:0x0 ID:46440 IpLen:20 DgmLen:517 DF
***AP*** Seq: 0x7DF3D391  Ack: 0x26646FB2  Win: 0x2238  TcpLen: 20 POST
/_vti_bin/shtml.exe/_vti_rpc HTTP/1.0..Date: Sun, 29 Sep 2002 02:33:11 GMT..Mime-Version:
1.0..User-Agent: MSFrontPage/4.0..Accept: auth/sicily..Content-Length: 41..Content-Type:
application/x-www-form-urlencoded..X-Vermeer-Content-Type: application/x-www-form-
urlencoded..Pragma: no-cache..Via: 1.1 - (DeleGate/7.9.3), 1.0 px15.hitachi.co.jp:8080
(Squid/2.3.STABLE1)..X-Forwarded-For: unknown..Host: www.XXXXXXXX..Cache-Control: max-
age=259200..Connection: keep-alive....
```

It appears that 61.193.164.211 is a proxy server connecting to the web server 115.74.249.202. The address 61.193.164.211 is assigned to Nissan Rinkai Construction Co.,Ltd. and in fact it's the name server for nissan-con.co.jp. Querying the name servers for that domain, we find that dns1.nissan-con.co.jp is 61.193.164.211. Hence it possible that it has a dual role for nissan-con.co.jp.

2.1.3.1.1.3 "WEB-IIS _vti_inf access" Exploit

One odd thing about host 61.193.164.211 is that as mentioned in Table 2, the TTL associated with it is both 40 and 231 as shown in the following extracts:

```
16:34:16.546507 IP (tos 0x0, ttl 40, id 36134, len 585) 61.193.164.211.63871 >
115.74.249.202.80

16:34:17.366507 IP (tos 0x0, ttl 231, id 46441, len 81) 61.193.164.211.39021 >
115.74.249.202.80
```

Given the very large differences in TTL over such a small period of time, one would suspect crafted packets are being sent from this host. Using windump to dump the packets with a TTL of 40 we find two, one of which is:

```
09/28-16:34:16.546507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x257
61.193.164.211:63871 -> 115.74.249.202:80 TCP TTL:40 TOS:0x0 ID:36134 IpLen:20 DgmLen:585 DF
***AP*** Seq: 0x126CFEA3  Ack: 0x2610A3D1  Win: 0x2238  TcpLen: 32 TCP Options (3) => NOP NOP
TS: 9316442 7698513 GET /_vti_inf.html HTTP/1.0..X-Locking:
133.241.8.2:/var/spool/delegate/cache/http/www.XXXXXXXX/_vti_inf.html..X-Cache-ID:
3d95c5b7/3d966666..Date: Sun, 29 Sep 2002 02:33:10 GMT..Mime-Version: 1.0..Accept: */*..User-
Agent: Mozilla/2.0 (compatible; MS FrontPage 4.0)..Accept: auth/sicily..Content-Length:
0..Pragma: no-cache..Accept-Encoding: identity..Via: 1.1 - (DeleGate/7.9.3), 1.0
px14.hitachi.co.jp:8080 (Squid/2.3.STABLE4)..X-Forwarded-For: unknown..Host:
www.XXXXXXXX..Cache-Control: max-age=259200..Connection: keep-alive...
```

These packets would be expected to trigger the following “WEB-IIS _vti_inf access" signature found in the web-cgi.rules file:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS _vti_inf
access";flow:to_server,established; uricontent:"_vti_inf.html"; nocase; classtype:web-
application-activity; sid:990;  rev:5;)
```

In the 990.txt message file, the attack scenario is that an attacker can craft a URL to access the '_vti_inf.html' file to learn the version and scripting paths of FrontPage. Without the return traffic, we cannot tell if this attempt succeeded.

More information on this exploit is available from http://www.securityfocus.com/bid/1608.

2.1.3.1.1.4 "WEB-MISC WebDAV propfind access" Exploit

As shown in Section 2.1.3.1 the web server is Apache/1.3.12 server running on Redhat so it may be configured to support WebDAV and hence vulnerable to exploits such as the following “WEB-MISC WebDAV propfind access " found in the web-cgi.rules file:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC WebDAV propfind
access"; content:"<a\:propfind"; nocase; content:"xmlns\:a=\"DAV\">"; nocase;
flow:to_server,established; reference:bugtraq,1656; reference:cve,CVE-2000-0869;
classtype:web-application-activity; sid:1079; rev:8;)
```

In the 1079.txt file, the attack scenario is that “An attacker can get a directory listing for all directories configured to support WebDAV in an Apache web server. Certain configurations of Apache, such as those in SuSE 6.0-7.0 and RedHat 6.2-7.0, have WebDAV enabled and misconfigured in such a way to allow directory listings of the entire server file structure.”

Running “*windump -nXvv -r 2002.8.28.detect host 115.74.249.202*” we see the following types of interesting packets involving sending “PROPFIND /main/ HTTP/1.1” to the web server:

```
09/27-21:58:46.526507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xEC
213.73.200.122:11757 -> 115.74.249.202:80 TCP TTL:112 TOS:0x0 ID:5669 IpLen:20 DgmLen:222 DF
```

```
***AP*** Seq: 0xF9A87A2B  Ack: 0xB2337B4F  Win: 0xF88B  TcpLen: 20 PROPFIND /main/
HTTP/1.1..Depth: 0..translate: f..User-Agent: Microsoft-WebDAV-MiniRedir/5.1.2600..Host:
www.XXXXXXXX..Content-Length: 0..Connection: Keep-Alive..Pragma: no-cache....
```

2.1.4 Probability the source address was spoofed

To be successful, both the Formmail and FrontPage exploits require the completion of the TCP three-way handshake hence IP address spoofing is unlikely. As well the hacker needs to see the return traffic from the web server. Although it is possible that the hacker could being use a sniffer or a tap to observe the return traffic, he/she would still need to control the source IP address to initiate the TCP connection and the exploits. This control could be via a number of mechanisms such as a Trojan or by using a proxy server as we saw.

As can be seen in Table 2, most of the source IP addresses are from blocks assigned to ISPs. Hence if they do not belong directly to the hacker, then they could be trojaned.

2.1.5 Description of attack

The description of attack for each of the exploits is found in the sub-sections of Section 2.1.3.1.1 follows:

- "WEB-CGI formmail" Exploit - In this exploit, we see an attempt to use the formmail script in the cgi-bin to mail data to afgman@aol.com for future use.

- "WEB-FRONTPAGE shtml.exe access" Exploit – Certain requests sent to a webserver running FrontPage Server Extensions involving a URL request for a MS-DOS device through "shtml.exe" can cause the server to crash[47]. However in this case we have the request "/_vti_bin/shtml.exe/_vti_rpc" which has been associated with web site defacements since it can allow contents to be posted[48].

- "WEB-IIS _vti_inf access" Exploit – The attacker tried to GET the '_vti_inf' file to learn the version and scripting paths of FrontPage[49].

- "WEB-MISC WebDAV propfind access" Exploit – The attacker repeatedly tries uses the "PROPFIND ./main.HTTP/1.1" request to can get a directory listing for all directories configured to support WebDAV in an Apache web server[50].

2.1.6 Correlations

The number of references about these exploits, e.g. CVE, Snort and Bugtraq, are found in the sub-sections of Section 2.1.3.1.1 and in Section 2.1.5.

Loic Juillard discusses the Spam relay scanning targeted against this web server (http://cert.uni-stuttgart.de/archive/intrusions/2003/08/msg00151.html).

2.1.7 Evidence of active targeting

This web server is being actively targeted by a number of hosts that are listed in Table 2. It is not haphazard traffic, rather it is active targeting since the exploits are targeted against the specific web server and associated extensions. Now given that many of the attacking IP addresses belong to ISPs, the traffic seen could have been coordinated exploits using Trojaned hosts.

2.1.8 Severity

The severity of this attack is determined by the following formula:

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

where each value is ranked on a scale from 1 (lowest) to 5 (highest).

- Criticality (a measure of how critical the targeted system is): The assigned value is a "4" since a public web server is normally an important asset and should not be defaced or made available.

- Lethality (a measure of how severe the damage to the targeted system would be if the attack succeeded): The assigned value is a "4" since these are real potential vulnerabilities for this particular web server.

- System countermeasures (a measure of the strength of the defensive mechanisms in place on the host itself). The assigned value is a "3" since we do not know these vulnerabilities have patches available.

- Network countermeasures (a measure of the strength of the defensive mechanisms in place on the network): The assigned value is a "4" since many of the attempted connection attempts appear not to have been established as they are repeated.

Severity Calculation: 1 = (4 + 4) – (3 + 4)

2.1.9 Defensive recommendation

The web server administrator needs to verify that the web server is patched against the vulnerabilities for the specific web server and associated extensions. As well the web server should be configured not to readily provide the versions of the software that it is running.

2.1.10 Multiple choice test question

Question: Which one of the following URL requests would raise concern that an HTTP client is trying to trivially learn version information about a web server?

A. POST /_vti_bin/shtml.exe

B. POST /cgi-bin/formmail.pl
C. GET /_vti_inf.html HTTP/1.0
D. GET /personal/ HTTP/1.1

Answer: C

Explanation: Answer C is correct because a GET can be used to trivially try to retrieve web server information. Although answer D is a GET, it's highly unlikely that a web server stores its version information in a file named "personal".

2.2 Network Detect #2 - "BAD-TRAFFIC ip reserved bit set" alert

2.2.1 Snort Alerts

Running Snort v2.1.0 against the source file named 2002.9.26 generated 1,139 alerts. The breakdown of these alerts by type is found in Table 4.

Alert	Alert Count
SCAN Squid Proxy attempt	508
SCAN Proxy Port 8080 attempt	508
(http_inspect) BARE BYTE UNICODE ENCODING	78
(http_inspect) NON-RFC HTTP DELIMITER	27
SCAN nmap TCP	9
SCAN SOCKS Proxy attempt	4
(http_inspect) DOUBLE DECODING ATTACK	3
BAD-TRAFFIC ip reserved bit set	1
(http_inspect) IIS UNICODE CODEPOINT ENCODING	1

Table 4. Alerts found in Detect #2 File

Of these alerts, the following one will be examined in detail:

```
[**] [1:523:4] BAD-TRAFFIC ip reserved bit set [**]
[Classification: Misc activity] [Priority: 3]
10/26-01:04:47.966507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
217.85.127.102 -> 32.245.71.165 TCP TTL:244 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0079   Frag Size: 0x0014
```

According to the Snort Signature Database[35], this event is generated when packets on the network have the reserved bit set. This alert may be an indicator of the use of the reserved bit by a malicious user to instigate covert channel communications, an indicator of unauthorized network use, reconnaissance activity or system compromise. These rules may also generate an event due to improperly configured network devices.

2.2.2 Source of Trace

The source of this detect is a file named 2002.9.26 that is found on http://www.incidents.org/logs/Raw/. It is 529,517 bytes and dated Mon Dec 2 15:39:41 2002. The details from the associated README file are found in Section 2.1.3.

To determine what the IDS is monitoring, we run the following commands to see first the source MAC addresses and then the destination ones in the traffic as seen by the IDS:

tcpdump -ne -r iplog.MY.NET.101.21.200402091650.6 | awk '{print $2}' | sort -u
0:0:c:4:b2:33
0:3:e3:d9:26:c0

tcpdump -ne -r iplog.MY.NET.101.21.200402091650.6 | awk '{print $3}' | sort -u
0:0:c:4:b2:33
0:3:e3:d9:26:c0

Both MAC address prefixes, i.e. 00000C and 0003E3 are assigned to Cisco Systems according to the IEEE Organizationally Unique Identifier (OUI) listing[26]. Now looking at the flow of the traffic, we see that:

1. The inside network appears to be the Class B network 32.245.0.0/16.

2. The Cisco device with a MAC address of 0:3:e3:d9:26:c0 is upstream from that with a MAC address of 0:0:c:4:b2:33

3. Host 32.245.166.236 is apparently running a web browser:

```
21:34:41.046507 IP (tos 0x10, ttl 240, id 0, len 2960) 32.245.166.236.63794 >
207.68.176.190.80: P 2131193308:2131196228(2920) ack 2163772529 win 17520bad cksum 0 (-
>7773)!
***AP*** Seq: 0x33F3CA71  Ack: 0xB4EC56E1  Win: 0x4470  TcpLen: 20
3Ewww.detroit.ru/links/32.php3</displayurl><url>http://www.detro
it.ru/links/32.php3</url>
```

4. Host 32.245.166.119 has a web server running on port tcp/80:

```
01:19:00.616507 32.245.166.119.80 > 194.165.8.161.1025:
<HTML><HEAD>.<TITLE>403.Forbidden</TITLE></HTML>
```

Taking these points into consideration, it is likely that the Snort IDS is located either on a tap or on a port of a switch between two Cisco router/firewall devices with the traffic between them spanned to it as shown in Figure 5. Hence the IDS is not on the subnet of any hosts in seen in the detect file.

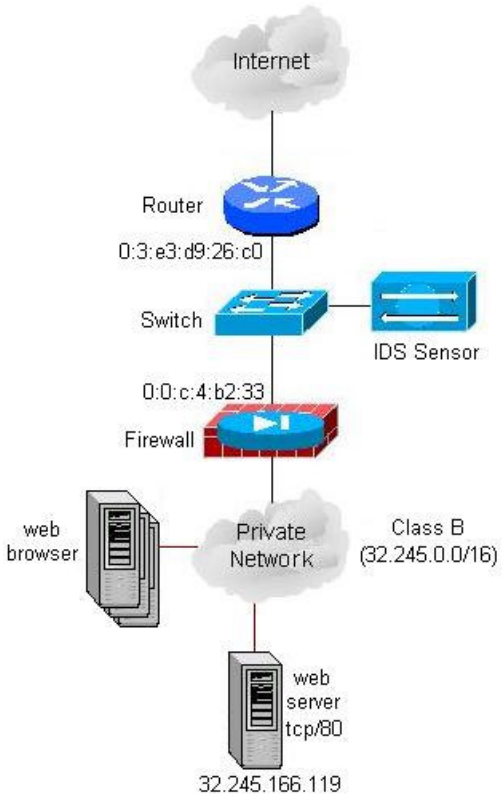

Figure 5. Representative Locations of hosts and IDS Sensor on Network #2

2.2.3 Detect was generated by?

This detect was generated by using Snort 2.1.0 and the "c:\Snort210\bin\snort -c C:\Snort210\etc\snort.conf -r 2002.9.26" -l C:\Snort210\etc -Xde " command. This source file contains the logged packets in the standard tcpdump binary format. So any program capable of reading this format can be used against the detect file, e.g. tcpdump/windump, snort or ethereal.

The summary of running this snort command against this iplog file follows:

```
===================================================
Snort processed 1521 packets.
Breakdown by protocol:                Action Stats:

    TCP: 1521       (100.000%)         ALERTS: 1139
    UDP: 0          (0.000%)          LOGGED: 1248
   ICMP: 0          (0.000%)          PASSED: 0
    ARP: 0          (0.000%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 0          (0.000%)
===================================================
Wireless Stats:
Breakdown by type:
    Management Packets: 0          (0.000%)
```

```
    Control Packets:    0          (0.000%)
    Data Packets:       0          (0.000%)
===========================================================
Fragmentation Stats:
Fragmented IP Packets: 1          (0.066%)
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
===========================================================

TCP Stream Reassembly Stats:
   TCP Packets Used:      1521       (100.000%)
   Reconstructed Packets: 0          (0.000%)
   Streams Reconstructed: 766
===========================================================
```

According to the README file, all of the IP addresses of the protected network space have been "munged". In this detect file all the inside IP addresses have been munged into the Class B network of 32.245.0.0/16.

Before looking at the "BAD-TRAFFIC ip reserved bit set" packet in detail, we'll take an overview of all the traffic. The basic information on the inbound packets found in the trace file is shown in Table 5.

Source IP	Destination IP	Destination Port	TTL	Packet Count	Suspicious?
66.28.100.206	32.245.157.0-32.245.157.253	tcp/8080	43	508	recon
		tcp/3128	43	508	recon
63.111.48.133	32.245.166.236	tcp/ephemeral	113	28	web server response
255.255.255.255[1]	various in 32.245.0.0/16	tcp/515	15	37	Yes (source port: 31337)
207.188.7.150	32.245.166.236	tcp/61638	52	12	No - web server response
66.75.87.174	32.245.166.119	tcp/80	115	8	"WEB-CGI formmail"
65.190.93.101	32.245.166.119	tcp/80	113	6	"WEB-IIS _vti_inf access"
128.167.120.13	32.245.166.236	tcp/ephemeral	48	5	No - web server response
202.29.28.1	32.245.90.118	tcp/80	45	3	Yes - source port tcp/80
	32.245.28.52		45	4	
195.119.1.180	32.245.107.128	tcp/1080	38	4	Yes – syn to Socks
195.2.66.175	32.245.166.119	tcp/80	48	3	No – web server traffic
208.184.39.132	32.245.166.236	tcp/ephemeral	54	3	No - web server response
66.181.168.242	32.245.166.236	tcp/ephemeral	45	3	No - web server response
140.128.251.21	32.245.229.244	tcp/80	50	3	Yes - source port tcp/80
141.154.28.76	32.245.166.119	tcp/80	108	1	No – web server traffic
194.29.197.13			39	1	
194.29.197.27			39	1	
202.57.125.41	32.245.166.119	tcp/80	44	1	"WEB-CGI formmail"
195.2.82.13	32.245.166.236	tcp/ephemeral	48	1	No - web server response
204.202.148.19	32.245.166.236	tcp/ephemeral	113	1	No - web server response
208.184.29.231	32.245.166.236	tcp/ephemeral	54	1	No - web server response
63.241.16.76	32.245.166.236	tcp/ephemeral	46	1	No - web server response

Source IP	Destination IP	Destination Port	TTL	Packet Count	Suspicious?
210.0.222.33	32.245.166.119	tcp/80	42	1	"WEB-CGI formmail"
24.167.47.7	32.245.166.119	tcp/80	109	1	"WEB-CGI formmail"
4.33.83.177	32.245.166.119	tcp/80	112	1	"WEB-CGI formmail"
64.208.107.1	32.245.166.119	tcp/80	112	1	"WEB-CGI formmail"
66.178.21.210	32.245.166.119	tcp/80	47	1	"WEB-FRONTPAGE /_vti_bin/ access"
212.176.56.99	32.245.166.132	tcp/139	100	1	Yes – SMB traffic
61.219.65.74			109	1	
62.202.66.174			113	1	
200.33.24.12			242	1	
217.85.127.102	32.245.71.165		244	1	Yes - frag 0:20@968

Table 5. Inbound Packets found in Detect #2 File

[1] Note: The traffic in this trace with a source IP address of 255.255.255.255 was analyzed by Peter Storm[34]. The reader is referred to that analysis for more information.

The basic information on the outbound packets found in the detect file are shown in Table 6. The fact that 32.245.166.119 generates outbound traffic from a source port of tcp/80 indicated that it is running a publicly accessible web server, apparently an Apache/1.3.12 server on Red.Hat Linux with FrontPage server extensions v4.0.4.3.

```
01:19:00.616507 IP (tos 0x0, ttl 63, id 40697, len 623) 32.245.166.119.80 >
194.165.8.161.1025: P [bad tcp cksum 6fd3 (->a82b)!] 1728057789:1728058372(583) ack 81213162
win 32120 (DF)bad cksum f2c4 (->7dd)!
E..o..@.?......w.....P..g.....6.P.}xo...HTTP/1.1.403.Forbidden..Date:.Sat,.26.Oct.2002.10:08:
45.GMT..Server:.Apache/1.3.12.(Unix)..(Red.Hat/Linux).FrontPage/4.0.4.3..Keep-
Alive:.timeout=15,.max=100..Connection:.Keep-Alive..Transfer-Encoding:.chunked..Content-
Type:.text/html;.charset=iso-8859-1..X-
Pad:.avoid.browser.bug....11f..<!DOCTYPE.HTML.PUBLIC."-
//IETF//DTD.HTML.2.0//EN">.<HTML><HEAD>.<TITLE>403.Forbidden</TITLE>.</HEAD><BODY>.<H1>Forbid
den</H1>.You.don't.have.permission.to.access./main/anpdf/an412.pdf.on.this.server.<P>.<HR>.<A
DDRESS>Apache/1.3.12.Server.at.www.XXXXXXXX.Port.80</ADDRESS>.</BODY>
</HTML>...0....
```

Source IP	Destination IP	Source Port	TTL	Packet Count	Suspicious?
32.245.166.119	194.165.8.161	tcp/80	63	4	No
Source IP	**Destination IP**	**Destination Port**	**TTL**	**Packet Count**	**Suspicious?**
32.245.166.236	various web sites	tcp/80	124,240	274	Yes – varying TTL

Table 6. Outbound Packets found in Detect #2 File

Note that as shown in the following extract, the TTL of packets coming from host 32.245.166.236 have two very different values, i.e. 124 or 240 even though they are apparently coming from the same source IP/source port combination and the same destination IP/destination port combination and occur almost simultaneously. This may be due to the way that the protected network space has been "munged" but further investigation is warranted.

```
19:38:57.886507 IP (tos 0x0, ttl 124, id 13458, len 246) 32.245.166.236.62384 >
64.154.80.48.80
```

```
19:38:58.166507 IP (tos 0x10, ttl 240, id 0, len 245) 32.245.166.236.62384 > 64.154.80.48.80
```

The Snort rule that generated the “BAD-TRAFFIC ip reserved bit set” alert we are interested in is listed below. This rule triggers on IP packets when the reserved bit is set ('fragbits:R').

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC ip reserved bit set";
fragbits:R; sid:523;  classtype:misc-activity; rev:4;)
```

2.2.4 Probability the source address was spoofed

If the “BAD-TRAFFIC ip reserved bit set” packet is not due to corruption then it is likely a reconnaissance attempt. For reconnaissance to be successful the originator needs to see the response, hence IP address spoofing is unlikely. Although it is possible that the hacker could being use a sniffer or a tap to observe the return traffic, he/she would still need to control the source IP address to initiate the TCP connection and the exploits. This control could be via a number of mechanisms such as a Trojan or by using a proxy server.

2.2.5 Description of Attack

The source IP of 217.85.127.102 is named pd9557f66.dip.t-dialin.net and according to RIPE belongs to Deutsche Telekom AG, Internet service provider. Hence the apparent source IP is a dialup user. This IP does not appear as an attacker in the DShield database at this time.

As seen in Table 5, there is only one packet in the trace file with a source IP of 217.85.127.102. This is the actual packet that triggered the “BAD-TRAFFIC ip reserved bit set” alert:

```
01:04:47.966507 IP (tos 0x0, ttl 244, len 40) 217.85.127.102 > 32.245.71.165: tcp (frag
0:20@968)bad cksum 71e7 (->8500)
!
0x0000   4500 0028 0000 8079 f406 71e7 d955 7f66        E..(...y..q..U.f
0x0010   20f5 47a5 8284 0050 410d 5a70 410d 5a70        ..G....PA.ZpA.Zp
0x0020   5004 0000 21a2 0000 0000 0000 0000             P...!.........
```

According to RFC 791[36], the 3-bit flags field should look at follows:

```
Flags:  3 bits
  Various Control Flags.

     Bit 0: reserved, must be zero
     Bit 1: (DF) 0 = May Fragment,  1 = Don't Fragment.
     Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.

        0   1   2
      +---+---+---+
      | 0 | DF| MF|
      +---+---+---+
```

Byte(s)	Explanation
45	4 = IPv4 datagram 5 = IHL (Internet header length) field is 5 words (20 bytes)

Byte(s)	Explanation
00	00 = Type of Service (TOS) field is set to zero so there are no special TOS requirements
0028	0028 = The total length of the IP datagram, including the data field, i.e. 40 bytes
0000	0000 = The 16-bit identification field, which allows a host to determine which datagram a newly arrived fragment belongs to. Each datagram has a unique identification number, and each fragment of a datagram has the same identification number.
8079	(100 0000001111001) - 3-bit flags field (100) = Don't fragment and More fragment bits are not set but the Reserved bit is set which triggered the alert - Fragmentation Offset Length (0000001111001) = 121 * 8 = 968
f4	f4 = TTL of 224
06	06 = TCP protocol
71e7	71e7 = TCP Header checksum which is incorrect as it should be "8500"
d955 7f66	d955 7f66 = 217.85.127.102 (Source IP)
20f5 47a5	20f5 47a5 = 32.245.71.165 (Destination IP)
8284 to 0000	8284 0050 410d 5a70 410d 5a70 5004 0000 21a2 0000 = 20 bytes of data in the current fragment

Table 7. Forensics of packet triggering "BAD-TRAFFIC ip reserved bit set" alert

In Table 7 we can see the reserved bit of the 3-bit flags field is set and therefore not in accordance with RFC 791. The other notable features of this packet are:

1. The TCP header check sum is "71e7" which is incorrect. It should be "8500". This is undoubtedly due to the "munging" of the packets.

2. It is a fragmented packet – The Fragment Offset is 968, i.e. 0x8079.

3. It is the last fragment in the packet as the "More fragment" bit is not set.

4. The Fragment ID = 0

With only 20 bytes of data, we expect this to be the last fragment since typically fragmentation does not occur with data less than 512 bytes, the minimum MTU.

The packet has a Fragment ID = 0 which is possible but we do not have any other traffic from this source IP to see if the Fragment ID changes in different datagrams as we would expect.

Possible benign causes of this packet having the 'reserved bit set' are corruption and data munging. Unfortunately the munging inherently invalidates the TCP Header checksum so that it cannot be readily used in the forensics. As there is only one packet, there is insufficient information to decide if the packet was mangled in transit.

This packet may be an attempt to bypass certain security devices that filter on destination port since this not the first fragment and so does not contain port

information. The sender may hope that the security device does not maintain a state table and so will let this fragment through.

As to why someone might want to do this, the obvious reason is for reconnaissance. Various scanning tools, e.g. nmap, use packets that deviate from RFCs as stimuli to identify the OS on the target host. This fingerprinting approach frequently works since the packets deviate from the standards and by definition there is no standard way of handling them and each manufacturer can choose a different way of doing so. In this alert the deviant packet has the 'reserved bit set'.

This packet is certainly not associated with a fragmentation-based DoS attack, e.g. Teardrop, since it appears to be a single packet. The fact that we only see one packet from the source IP could be because it is conducting a "low and slow" reconnaissance.

Finally the TTL of 224 and the apparent location of the source IP in Germany suggests that the source host is running Solaris 2.x[37].

2.2.6 Correlations

The CVE database[39] does not an entry that corresponds to the packet seen. The closest match is CAN-1999-0240, which is described as "Some filters or firewalls allow fragmented SYN packets with IP reserved bits in violation of their implemented policy." This is only a candidate for inclusion in the CVE list.

The " BAD-TRAFFIC ip reserved bit set " alert was discussed by Ron Shuck (Mon, 10 Feb 2003 20:14:53 -0600) and Soren Macbeth (Tue, 8 Oct 2002 14:35:18 -0400). Their postings to incidents.org can be found on the http://cert.uni-stuttgart.de/archive/intrusions/ mirror.

James Maher also discussed this alert in his practical[38].

2.2.7 Evidence Of Active Targeting

As shown in Table 5, the destination IP of 32.245.71.165 appears only once in the trace. Hence there is no evidence to support a conclusion of active targeting of the destination host.

2.2.8 Severity

The severity of this attack is determined by the following formula:

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

where each value is ranked on a scale from 1 (lowest) to 5 (highest).

- Criticality (a measure of how critical the targeted system is): The assigned value is a “4” since we do not know the importance of this system and so must assume that it is important.

- Lethality (a measure of how severe the damage to the targeted system would be if the attack succeeded): The assigned value is a “1” since no matching vulnerability was found.

- System countermeasures (a measure of the strength of the defensive mechanisms in place on the host itself). The assigned value is a “1” since we do not know the countermeasures in place and so must assume that they are minimal.

- Network countermeasures (a measure of the strength of the defensive mechanisms in place on the network): The assigned value is a “3” since many of the attempted connection attempts the triggered other alerts are not established.

Severity Calculation: 1 = (4 + 1) – (1 + 3)

2.2.9 Defensive Recommendation

Non-RFC compliant packets should be blocked at the perimeter. The inside security device protecting the inside network should be a stateful device so that it is not vulnerable to exploits that are based on type of packet seen here.

As well it would be worthwhile to look for future activity from the source IP because of the possibility of that it could be conducting a "low and slow" reconnaissance.

2.2.10 Multiple Choice Test Question

Question: What is unusual about the following fragmented packet found in a tcpdump file?

```
01:04:47.966507 217.85.127.102 > 32.245.71.165: (frag 0:20@968)
0x0000   4500 0028 0000 2079 f406 8500 d955 7f66
0x0010   20f5 47a5 8284 0050 410d 5a70 410d 5a70
0x0020   5004 0000 21a2 0000 0000 0000 0000
```

E. The More fragment bit is set.
F. The packet has a Fragment ID = 0 which is invalid.
G. Bit 0 of the 3-bit flags field is set.
H. The total length of the IP datagram is 41 bytes.

Answer: A

Explanation: Answer A is correct because the More fragment bit is set yet there is only 20 bytes of data in the packet.

2.2.11 Submission to "intrusions-subscribe@incidents.org"

This detect was submitted to "intrusions-subscribe@incidents.org" on Fri, 5 Mar 2004 00:03:39 –0500. As of the day of submission of this Practical Assignment, 11 March 2004, no feedback was received.

2.3 Network Detect #3 – "SNMP public access udp" alert

2.3.1 Snort Alerts

Running Snort v2.1.0 against the source file named iplog.MY.NET.101.21.200402091650.6 generated the following 3 alerts were generated:

```
[**] [1:1979:1] WEB-MISC perl post attempt [**]
[Classification: Web Application Attack] [Priority: 1]
02/09-17:50:19.000000 0:6:D7:3:17:80 -> 0:D0:FF:7C:14:0 type:0x800 len:0x359
MY.NET.101.21:1238 -> 206.65.188.241:80 TCP TTL:125 TOS:0x1B ID:49193 IpLen:20 DgmLen:843 DF
***AP*** Seq: 0x8BD2D0C  Ack: 0x17222C7F  Win: 0x2058  TcpLen: 20
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=11158][Xref =>
http://www.securityfocus.com/bid/5520]

[**] [1:1411:3] SNMP public access udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
02/09-17:50:19.000000 0:6:D7:3:17:80 -> 0:D0:FF:7C:14:0 type:0x800 len:0x78
MY.NET.101.21:41080 -> 172.18.250.142:161 UDP TTL:125 TOS:0x0 ID:11630 IpLen:20 DgmLen:106
Len: 78
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0013][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0012][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0517]

[**] [119:12:1] (http_inspect) APACHE WHITESPACE (TAB) [**]
02/09-17:50:20.000000 0:6:D7:3:17:80 -> 0:D0:FF:7C:14:0 type:0x800 len:0x59A
MY.NET.101.21:4520 -> 66.35.229.175:80 TCP TTL:236 TOS:0x0 ID:62396 IpLen:20 DgmLen:1420
***A*R** Seq: 0x157AA768  Ack: 0xB569B799  Win: 0x8363  TcpLen: 20
```

The "SNMP public access udp" alert will be examined in detail because of the recent widely publicized vulnerabilities in SNMP v1.0.

2.3.2 Source of Trace

The source of this detect is an iplog file named iplog.MY.NET.101.21.200402091650.6 that was generated by a Cisco IDS v3.1 sensor set to log traffic from MY.NET.101.21 in a tcpdump format after a signature ID of 3050 (Half-open SYN Attack). The file is 1,047,269 bytes and dated 1650 hours on 9 February 04.

The IDS is supposed to be monitoring the traffic between a border router and a firewall. To confirm this we run the following commands to see first the source MAC addresses and then the destination ones in the traffic as seen by the IDS:

```
tcpdump -ne -r iplog.MY.NET.101.21.200402091650.6 | awk '{print $2}' | sort -u
0:6:d7:3:17:80
0:d0:ff:7c:14:0

tcpdump -ne -r iplog.MY.NET.101.21.200402091650.6 | awk '{print $3}' | sort -u
0:6:d7:3:17:80
0:d0:ff:7c:14:0
```

Both MAC address prefixes, i.e. 0006D7 and 00D0FF are assigned to Cisco Systems according to the IEEE Organizationally Unique Identifier (OUI) listing[25].

Looking at the flow of the traffic as seen in the following extract, we see that the MY.NET address traffic is coming from the device with a MAC of 0:6:d7:3:17:80:

```
17:50:19.000000 0:6:d7:3:17:80 0:d0:ff:7c:14:0 0800 60: IP MY.NET.101.21.61752 >
192.206.43.77.20: . ack 3020 win 64512 (DF)
17:50:19.000000 0:d0:ff:7c:14:0 0:6:d7:3:17:80 0800 1414: IP 192.206.43.77.20 >
MY.NET.101.21.61752: . 3020:4380(1360) ack 1 win 65535 (DF)
```

That the IDS is only dumping traffic from MY.NET.101.21 is confirmed by the fact the following two word counts are identical:

```
tcpdump -ne -r iplog.MY.NET.101.21.200402091650.6 | grep "0:6:d7:3:17:80" | grep "
MY.NET.101.21" | wc -l
   1982
tcpdump -ne -r iplog.MY.NET.101.21.200402091650.6 | grep "0:6:d7:3:17:80" | wc -l
   1982
```

We also know that MY.NET.101.21 is one of the IP addresses in the NAT pool of the firewall.

Taking all this into consideration, we have confirmed that the IDS sensor is located on a port of a switch (taps are not being used) between two Cisco router/firewall devices with the traffic between them spanned to it as shown in Figure 6. Hence the IDS is not on the subnet of any hosts in seen in the detect file.

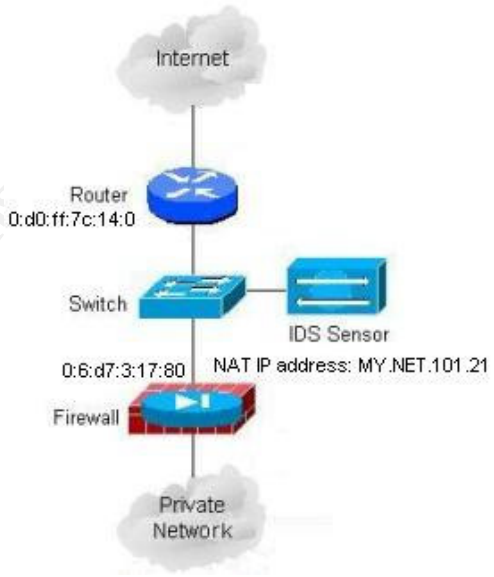

Figure 6. Representative Locations of IDS Sensor for Detect #3

2.3.3 Detect was generated by?

This detect was generated by using Snort 2.1.0 and the "c:\Snort210\bin\snort -c snort.conf -r iplog.MY.NET.101.21.200402091650.6" -l C:\Snort210\etc -Xde " command against the iplog.MY.NET.101.21.200402091650.6 file. This file contains the logged packets in the standard tcpdump binary format. So any program capable of reading this format can be used against the detect file, e.g. tcpdump/windump, snort or ethereal.

The summary of running this snort command against this iplog file follows:

```
===================================================

Snort processed 1982 packets.
Breakdown by protocol:                Action Stats:

    TCP: 1980       (99.899%)         ALERTS: 3
    UDP: 2          (0.101%)          LOGGED: 4
   ICMP: 0          (0.000%)          PASSED: 0
    ARP: 0          (0.000%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 0          (0.000%)
===================================================
Wireless Stats:
Breakdown by type:
    Management Packets: 0        (0.000%)
    Control Packets:    0        (0.000%)
    Data Packets:       0        (0.000%)
===================================================
Fragmentation Stats:
Fragmented IP Packets: 0        (0.000%)
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
===================================================

TCP Stream Reassembly Stats:
   TCP Packets Used:      1980       (99.899%)
   Reconstructed Packets: 38         (1.917%)
   Streams Reconstructed: 69
===================================================
```

The "SNMP public access udp" alert found in the trace looks as follows:

```
[**] [1:1411:3] SNMP public access udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
02/09-17:50:19.000000 0:6:D7:3:17:80 -> 0:D0:FF:7C:14:0 type:0x800 len:0x78
MY.NET.101.21:41080 -> 172.18.250.142:161 UDP TTL:125 TOS:0x0 ID:11630 IpLen:20 DgmLen:106
Len: 78
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0013][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0012][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0517]
```

The "SNMP public access udp" alert is generated when an SNMP connection over UDP using the default 'private' community is made[29].

2.3.4 Description of Attack

The IP address 172.18.250.142 is in the IANA blackhole network of 172.16.0.0/12. Hence the firewall shown in Figure 6 is letting out traffic that should be blocked unless the border router has an interface to a private network that uses addresses from 172.16.0.0/12.

The output file is produced by running "*windump -nXvv -r iplog.MY.NET.101.21.200402091650.6*" and then grepped for packets with MY.NET.101.21:41080 and 172.18.250.142:161. The following is the only such packet in the trace and the only packet destined for the host 172.18.250.142:

```
grep "167\.83\.101\.21\.41080" windump_nXvvr_iplog.MY.NET.101.21.200402091650.6 | grep "172\.18\.250\.
142\.161"

17:50:19.000000 IP (tos 0x0, ttl 125, id 11630, len 106) MY.NET.101.21.41080 > 172.18.250.142.161: [udp sum ok]  {
SNMPv1 { GetRequest(63) R=214  .1.3.6.1.2.1.25.3.2.1.5.1 .1.3.6.1.2.1.25.3.5.1.1.1 .1.3.6.1.2.1.25.3.5.1.2.1 } }
0x0000          4500 006a 2d6e 0000 7d11 5d0b a753 6515        E..j-n..}.]..Se.
0x0010          ac12 fa8e a078 00a1 0056 e8a3 304c 0201        .....x...V..0L..
0x0020          0004 0670 7562 6c69 63a0 3f02 0200 d602        ...public.?.....
0x0030          0100 0201 0030 3330 0f06 0b2b 0601 0201        .....030...+....
0x0040          1903 0201 0501 0500 300f 060b 2b06 0102        ........0...+...
0x0050          0119 0305 0101 0105 0030 0f06 0b2b 0601        .........0...+..
0x0060          0201 1903 0501 0201 0500                       ..........
```

Furthermore, running the following command on the output file shows that there is only one packet to or from port 161:

```
grep "\.161" windump_nXvvr_iplog.MY.NET.101.21.200402091650.6 | grep -v "\.161\."
```

Examining the SNMP GetRequest packet in more detail, we see that the host MY.NET.101.21 is requesting the following object identifiers (OID):

```
.1.3.6.1.2.1.25.3.2.1.5.1 $DEVICE_STATUS_OID    = '1.3.6.1.2.1.25.3.2.1.5.1';    # hrDeviceStatus.1
.1.3.6.1.2.1.25.3.5.1.1.1 $PRINTER_STATUS_OID = '1.3.6.1.2.1.25.3.5.1.1.1';    # hrPrinterStatus.1
.1.3.6.1.2.1.25.3.5.1.2.1 $ERROR_STATE_OID       = '1.3.6.1.2.1.25.3.5.1.2.1';    # hrPrinterDetectedErrorState.1
```

The hr OIDs are from the Host Resources (hr) MIB used for managing host systems. This MIB instruments attributes common to all internet hosts including, for example, both personal computers and systems that run variants of Unix. The requested ones are defined as follows[30.]:

- hrDeviceStatus - The current operational state of the device which can be: unknown, running, warning, testing or down.
- hrPrinterStatus - The current status of this printer device which can be: other, unknown, idle, printing or warmup.
- hrPrinterDetectedErrorState - This object represents any error conditions detected by the printer which can be: lowPaper, noPaper, lowToner, noToner, doorOpen, jammed, offline and serviceRequested.

If the source of this request was an authorize SNMP management station then the information provided by these OIDs would be useful for alerting an operator to specific warning or error conditions that may occur, especially those requiring human intervention."

It appears that the host MY.NET.101.21 is interrogating the host 172.18.250.142 for printer related information using the default SNMP read community string of “public”. Hence the host 172.18.250.142 is likely a printer.

2.3.5 Attack Mechanism

SNMP request messages are sent from managers to agents. SNMP agents must properly decode request messages and process the resulting data. In testing, OUSPG found multiple vulnerabilities in the way many SNMP agents decode and process SNMP request messages. Vulnerabilities in the decoding and subsequent processing of SNMP messages by both managers and agents may result in denial-of-service conditions, format string vulnerabilities, and buffer overflows. Some vulnerabilities do not require the SNMP message to use the correct SNMP community string. These vulnerabilities have been assigned the CVE identifiers CAN-2002-0012 and CAN-2002-0013, respectively[31].

It has been reported that the HP JetDirect firmware is more susceptible to SNMP vulnerabilities than originally referenced in the CERT Advisory CA-2002-03[32]. The testing indicated that devices with JetDirect firmware x.08.32 crash each time a single malformed SNMP packet was received.

2.3.6 Correlations

The "SNMP Public Access udp" alert was discussed on the Snort-users mailing list[33]. The essence of the thread was that Windows client drivers for HP Printers containing JetDirect cards use SNMP to determine the printer's extended status usually using the default SNMP community of “public”.

2.3.7 Evidence Of Active Targeting

The trace shows that the host MY.NET.101.21 is interrogating the host 172.18.250.142 via SNMP for a count of one packet. However looking at some other available traces we see that three such packets:

```
17:50:19.000000 IP (tos 0x0, ttl 125, id 11630, len 106) MY.NET.101.21.41080 > 172.18.250.142.161: [udp sum ok]  {
SNMPv1 { GetRequest(63) R=214  .1.3.6.1.2.1.25.3.2.1.5.1 .1.3.6.1.2.1.25.3.5.1.1.1 .1.3.6.1.2.1.25.3.5.1.2.1 } }

17:50:26.000000 IP (tos 0x0, ttl 125, id 11638, len 106) MY.NET.101.21.41080 > 172.18.250.142.161: [udp sum ok]  {
SNMPv1 { GetRequest(63) R=214  .1.3.6.1.2.1.25.3.2.1.5.1 .1.3.6.1.2.1.25.3.5.1.1.1 .1.3.6.1.2.1.25.3.5.1.2.1 } }

17:50:32.000000 IP (tos 0x0, ttl 125, id 11696, len 106) MY.NET.101.21.41080 > 172.18.250.142.161: [udp sum ok]  {
SNMPv1 { GetRequest(63) R=214  .1.3.6.1.2.1.25.3.2.1.5.1 .1.3.6.1.2.1.25.3.5.1.1.1 .1.3.6.1.2.1.25.3.5.1.2.1 } }
```

Since these three packets are so close together and they all use the same source port, it is likely that then same inside host is responsible for this traffic and that the

destination IP did not respond. We could determine if the same inside host is responsible for this traffic by looking at the firewall builds and teardowns events found on the syslog server. Recall that the source IP is an address in the firewall's NAT pool so the true inside IP address is not obvious.

2.3.8 Severity

The severity of this attack is determined by the following formula:

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

where each value is ranked on a scale from 1 (lowest) to 5 (highest).

- Criticality (a measure of how critical the targeted system is): The assigned value is a "1" since it is likely that the targeted host is a simple print server and alternate printer probably available.

- Lethality (a measure of how severe the damage to the targeted system would be if the attack succeeded): The assigned value is a "4" since the SNMP vulnerability discussed may cause denial-of-service conditions, service interruptions and in some cases may allow an attacker to gain access to the affected device.

- System countermeasures (a measure of the strength of the defensive mechanisms in place on the host itself). The assigned value is a "1" since due to workload, it is unlikely that the targeted printer server was patched against the SNMP vulnerability identified in CERT Advisory CA-2002-03.

- Network countermeasures (a measure of the strength of the defensive mechanisms in place on the network): The assigned value is a "2" since the traffic passed through the firewall and inside router that are used to control what inside users are allowed to do.

Severity Calculation: 2 = (1 + 4) – (1 + 2)

2.3.9 Defensive Recommendation

If the investigation shows that inside hosts are required to issue SNMP queries then the ACLs on the firewall and inside router must be checked to ensure that this SNMP traffic is restricted only to specific destination hosts and from specific inside hosts or networks. As well, all print servers need to be updated in accordance with CERT Advisory CA-2002-03.

2.3.10 Multiple Choice Test Question

Question: How can you most efficiently and effectively determine if SNMP enabled devices on a network are using the default read and write community strings?

A. Log on to each device and check its configuration?
B. Use a network scanner and set it to look for the response to the default read and write community strings?
C. Examine the syslog server for ACL hits or events involving tcp/161?
D. Examine the syslog server for events containing the phrases “public” or “private”?

Answer: B

Explanation: Answer B is correct because many scanners can be easily configured to scan for devices that accept the default community strings and once configured they can be scheduled to run periodically and send out the reports. The other answers are either wrong or not as efficient or effective.

Part 3 - Analyze This

3.1 Executive Summary

This section is a security audit of a university that was performed by analyzing logs from their Intrusion Detection System (IDS) over the period of October 1-5, 2003. This audit paid particular attention to signs of compromised systems and other network problems.

The top talkers, top listeners, top signatures, top source ports and top destination ports were extracted to determine the most important detects over the aforementioned period. Next more specific information was extracted from the log files and the following detects were examined in detail:

1. SMB Name Wildcard Signature (902,224 hits)
2. “MY.NET.30.4 activity” Signature (50,224 hits)
3. Incomplete Packet Fragments Discarded (7,604 hits)
4. MY.NET.30.3 activity (7,216 hits)
5. High port 65535 tcp & udp - possible Red Worm – traffic (9,038 hits)
6. Null scan! (2,903 hits)
7. Tiny Fragments - Possible Hostile Activity (2,375 hits)
8. EXPLOIT x86 NOOP (1,462 hits)
9. Possible Trojan server activity (489 hits)

The study of these detects showed that a handful of student systems accounted for an inordinate number of the events reported by the IDS sensor. Action is required to ensure that these systems stop generating such traffic. To this end, the university’s Acceptable User Policy (UAP) needs to be strengthened so that malicious traffic is clearly defined and systems generating such traffic can be removed from the network until it stops.

On the practical side, there are a number of Snort modifications recommended in this audit that need to be actioned so as to make the sensor output more useful and thereby enhancing the university’s security posture.

Finally it is recommended that management adopt the following two measures to improve the university's security posture in a cost effective manner:

1. A security policy based on an “only allow what is explicitly permitted and deny everything else” approach.

2. A defense in depth approach to reduce the chance of an intrusion.

3.2 List of the files analyzed

The requirement states that "You must select five (5) consecutive days worth of files in other words, you should have a minimum of five (5) files of each log type (Scans, Alerts, and OOS (Out of Spec) at least one file of each type for each day) for analysis."

For this part I used the files shown in Table 8 those data covered the period of October 1-5, 2003.

File Type	File 1 (Oct 1)	File 2 (Oct 2)	File 3 (Oct 3)	File 4 (Oct 4)	File 5 (Oct 5)
Scan Files	scans.031001.gz 17,491,421 Sun Oct 5 05:02:28 2003	scans.031002.gz 20,924,079 Mon Oct 6 05:02:13 2003	scans.031003.gz 16,121,849 Tue Oct 7 05:02:20 2003	scans.031004.gz 16,716,731 Wed Oct 8 05:01:26 2003	scans.031005.gz 16,381,378 Thu Oct 9 05:00:53 2003
(Dates covered)	Oct 1 00:00:00 to Oct 1 23:55:27	Oct 2 00:00:01 to Oct 2 23:59:12	Oct 3 00:00:01 to Oct 3 23:54:45	Oct 4 00:00:01 to Oct 4 23:55:28	Oct 5 00:00:01 to Oct 5 23:57:07
Alert Files	alert.031001.gz 6,334,079 Sun Oct 5 05:02:06 2003	alert.031002.gz 7,343,955 Mon Oct 6 5:01:44 2003	alert.031003.gz 4,874,492 Tue Oct 7 05:01:53 2003	alert.031004.gz 2,524,857 Wed Oct 8 05:01:02 2003	alert.031005.gz 2,468,708 Thu Oct 9 05:00:38 2003
(Dates covered)	10/01-00:00:00.801771 to 10/02-00:10:56.533465	10/02-00:00:01.503292 to 10/02-23:41:18.172608	10/03-00:00:01.083849 to 10/04-00:09:48.903583	10/04-00:16:03.195630 to 10/05-00:07:24.597884	10/05-00:16:05.131231 to 10/06-00:07:27.498096
OOS Files	OOS_Report_2003_10_02_3730 1,218,563 Thu Oct 2 00:08:13 2003	OOS_Report_2003_10_03_10388 870,403 Fri Oct 3 00:08:09 2003	OOS_Report_2003_10_04_7703 931,843 Sat Oct 4 00:05:16 2003	OOS_Report_2003_10_05_7893 834,563 Sun Oct 5 00:08:11 2003	OOS_Report_2003_10_06_14370 890,883 Mon Oct 6 00:05:16 2003
(Dates covered)	10/01-00:05:10.064411 to 10/02-00:00:02.552338	10/02-00:05:50.065593 to 10/03-00:02:18.032469	10/03-00:05:22.099104 to 10/04-00:04:01.804947	10/04-00:06:41.950857 to 10/04-23:49:21.969584	10/05-00:07:45.598277 to 10/05-23:57:16.038311

Table 8. Log files selected to Analyze

As can be seen in Table 8, there is no obvious correspondence between dates of the files and the actual dates of the alerts. Unfortunately this means that time is wasted trying to find the matching OOS, Scan and Alert data and contributes nothing to the exercise.

These files are found at the URL: http://www.incidents.org/logs. The README file (see Section 2.1.3 for full extract) states that:

> "The log files are the result of a Snort instance running in binary logging mode. This means that only the packets that violate the ruleset will appear in the log. The logs themselves have been sanitized."

The detect file containing the logged packets is in the standard tcpdump binary format. So any program capable of reading this format can be used against the detect file, e.g. tcpdump/windump, snort or ethereal.

3.3 Scan Analysis

In this section the scanning activity will be analyzed. Based on an examination of the scans, alerts and OOS files and the fact that they are related, it is assumed that 130.85.x.x is the same network as MY.NET.x.x.

These scans contain a number of alerts with "SYN 12****S* RESERVEDBITS" as shown below. According to Martin Roesch[23] this type of alert is due to a bug that required modifications to the TOS plugin that allow better detection of non-ECN reserved bit usage.

```
Oct  5 23:44:02 194.249.91.190:38793 -> 130.85.24.44:80 SYN 12****S* RESERVEDBITS
Oct  5 23:50:36 63.71.152.2:56385 -> 130.85.100.230:113 SYN 12****S* RESERVEDBITS
```

The internal "top talkers" list for scanning activity is found in Table 9. These hosts are the overall "top talkers", accounting for 86% (9,626,391) of the 11,186,574 scanning events that were triggered. However, as can be seen in the remark column, many of these scanning events are in fact benign since they are due to legitimate network activity. The potentially infected hosts identified in Table 9 need to be investigated.

Source IP	Event Count	Remark
130.85.1.3	2,753,737	Mainly to destination port udp/53 on Internet name servers, hence the host is probably an internal name server.
130.85.84.194	1,759,332	SYN scanning tcp/135 of Internet address spaces including Tyndall AFB (131.55.0.0) and Upper Heyford AFB (131.56.0.0), hence the host probably has a worm.
130.85.163.107	1,750,341	SYN scanning tcp/135 of Internet address spaces including University of Alabama (130.160.0.0) and Ericsson (130.100.0.0), hence the host probably has a worm.
130.85.84.232	1,204,595	Mainly from source port udp/3383 (destinations are Internet hosts which appear to be end-users such as mrdh-a-160.resnet.purdue.edu (24.153.23.66) and 12-206-176-242.client.attbi.com (24.153.23.66)) – while udp/3383 in IANA listed as the Enterprise Software Products License Manager (esp-lm), this traffic might be due to some P2P software. Also some SYNs to Internet hosts to non-standard ports[i].
130.85.163.76	633,618	Mainly from source port udp/6257 (destinations are Internet hosts mainly udp/6257) which is probably WinMX file-sharing program (http://www.solidshare.com/winmx.html). This host may be infected with the Kuang2 the Virus Trojan[ii] (see http://isc.incidents.org/port_details.html?port=17300).
130.85.162.118	633,161	Mainly from source port udp/1025-1026 while destination port is udp/137 (destinations are Internet hosts). This host may be infected with the Kuang2 the Virus Trojan[iii] (see http://isc.incidents.org/port_details.html?port=17300).
130.85.1.5	407,677	Mainly to destination port udp/53 on Internet name servers, hence the host is probably an internal name server, some ntp traffic (source port udp/123).
130.85.84.143	240,290	Mainly from source and destination port udp/4672 (destinations are Internet hosts which appear to be end-users such as port-212-202-71-10.reverse.qsc.de (212.202.71.10[iv]) which is probably

Source IP	Event Count	Remark
		eMule P2P program file-sharing program (http://www.eMule-project.net/home/perl/help.cgi?l=1&rm=show_topic&topic_id=122).
130.85.112.151	136,934	SYN scanning tcp/135 of Internet address spaces including UUNET Technologies (63.109.0.0), hence the host probably has a worm. This host may be infected with the Kuang2 the Virus Trojan[v] (see http://isc.incidents.org/port_details.html?port=17300).
130.85.70.176	106,706	Mainly from source port udp/6257 (destinations are Internet hosts mainly udp/6257) which is probably WinMX file-sharing program (http://www.solidshare.com/winmx.html). This host may be infected with the Kuang2 the Virus Trojan[vi] (see http://isc.incidents.org/port_details.html?port=17300).

Table 9. Internal "Top Talkers" list for scanning activity

Notes:

[i] Event showing host 130.85.84.232 trying to connect to tcp/2125: "Oct 5 00:59:08 130.85.84.232:1475 -> 172.197.187.92:2125 SYN ******S*" (from scans.031005.gz).

[ii] Event showing end-user host ip68-106-40-188.ph.ph.cox.net (68.106.40.188) trying to connect to tcp/17300 on 130.85.163.76: "Oct 5 18:05:51 68.106.40.188:2468 -> 130.85.163.76:17300 SYN ******S*" (from scans.031005.gz).

[iii] Event showing end-user host pcp02561432pcs.owngsm01.md.comcast.net (68.55.31.197) trying to connect to tcp/17300 on 130.85.163.76: "Oct 5 19:44:11 68.55.31.197:4504 -> 130.85.162.118:17300 SYN ******S*" (from scans.031005.gz).

[iv] Event showing end-user host pcp02561432pcs.owngsm01.md.comcast.net (68.55.31.197) trying to connect to tcp/17300 on 130.85.163.76: "Oct 5 19:44:11 68.55.31.197:4504 -> 130.85.162.118:17300 SYN ******S*" (from scans.031005.gz).

[v] Event showing end-user host gso88-192-169.triad.rr.com (24.88.192.169) trying to connect to tcp/17300 on 130.85.112.151: "Oct 5 14:48:13 24.88.192.169:4443 -> 130.85.112.151:17300 SYN ******S*" (from scans.031005.gz).

[vi] Event showing end-user host dsl-pb-1777.linkline.com (64.30.211.151) trying to connect to tcp/17300 on 130.85.70.176: "Oct 5 23:42:10 64.30.211.151:2137 -> 130.85.70.176:17300 SYN ******S*" (from scans.031005.gz).

The "Top Destination Port" list for scanning activity is found in Table 10. These ports account for 83% (9,230,661) of the destination ports associated with the scanning events that were triggered.

The recommendation column of Table 10 is designed to better restrict the traffic to legitimate network activity in accordance with the organization's putative Acceptable Use Policy (AUP). Normally network users must agree to the AUP. Typically this AUP would restrict the use of the network to business-related activities. Hence the use of Peer to Peer (P2P) programs such as eMule would probably be restricted.

Destination Port	Event Count	Likely Service	Recommendation
135	3,838,043	Microsoft Remote Procedure Call (RPC) Service	Block inbound and outbound traffic at edge devices
53	3,146,401	Domain Name Service (DNS)	Restrict inbound and outbound traffic at edge devices to authorized name servers
6257	697,650	Probably WinMX file-sharing program	Block inbound and outbound traffic at edge devices (policy violation)
137	642,146	Microsoft NetBIOS Name Service	Block inbound and outbound traffic at edge devices
80	357,102	HTTP	Block inbound traffic to tcp/80 at edge devices to authorized web servers
17300	164,482	Kuang2 the Virus Trojan	Block inbound and outbound traffic at edge devices
4672	131,857	eMule P2P program file-sharing program[ii]	Block inbound and outbound traffic at edge devices (policy violation)
4662	107,877	eMule P2P program file-sharing program[ii]	Block inbound and outbound traffic at edge devices (policy violation)
554	80,430	Real Time Streaming Protocol (RTSP) used by RealServer software that streams media (RealAudio and RealVideo)[i]	Block inbound and outbound traffic at edge devices (policy violation)
25	64,673	SMTP	Restrict inbound and outbound traffic at edge devices to authorized mail servers

Table 10. "Top Destination Port" list for scanning activity

Notes:

[i] RealServer FAQ, http://www.servicad.com/network/pdfs/RealServerFAQ.pdf

[ii] eMule Ports, http://www.emule-project.net/home/perl/help.cgi?l=1&rm=show_topic&topic_id=122

Although destination port tcp/17300 is not in the top ten listed in Table 10, it is the 11th most frequently scanned port (4,516 hits). The hosts that are the subject of port tcp/17300 scanning ought to be examined to see if they are infected with the Kuang2 the Virus Trojan.

3.4 Alert Analysis

In Table 11 the number of alerts are shown by the following general categories:

1. "*spp_portscan*" alerts: "spp_portscan" stands alerts generated by the portscan Snort Preprocessor Plugin. The Snort Portscan Preprocessor logs the start and end of portscans from a single source IP to the standard logging facility and if a log file is

specified, it logs the destination IPs and ports scanned as well as the type of scan. A portscan is defined as TCP connection attempts to more than P ports in T seconds or UDP packets sent to more than P ports in T seconds. Ports can be spread across any number of destination IP addresses, and may all be the same port if spread across multiple IPs. So portscans possibilities are single IP -> single IP and single IP -> many IPs.

The format for portscan is : <*monitor network*> <*number of ports*> <*detection period*> <*file path*>

In the logs, these alerts look as follows:

```
10/05-00:16:11.011277  [**] spp_portscan: portscan status from 194.249.91.190: 1 connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
10/05-00:16:15.855629  [**] spp_portscan: End of portscan from 194.249.91.190: TOTAL time(0s) hosts(1) TCP(1) UDP(0) STEALTH [**]
10/05-00:16:19.014836  [**] spp_portscan: PORTSCAN DETECTED from 202.196.105.27 (THRESHOLD 12 connections exceeded in 0 seconds) [**]
```

These alerts are used for correlation purposes when “non-ICMP (non-spp_portscan)” alerts are examined.

2. “*non-ICMP (non-spp_portscan)* “ alerts: These alerts are neither spp_portscan alerts nor are they ICMP-related ones, so they are called “non-ICMP (non-spp_portscan)” alerts.

   These top of this type of alert is examined in detail in Section 3.6 .

3. “*ICMP (non-spp_portscan)*” alerts: These alerts are ICMP alerts that are not spp_portscan alerts, so they are called “ICMP (non-spp_portscan)” alerts. These alerts look as follows:

```
10/05-01:16:21.787722 ;ICMP SRC and DST outside network;172.161.135.126;172.163.59.154
10/05-02:07:13.383605 ;ICMP SRC and DST outside network;172.139.162.66;172.137.243.248
```

All of this type of alerts involved ICMP traffic to and from hosts outside the internal network. They will not be examined further, in fact the purpose of this type of alert needs to be examined, especially if ICMP is not allowed into the network.

Number of Alerts	File 1 (Oct 1 Wed)	File 2 (Oct 2 Thu)	File 3 (Oct 3 Fri)	File 4 (Oct 4 Sat)	File 5 (Oct 5 Sun)
non-ICMP (non-spp_portscan)	323,051	389,185	213,055	47,074	17,598
ICMP (non-spp_portscan)	327	313	287	313	262
spp_portscan	226,224	254,824	220,058	199,690	202,047

Table 11. Number of Alerts by General Categories

In Figure 7, we plot the three general categories of alerts found in Table 11 over the five-day period (note that the ICMP (non-spp_portscan) category of alerts uses the right-hand side Y-axis). We can see that the "spp_portscan" and "ICMP (non-spp_portscan)" alerts remain constant over this period while the "non-ICMP (non-spp_portscan)" alerts dramatically drop off on the weekend. Clearly those users that caused the "non-ICMP (non-spp_portscan)" alerts took the weekend off. In fact looking at the statistics shown in Table 16, we can see the reason is that several inside hosts are not active, e.g. MY.NET.162.118 and MY.NET.150.133 that accounted for 90% of these alerts do not appear on the weekend.

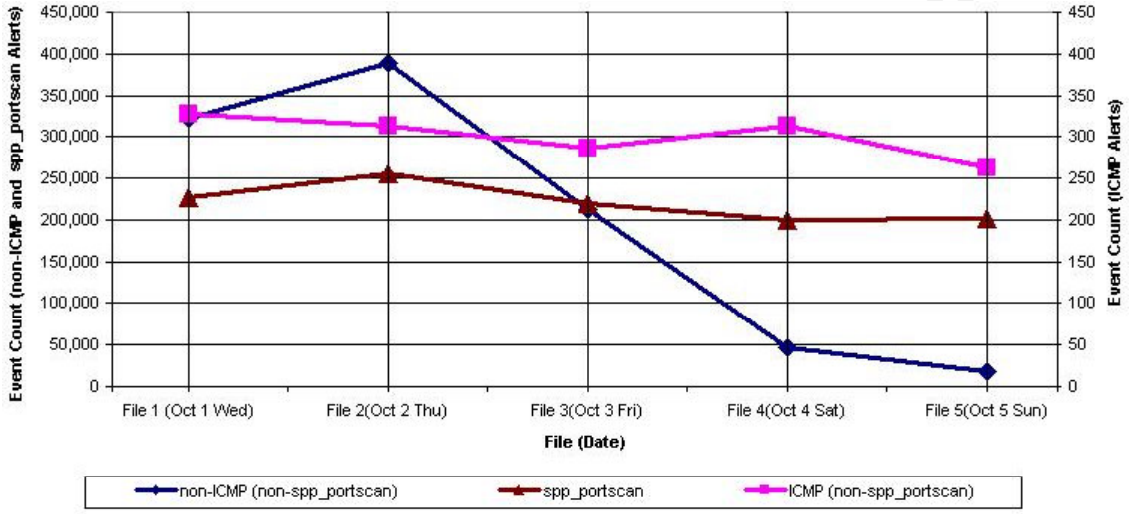

Figure 7. Graphs of the 3 Categories of Alerts over the five-day Period

3.5 OOS Analysis

The OOS files contain packets that were logged because they do not meet RFC standards for some reason, e.g., packets with UPSF flags set. The reason usually involves the layer 3 and 4 headers. Some of these packets may also be in the scan logs or possibly the alert logs.

Now an attacker is free to set whatever flags he chooses to set on any packet. RFC 793 defines the meaning of certain bits, but does not address scenarios in which odd bit combinations are encountered. The method in which the packet is handled is left up to the TCP stack designer and as you might imagine, different implementations handle these "odd bit combinations" in a variety of ways. Some implementations are more liberal in what they accept, and choose to treat any packet with the SYN bit set (regardless of what other bits are set or unset) as the opening in the three-way handshake. Some implementations are more conservative, and drop packets with unexpected combinations of flags. In the case of a liberal packet-filtering firewall, it may allow a packet to pass if, for example, the FIN bit is set, even if the SYN bit is set[46].

Table 12 lists the Top Ten source IP addresses and their destinations hosts and ports that are found in the OOS Logs.

Source IP	Alert Count	Destination IP	Destination Port
194.249.91.190	2942	MY.NET.24.44:80	80
195.101.94.101	657	24 unique MY.NET.x.x hosts	80
195.101.94.208	540	17 unique MY.NET.x.x hosts	80
195.101.94.209	457	14 unique MY.NET.x.x hosts	80
MY.NET.216.50	443	13 unique MY.NET.x.x hosts	- well-known ports: 21, 22, 25, 80, 119, 143, 389, 443, 465 - ephemeral ports: 4071-4141, 8765
205.244.242.28	377	3 unique MY.NET.x.x hosts	25
213.186.35.9	353	9 unique MY.NET.x.x hosts	- well-known ports: 23, 80, 81 - ephemeral ports: 1080, 3128, 6588, 8000, 8001, 8080, 8081, 8888
216.95.201.13	337	5 unique MY.NET.x.x hosts	25
66.225.198.20	333	MY.NET.12.6	25
216.95.201.18	327	5 unique MY.NET.x.x hosts	25

Table 12. Top Ten Source IP Addresses and their Destinations in the OOS Logs

RFC 793 reserved the 6 bits before the TCP flags for future use and stated that they must be zero[40]. The TCP flags or control bits are the 6 lower order bits in the 13th byte offset of the TCP header and these bits are defined as follows (see Figure 8):

URG: Urgent Pointer field significant
ACK: Acknowledgment field significant
PSH: Push Function
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: No more data from sender

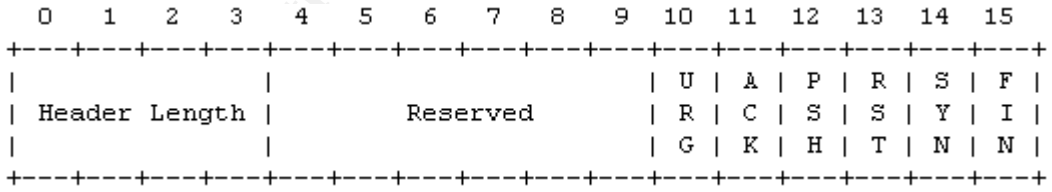

Figure 8. RFC 793 definition of bytes 13 and 14 of the TCP header

More recently, RFC 3168[41] redefined bytes the 13th byte offset of the TCP header as shown in Figure 9. The two new flags in the Reserved field of the TCP header are the ECN-Echo (ECE) flag (Bit 9) and the Congestion Window Reduced (CWR) flag (Bit 8). These are used to add Explicit Congestion Notification (ECN) to IP.

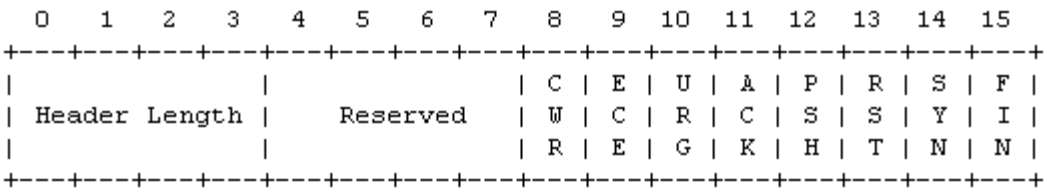

Figure 9. RFC 3168 definition of bytes 13 and 14 of the TCP header

Snort represents the TCP flags including the two ECN as “12UAPRSF” when they are all set[42]. The reserved bits can be used to detect unusual behavior, such as IP stack fingerprinting attempts or other suspicious activity. To handle writing rules for session initiation packets such as ECN where a SYN packet is sent with the previously reserved bits 12 set, an option mask may be specified. For example, the following rule checks for a SYN FIN packet regardless of the values of the reserved bits:

```
alert any any -> 192.168.1.0/24 any (flags: SF,12; msg: "Possible SYN FIN scan";)
```

Table 13 lists the breakdown of OOS alerts based on the flag settings in the packets. This table is folded into three double columns to conserve space.

Flags	Count	Flags	Count	Flags	Count
12****S*	14518	12U***SF	1	1*U*PRSF	1
********	311	12U*P*SF	1	1*UAP*SF	1
****P***	141	12U*PR*F	1	1*UAPRSF	1
12***R**	42	12U*P**F	1	1*****SF	1
12*A*R**	3	12U*P***	1	1***PRSF	1
12UAPRS*	2	12UA**SF	1	**U**RSF	1
12**PRS*	2	12UAP*SF	1	**U*****	1
12**PR*F	2	12UAPRSF	1	******SF	1
12*A*R*F	2	12UA***F	1	*****RSF	1
1*U**RSF	2	12***RS*	1	****P*SF	1
**U*P*SF	2	12***R*F	1	***AP*SF	1
**UAP*SF	2	12**P**F	1	***APRSF	1
****PRSF	2	12*A*RSF	1	*2U**RSF	1
*2****SF	2	12*A*RS*	1	*2UA**SF	1
*2*A*RSF	2	12*AP*S*	1		

Table 13. Flag Settings found in the OOS packets

As can be seen in Table 13, 96.38% of the OOS alerts are simply SYN packets with using ECN. There is nothing inherently abnormal about these packets but in the days before ECN these packets were indicative of potential malicious activity such as a Queso Fingerprint attempt. The following old simple rule for Queso would have produced false positives:

```
alert tcp any any -> $HOME_NET any (msg:"Possible Queso Fingerprint attempt"; flags: S12;)
```

This problem was recognized and the sophistication of these types of rules was increased to not produce false positives because ECN was being used[43].

The second most frequent OOS alert, 2.06%, was due to packets with no flags set, i.e. "********". These are probably due to fragmented packets. Detect #2 in Section 2.2 discusses this type of packet in detail.

The Xmas packets are interesting as they have all TCP flags set and possibly the ECN ones as well. These are detected by the following Snort rule:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN XMAS"; flags:SRAFPU,12; reference:arachnids,144;
classtype:attempted-recon; sid:625; rev:2;)
```

According to the Snort Signature Database[44], this alert is generated when TCP packets have the ACK, FIN, PSH, RST, SYN, and URG control bits were set. Typically thise type of packet is associated with system recon since different operating systems will respond in different ways depending on their particular stack implementation, which allows attackers to determine things such as open/closed ports, ACLs, and the like.

There are two such Xmas packets found in the OOS files, namely the 1*UAPRSF and 12UAPRSF packets listed in Table 13. The associated SCAN XMAS alerts are:

```
10/05-12:00:26.174228 24.35.51.121:0 -> MY.NET.29.3:1748
TCP TTL:116 TOS:0x0 ID:44545 IpLen:20 DgmLen:40 DF
1*UAPRSF Seq: 0x5007A9  Ack: 0xB4C770FF  Win: 0x5010  TcpLen: 24  UrgPtr: 0x919
TCP Options (1) => EOL

10/02-13:55:01.255408 68.50.218.176:1679 -> MY.NET.12.7:443
TCP TTL:112 TOS:0x0 ID:19229 IpLen:20 DgmLen:40 DF
12UAPRSF Seq: 0x2F564C  Ack: 0xD6CCABEF  Win: 0x5010  TcpLen: 0  UrgPtr: 0x3C5
```

The first packet is interesting since the source IP of 24.35.51.121 is an old friend that appears in the list of Top Source IPs in the "Null scan!" Alerts (Table 24). Table 14 shows what this host was up to over the period covered in this report.

Time	Source IP	Source Port	Destination IP	Destination Port	Remarks
10/04-18:17:15	24.35.51.121	0	MY.NET.24.74	4193	NULL ********
10/04-18:17:15	24.35.51.121	0	MY.NET.24.74	4193	Null scan!
10/04-18:17:16	24.35.51.121	4201	MY.NET.24.74	443	SYN ******S*
10/04-18:17:18	24.35.51.121	4193	MY.NET.24.74	443	SYN ******S*
10/04-18:18:22	24.35.51.121	4244	MY.NET.29.3	80	SYN ******S*
10/04-18:18:23	24.35.51.121	4243	MY.NET.29.3	80	INVALIDACK **UAP*SF
10/04-18:36:03	24.35.51.121				spp_portscan:PORTSCAN DETECTED (STEALTH)
10/04-18:36:07	24.35.51.121				spp_portscan:portscan status
10/04-18:36:10	24.35.51.121				spp_portscan:portscan status
10/04-18:36:14	24.35.51.121				spp_portscan:End of portscan
10/04-18:38:06	24.35.51.121				spp_portscan:PORTSCAN DETECTED (STEALTH)
10/04-18:38:08	24.35.51.121				spp_portscan:portscan status
10/04-18:38:11	24.35.51.121				spp_portscan:End of portscan
10/05-11:32:52	24.35.51.121	0	MY.NET.29.3	1381	NOACK *2U***S* RESERVEDBITS
10/05-11:50:42	24.35.51.121				spp_portscan:PORTSCAN DETECTED (STEALTH)
10/05-11:50:44	24.35.51.121				spp_portscan:portscan status
10/05-11:50:49	24.35.51.121				spp_portscan:End of portscan
10/05-12:00:25	24.35.51.121	1749	MY.NET.29.3	80	SYN ******S*

Time	Source IP	Source Port	Destination IP	Destination Port	Remarks
10/05-12:00:26	24.35.51.121	0	MY.NET.29.3	1748	FULLXMAS 1*UAPRSF RESERVEDBITS
10/05-12:00:26	24.35.51.121	1748	MY.NET.29.3	80	INVALIDACK *2UA*RS* RESERVEDBITS
10/05-12:00:26	24.35.51.121	0	MY.NET.29.3	1748	SYN ******S*
10/05-12:00:29	24.35.51.121	1750	MY.NET.29.3	80	NULL ********
10/05-12:00:29	24.35.51.121	1748	MY.NET.29.3	80	Null scan!
10/05-12:00:29	24.35.51.121	1748	MY.NET.29.3	80	Null scan!
10/05-12:16:56	24.35.51.121				spp_portscan:PORTSCAN DETECTED (STEALTH)
10/05-12:17:02	24.35.51.121				spp_portscan:portscan status
10/05-12:17:07	24.35.51.121				spp_portscan:portscan status
10/05-12:17:12	24.35.51.121				spp_portscan:End of portscan

Table 14. Activity of source IP of 24.35.51.121

We know the following about the source IP 24.35.51.121:

1. It's named "cmu-24-35-51-121.mivlmd.cablespeed.com".
2. cablespeed.com is headquartered in Millersville, MD and is allocated the network block 24.35.0.0 - 24.35.127.255
3. It does not appear as an attacker in the DShield database.
4. The TTL=116 in the OOS packet, so assuming that the TTL is not crafted and given the geographical proximity of the source and destination hosts, it is likely that the source host is a Windows NT 4.0 or Windows 98 box[37].

We know the following about the destination IPs:

1. The destination IP MY.NET.24.74 is a web server that offers webmail via SSL.
2. The destination IP MY.NET.29.3 is a web server that offers students courses over the Internet.

Looking at the traffic reported in Table 14 we see the following oddities in the traffic:

1. Traffic with source port 0 is sent to high ports on both web servers. Now in RFC 1700, tcp/0 is supposed to be a reserved port so we do not expect to see especially since a client normally uses an ephemeral port when connecting to a server. Programs such as nmap and hping2[45] allow a user to easily craft arbitrary packets including specifying a desired source port. With a source port of 0 and no flags being set the sender of the packet may be hoping to get through any security devices and elicit a response from the destination IP for OS fingerprinting purposes.

2. The source IP sends packets to certain ephemeral ports and then turns around and uses those same ephemeral ports as source ports. For example packets are sent from 0 -> 4193 and then from 4193 -> 443, and from 0 -> 1748 and then from 1748 -> 80. If seems that user probes for a server like eDonkey and them turns around and tries sending traffic as if he is now the server.

The conclusion looking at the activity of the source IP of 24.35.51.121 is that this host is trying to fingerprint the OS and applications on the two web servers. As this

reconnaissance may be the precursor to an attack based on accurate OS and application fingerprinting, the patches levels of the two web server ought to be reviewed and the security devices configured to drop packets with a source port of 0. As well future activity from the source IP of 24.35.51.121 ought to be reviewed and its activities reported to abuse@cablespeed.com.

3.6 Detects prioritized by number of occurrences

Table 15 shows the total alert statistics for the top 10 signatures, source IP addresses and ports, and destination IP addresses and ports for the "non-ICMP (non-spp_portscan)" alerts over the five-day period in this audit. Each of the top 10 signatures will be examined in more detail.

	Totals (Period Oct 1 to Oct 5)	
Top Signature IDs	Name	Count
	SMB Name Wildcard	902,224
	MY.NET.30.4 activity	50,224
	Incomplete Packet Fragments Discarded	7,604
	MY.NET.30.3 activity	7,216
	High port 65535 udp - possible Red Worm - traffic	5,214
	High port 65535 tcp - possible Red Worm - traffic	3,824
	Null scan!	2,903
	Tiny Fragments - Possible Hostile Activity	2,375
	EXPLOIT x86 NOOP	1,462
	connect to 515 from outside	1,198
Top Source IP	Address	Count
	MY.NET.162.118	846,994
	MY.NET.150.133	38,097
	68.65.100.189	35,974
	MY.NET.66.33	5,667
	MY.NET.42.6	5,241
	138.88.168.198	2,606
	220.99.94.77	2,529
	MY.NET.11.6	2,335
	202.188.114.50	2,165
	68.48.217.68	2,125
Top Source Port	Port	Count
	1026	423,811
	1025	422,913
	4043	18,569
	4041	16,968
	137	13,873
	1560	12,296
	1693	7,021
	65535	4,491
	1030	2,534
	0	2,450
Top Destination IP	Address	Count
	MY.NET.30.4	50,563
	MY.NET.30.3	5,260
	MY.NET.30.3	3,639
	MY.NET.66.2	2,201
	MY.NET.30.4	1,722

	Totals (Period Oct 1 to Oct 5)	
	Name	Count
	146.82.109.220	1,591
	MY.NET.163.76	1,558
	146.82.109.225	1,436
	128.183.110.242	1,384
	199.72.154.71	922
Top Destination Port	Port	Count
	137	901,292
	51443	43,775
	524	8,020
	65535	4,590
	80	3,973
	0	2,749
	6257	2,553
	8009	2,120
	515	1,883
	3019	648

Table 15. Top N Statistics for the “non-ICMP (non-spp_portscan)” Alerts

Table 16 shows a breakdown of the “non-ICMP (non-spp_portscan)” alert statistics for each day of the five-day period.

	File 1 (Oct 1 Wed)		File 2 (Oct 2 Thu)		File 3 (Oct 3 Fri)		File 4 (Oct 4 Sat)		File 5 (Oct 5 Sun)	
Signature	Name	Count	Name	Count	Name	Count	Name	Count	Name	Count
	SMB Name Wildcard	312,168	SMB Name Wildcard	380,741	SMB Name Wildcard	198,604	MY.NET.30.4 activity	38,682	SMB Name Wildcard	7,423
	MY.NET.30.4 activity	3,893	MY.NET.30.4 activity	1,819	Incomplete Packet Fragments Discarded	4,627	SMB Name Wildcard	3,288	MY.NET.30.4 activity	4,108
	MY.NET.30.3 activity	2,634	High port 65535 udp - possible Red Worm - traffic	1,253	Tiny Fragments - Possible Hostile Activity	2,344	Incomplete Packet Fragments Discarded	1,997	High port 65535 tcp - possible Red Worm - traffic	2,467
	High port 65535 udp - possible Red Worm - traffic	1,690	connect to 515 from outside	1,055	Null scan!	2,251	MY.NET.30.3 activity	1,165	MY.NET.30.3 activity	1,330
	connect to 515 from inside	694	MY.NET.30.3 activity	942	MY.NET.30.4 activity	1,722	High port 65535 udp - possible Red Worm - traffic	712	High port 65535 udp - possible Red Worm - traffic	675
	SUNRPC highport access!	293	Incomplete Packet Fragments Discarded	756	MY.NET.30.3 activity	1,145	Null scan!	457	EXPLOIT x86 NOOP	542
	NMAP TCP ping!	247	High port 65535 tcp - possible Red Worm - traffic	601	High port 65535 udp - possible Red Worm - traffic	884	High port 65535 tcp - possible Red Worm - traffic	174	Incomplete Packet Fragments Discarded	220
	High port 65535 tcp - possible Red Worm - traffic	217	EXPLOIT x86 NOOP	425	High port 65535 tcp - possible Red Worm - traffic	365	EXPLOIT x86 NOOP	108	Possible trojan server activity	195
	EXPLOIT x86 NOOP	165	NMAP TCP ping!	251	EXPLOIT x86 NOOP	222	NMAP TCP ping!	85	NMAP TCP ping!	101
	Possible trojan server activity	77	Possible trojan server activity	217	NMAP TCP ping!	181	connect to 515 from outside	78	External RPC call	89
Source IP	Address	Count	Address	Count	Address	Count	Address	Count	Address	Count
	MY.NET.162.118	292,882	MY.NET.162.118	361,206	MY.NET.162.118	192,635	68.65.100.189	35,974	MY.NET.66.33	3,363
	MY.NET.150.133	18,000	MY.NET.150.133	16,271	MY.NET.150.133	3,826	MY.NET.42.6	2,331	MY.NET.42.6	2,910
	138.88.168.198	2,606	MY.NET.66.33	1,840	202.188.114.50	2,165	24.104.7.195	1,769	68.48.217.68	2,125
	68.55.105.5	1,673	68.81.2.19	778	220.99.94.77	2,162	MY.NET.11.6	670	MY.NET.84.143	843
	MY.NET.163.76	843	131.118.229.7	730	MY.NET.42.8	905	68.57.90.146	572	141.157.9.122	801
	68.55.158.79	744	MY.NET.163.76	608	MY.NET.21.50	782	68.55.62.79	490	MY.NET.11.6	747
	MY.NET.162.41	692	68.55.62.79	404	MY.NET.21.37	746	220.99.94.77	367	217.132.44.109	675
	MY.NET.66.33	464	MY.NET.42.4	391	MY.NET.21.92	580	MY.NET.21.92	333	68.55.62.79	541
	68.55.52.234	373	MY.NET.11.6	340	MY.NET.21.79	565	MY.NET.21.67	328	194.199.203.7	456
	68.55.62.79	343	68.55.57.218	338	MY.NET.21.116	540	MY.NET.21.69	304	210.6.2.205	431
Source Port	Port	Count	Port	Count	Port	Count	Port	Count	Port	Count
	1026	146,860	1025	180,843	1026	96,589	4043	18,569	137	4364
	1025	146,022	1026	180,362	1025	96,048	4041	16,968	65535	1573
	1693	7,021	1560	8,167	0	2,029	1030	1,314	1028	1523
	1560	4,129	137	4,379	137	1,329	137	896	1030	812
	137	2,905	65535	870	1029	630	1029	618	4016	750
	1551	2,047	3496	863	65535	616	1028	538	3672	719
	2703	1,656	721	730	1227	538	65535	520	1029	675
	2462	1,040	2385	727	1030	408	0	421	4058	614
	2186	1,010	6257	647	6257	404	3010	381	3806	420
	65535	912	3343	577	1028	255	6257	263	4014	409
	1026	146,860	1025	180,843	1026	96,589	4043	18,569	137	4364
Destination IP	Address	Count	Address	Count	Address	Count	Address	Count	Address	Count
	MY.NET.30.4	3,887	206.24.190.158	1,840	MY.NET.66.2	2,201	MY.NET.30.4	38,682	MY.NET.30.4	4,107
	MY.NET.30.3	2,630	MY.NET.30.4	1,817	MY.NET.30.4	1,722	MY.NET.30.3	1,165	146.82.109.220	1,591
	MY.NET.163.76	779	MY.NET.24.15	1,055	MY.NET.30.3	1,145	169.254.0.0	674	146.82.109.225	1,436
	128.183.110.242	692	MY.NET.30.3	939	MY.NET.70.176	345	MY.NET.70.176	375	MY.NET.30.3	1,329
	199.72.154.71	461	MY.NET.163.76	520	64.94.189.7	325	MY.NET.66.2	368	217.132.44.109	842
	MY.NET.24.8	324	169.254.0.0	340	169.254.0.0	318	169.254.45.176	153	169.254.0.0	750
	219.31.76.94	285	MY.NET.150.6	262	169.254.45.176	238	MY.NET.84.143	81	MY.NET.84.143	683
	169.254.0.0	267	68.101.218.125	201	MY.NET.150.6	118	MY.NET.24.15	78	MY.NET.97.25	459
	61.199.46.200	182	219.173.131.15	195	210.194.220.239	100	MY.NET.12.6	73	210.6.2.205	420
	167.206.156.241	142	169.254.45.176	183	MY.NET.1.3	92	217.132.44.109	65	MY.NET.70.176	369
Destination Port	Port	Count	Port	Count	Port	Count	Port	Count	Port	Count
	137	311,835	137	380,341	137	198,405	51443	37,675	137	7,423
	51443	3,296	51443	1,185	0	2,015	137	3,288	8009	2,120
	524	2,723	515	1,055	524	1,295	524	1,616	65535	1,573
	65535	1,010	65535	987	51443	929	80	572	524	1,399
	6257	783	524	987	80	814	0	420	80	1,080
	515	750	80	895	65535	643	65535	377	51443	690
	80	612	6257	574	6257	425	6257	374	3019	648
	32771	295	27374	178	111	118	25	134	3672	542
	53	167	53	159	53	96	515	78	3806	431
	25	103	111	149	25	61	135	58	6257	397

Table 16. Daily Statistics for the “non-ICMP (non-spp_portscan)” Alerts

3.6.1 SMB Name Wildcard Signature (902,224 hits)

Typical alert from the "SMB Name Wildcard" signature look as follows:

```
10/05-00:49:10.666640  [**] SMB Name Wildcard [**] MY.NET.153.21:3273 -> 66.171.157.127:137
10/05-00:49:30.813710  [**] SMB Name Wildcard [**] MY.NET.11.6:137 -> 169.254.0.0:137
```

Examining the output of the "SMB Name Wildcard" signature that was triggered 902,224 times we can see that:

1. The alerts involve traffic to port tcp/137

2. There are no detects caused by traffic from an external host to an internal host, i.e. the source in all alerts is MY.NET.x.x. Table 17 shows the top source IP addresses for "SMB Name Wildcard" signature, which accounts 99.6% of the source addresses.

Source Address	Count
MY.NET.162.118	847,129
MY.NET.150.133	38,097
MY.NET.66.33	5,667
MY.NET.42.6	5,251
MY.NET.11.6	2,335

Table 17. Top Source IP Addresses for "SMB Name Wildcard" Signature

3. In general most destination addresses receive only one hit while the maximum count for a single destination address is 461. Looking at the alerts, we can see that in general the source addresses are scanning as a worm would do.

As there are no detects caused by traffic from an external host to an internal host, either there is a security device on the network that is blocking inbound traffic to port tcp/137 or else the rule was written to only trigger on traffic from internal hosts to port tcp/137 on external hosts.

Some worms use tcp/137 for transmission, e.g. W32.Nimda.A@mm[8]. The hosts listed in Table 17 should be scanned for worms and both outbound and inbound traffic to the Microsoft ports of 137-139 and 445 both udp and tcp should be blocked.

3.6.2 "MY.NET.30.4 activity" Signature (50,224 hits)

The "MY.NET.30.4 activity" signature is a custom one that alerts on connections made to an Internet assessable host with an IP address of MY.NET.30.4. The top destination ports accounting for 99.83% of this signature over the five-day period are shown in Table 18.

The following shows a typical alert for the "MY.NET.30.4 activity" signature:

```
10/05-00:08:12.699474  [**] MY.NET.30.4 activity [**] 68.55.62.79:1036 -> MY.NET.30.4:524
```

It can be seen that Internet hosts are attempting connections to ports that are typically not exposed to the Internet. Whether these attempt succeed is not clear since we do not have traces to examine.

Port 524 is typically used to access Netware Directory Services (NDS) on a NetWare server. The destination port of 51443 is associated with the Novell NetStorage/iFolder feature running on an Apache webserver on NetWare. The default port number for NetWare Enterprise Server is 80 for HTTP and 443 for HTTPS. If the NetWare Enterprise Server installed, by default the Apache Web Server will get port 51080 for HTTP and 51443 for HTTPS[7].

Hence it would appear that the "MY.NET.30.4 activity" signature monitors connections to a NetWare 6.x server that is exposed on the Internet.

Destination Port	Count
51443	43,775
80	2,609
8009	2,123
524	1,608
17300	17

Table 18. Activity to Destination Ports on MY.NET.30.4

The purpose of this signature is not clear. If host MY.NET.30.4 is special then the nature of access to this server from the Internet should be reviewed and tightened up if possible. Of course it is also possible that the host MY.NET.30.4 is a honey pot just as host MY.NET.30.3 might be.

3.6.3 Incomplete Packet Fragments Discarded (7,604 hits)

This signature is triggered because packet fragments were detected but not all the packets arrived so the stream could not be reassembled[9].

These alerts look as follows:

```
10/05-10:45:33.342218  [**] Incomplete Packet Fragments Discarded [**] 80.135.88.105:0 -> MY.NET.84.143:0
10/03-05:24:47.480493  [**] Incomplete Packet Fragments Discarded [**] MY.NET.21.68 -> 206.47.132.111
```

Note that there are the following two types of events:

1. If the source is an external address and the destination is a MY.NET.x.x addess then the source and destination ports are both 0.

2. If the source is a MY.NET.x.x addess and the destination is an external address then there are no source and destination ports recorded.

There are 7,604 hits for this signature over the five-day period.

1. On Oct 1, Wed all source IPs were outside ones and there were only 4 hits for this signature.

2. From Oct 2 to Oct 4 all the top source IPs were inside ones.

3. On Oct 5, Sun all source IPs were outside ones.

The top sources of the "Incomplete Packet Fragments Discarded" alerts are shown in Table 19. We can see that the top sources are all inside ones.

Source IP	Count
MY.NET.21.50	1,090
MY.NET.21.92	1,016
MY.NET.21.67	958
MY.NET.21.37	936
MY.NET.21.79	769
MY.NET.21.116	739
MY.NET.21.69	723
MY.NET.21.70	542
MY.NET.21.68	529

Table 19. Top Sources of "Incomplete Packet Fragments Discarded" Alerts

The top destinations of the "Incomplete Packet Fragments Discarded" alerts for each day are shown in Table 20. We see that most of the top destinations are outside ones.

Destination IP	Count	Date
64.62.132.135	2,360	3-Oct
213.249.98.11	1,142	4-Oct
212.71.43.58	1,006	3-Oct
69.39.225.230	837	4-Oct
69.31.67.217	725	2-Oct
203.135.43.233	716	3-Oct
206.47.132.111	515	3-Oct
MY.NET.153.31	168	5-Oct
MY.NET.12.4	3	1-Oct

Table 20. Top Destinations of "Incomplete Packet Fragments Discarded" Alerts

A search through the log files yielded no other traffic associated with some of the source addresses. For example the source IP address of 80.135.88.105 found in the alert shown above appears only once in the alert files over the five-day period, i.e. in "Incomplete Packet Fragments Discarded" alert itself. It could be speculated that the alerts with a source and destination port of 0 are some from of attack of reconnaissance attempts. However, this would correspond to a very low and slow scanning rate by a hacker. The other difficulty with these alerts is the fact that they are the top types pointed out above.

Given these facts, a more likely explanation for the alerts is a bug in the version of Snort that is running. Martin Roesch states[10] that this type of alert "means that you're using the defrag preprocessor instead of the newer frag2 preprocessor and that you should switch to frag2. The defrag preprocessor had some fairly nasty failure modes and has since been superceded by frag2, so I'd recommend using that for now."

Hence it is highly likely that this alert is currently mostly noise and the frag2 preprocessor should be used to eliminate the noise or else the signature disabled.

3.6.4 MY.NET.30.3 activity (7,216 hits)

The "MY.NET.30.3 activity" signature is a custom one that alerts on connections made to an Internet assessable host with an IP address of MY.NET.30.3. The top destination ports accounting for 99.83% of this signature over the five-day period are shown in Table 21.

Destination Port	Count
524	6,412
3019	648
80	62
17300	16
443	11

Table 21. Activity to Destination Ports on MY.NET.30.3

The following shows a typical alert for the "MY.NET.30.3 activity" signature:

```
10/05-09:42:47.454555  [**] MY.NET.30.3 activity [**] 68.55.62.79:1435 -> MY.NET.30.3:524
```

It can be seen that Internet hosts are attempting connections to ports that are typically not exposed to the Internet. Whether these attempt succeed is not clear since we not have traces to examine.

Again port 524 is typically used to access Netware Directory Services (NDS) on a NetWare server. Novell BorderManager version 3.6 uses tcp/3019 for Resource Manager[13.]. The default port number for NetWare Enterprise Server is 80 for HTTP and 443 for HTTPS.

Hence it would appear that the "MY.NET.30.3 activity" signature monitors connections to a Novell BorderManager system that is on the Internet.

The tcp/17300 activity could be due to hosts probing for backdoors opened by "Kuang2 The virus".

The purpose of this signature is not clear. If host MY.NET.30.3 is special then the nature of access to this server from the Internet should be reviewed and tightened up if possible. Of course it is also possible that the host MY.NET.30.3 is a honey pot just as host MY.NET.30.4 might be.

3.6.5 High port 65535 tcp & udp - possible Red Worm – traffic (9,038 hits)

Both the “High port 65535 udp - possible Red Worm” alerts (5,214 hits) and the “High port 65535 tcp - possible Red Worm” alerts (3,824 hits) will be considered together.

There is no rule for the Red Worm/Adore worm in the rule sets that I examined so it may be a custom signature triggered simply by traffic to or from tcp or udp port 65535. In any events these alerts look as follows:

```
10/05-00:06:59.284331  [**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.70.176:6257 ->
61.199.1.118:65535
10/05-00:06:59.698369  [**] High port 65535 udp - possible Red Worm - traffic [**] 61.199.1.118:65535 ->
MY.NET.70.176:6257
```

Red Worm is the original name of the Adore worm and it is similar to the Ramen and Lion worms. It started to spread in April 2001. Adore scans the Internet checking Linux hosts to determine whether they are vulnerable to any of the following well-known exploits: LPRng, rpc-statd, wu-ftpd and BIND. LPRng is installed by default on Red Hat 7.0 systems[22].

Adore does several things including the following:

- Attempts to send an email to several addresses including adore9000@21cn.com.
- Installs a Trojan backdoor that activates when it receives a ping packet with correct size, and then opens a shell in the port 65535[23].

Since the ephemeral port range is 1024 through 65535, and the rule is triggering on traffic to or from tcp/65535, we can expect many false positives. The two characteristics listed above for the Adore worm give us a way to check for false positives, i.e. the infected hosts that have triggered the “possible Red Worm” alert can be cross-correlated with hosts that have sent smtp traffic to the worm related mail servers. Unfortunately we do not have access to the traces that would allow such correlation, but we can look for connections from tcp/65535 to tcp/25 in the “Red Worm” alerts.

Table 22 shows the internal hosts that have triggered both the “possible Red Worm” alert and have also apparently sent e-mail to external hosts. As was mentioned, this combination could be a strong indicator that the internal hosts are infected with the adore worm. If the resolved name corresponds to that of a known mail exchanger then it is likely that the internal hosts is mail server and the “possible Red Worm” alerts are false positives and need not be checked. However if the resolved name does not appear to be that of a known mail exchanger then the internal hosts need to be checked to see if the are running Linux and if they are in fact infected by the Adore worm. The internal hosts that need to be checked are indicated by a “Y” in the “Check?” column if they are running Linux.

A destination IP of 127.0.0.2 could indicate that the source IP is a mail server using Real-time spam Black Lists (RBL).

Source IP	Source Port	Destination IP	Resolved Name	Dst Port	Count	Check ?
MY.NET.100.230	65535	127.0.0.2	N/A	25	15	N
MY.NET.100.13	65535	143.128.64.3	unpsun2.cc.unp.ac.za	25	3	Y
MY.NET.25.11	65535	149.174.40.6	siaag1ad.compuserve.com	25	5	Y
MY.NET.100.230	65535	150.217.15.247	not resolved	25	6	Y
MY.NET.25.10	65535	194.242.43.19	not resolved	25	2	Y
MY.NET.25.67	65535	194.67.18.128	relay2.aport.ru	25	3	N
MY.NET.25.67	65535	202.214.130.2	not resolved	25	2	Y
MY.NET.100.13	65535	203.151.37.1	not resolved	25	1	Y
MY.NET.25.67	65535	205.158.62.72	spf10.us4.outblaze.com	25	3	Y
MY.NET.25.68	65535	208.18.122.165	parker1.sprint.com	25	2	Y
MY.NET.25.10	65535	208.20.220.60	not resolved	25	10	Y
MY.NET.25.66	65535	209.202.214.116	smtp-06.sc8.finance.lycos.com	25	4	N
MY.NET.25.73	65535	209.202.214.117	smtp-07.sc8.finance.lycos.com	25	3	N
MY.NET.25.72	65535	209.204.62.47	pn13.essoc.com	25	4	Y
MY.NET.100.230	65535	209.241.185.109	not resolved	25	6	Y
MY.NET.25.69	65535	216.35.70.232	mailserver2.iexpect.com	25	2	N
MY.NET.24.20	65535	217.43.24.119	host217-43-24-119.range217-43.btcentralplus.com	25	4	Y
MY.NET.25.66	65535	64.0.64.130	host130.netreds.net	25	2	Y
MY.NET.25.68	65535	64.119.222.7	not resolved	25	5	Y
MY.NET.25.68	65535	64.154.80.196	postal.websidestory.com	25	2	N
MY.NET.25.10	65535	64.157.4.78	mta-v22.level3.mail.yahoo.com	25	3	N
MY.NET.24.20	65535	64.94.110.11	not resolved	25	8	Y
MY.NET.25.12	65535	65.54.200.30	support.msn.com	25	3	N
MY.NET.25.71	65535	80.201.209.168	168.209-201-80.adsl.skynet.be	25	2	Y
MY.NET.25.72	65535	81.218.218.134	bzq-218-218-134.red.bezeqint.net	25	5	Y

Table 22. Internal Hosts that triggered “possible Red Worm” Alert and sent mail

Table 23 shows “High port 65535 tcp & udp - possible Red Worm” activity of the top sources. Of these sources the ones the appears to merit further interest are:

1. MY.NET.163.76 and MY.NET.152.21 - Always used a source port of udp/6257 over the five-day period to connect to various external hosts on udp/65535.

2. MY.NET.70.176 - Always used a source port of udp/6257 over the five-day period to connect to various external hosts on udp/65535. The destination host resolves to ip68-0-189-136.tc.ph.cox.net, which appears to be a client’s host.

3. MY.NET.84.232 - Always used a source port of udp/3383 over the five-day period to connect to various external hosts on udp/65535.

4. MY.NET.84.143 - This host triggered the both the udp and tcp forms of this alert both involving the same destination host that resolves to DSL217-132-44-109.bb.netvision.net.il, which appears to be a client’s host. This host used source ports udp/4672 and tcp/4672 over the five-day period to connect to a single external host on udp/65535.

High port 65535 tcp					
Source IP	Source Port	Count	Destination IP	Resolved Name	Check?
MY.NET.84.143	various	892	217.132.44.109	DSL217-132-44-109.bb.netvision.net.il	Y
MY.NET.97.25	3806	420	210.6.2.205	210006002205.ctinets.com	Y
MY.NET.83.109	various	273	68.101.218.125	ip68-101-218-125.sd.sd.cox.net	Y
MY.NET.112.152	various	86	172.184.181.157	ACB8B59D.ipt.aol.com	Y
MY.NET.162.87	80	63	68.55.192.222	pcp229411pcs.catonv01.md.comcast.net	Y
High port 65535 udp					
Source IP	Source Port	Count	Destination IP	Resolved Name	Check?
MY.NET.163.76	6257	1555	various	N/A	Y
MY.NET.70.176	6257	806	68.0.189.136	ip68-0-189-136.tc.ph.cox.net	Y
MY.NET.152.21	6257	29	various	N/A	Y
MY.NET.84.232	3383	27	various	N/A	Y
MY.NET.84.143	4672	21	217.132.44.109	DSL217-132-44-109.bb.netvision.net.il	Y

Table 23. "High port 65535 tcp & udp - possible Red Worm" Activity

All the evidence considered, we conclude that some of the hosts may be infected with the Adore worm so the following steps are recommended:

a. The internal hosts that need to be checked are those indicated in Table 22 and Table 23 by a "Y" in the "Check?" column if they are running Linux.

b. Patch all Linux hosts against the Adore worm.

c. Block access to the destination hosts in Table 22 and Table 23 if investigation shows that any are associated with the Adore worm.

3.6.6 Null scan! (2,903 hits)

This signature is presumably based on the detection of a TCP frame with a sequence number of zero and all control bits are set to zero. This frame should never be seen in normal TCP operation. An attacker may scan hosts by sending these specially formatted frames to see what services are available and susceptible to attack. This is sometimes done in preparation for a future attack[17].

A NULL scan attack is looking for a RST from the target when the port is closed or no response which might mean the port is open[18].

The "Null scan!" rule is probably similar to the following current rule:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN NULL"; flags:0; seq:0; ack:0; reference:arachnids,4;
classtype:attempted-recon; sid:623; rev:1;)
```

These alerts look as follows:

```
10/02-11:46:56.538963  [**] Null scan! [**] 67.119.232.52:5384 -> MY.NET.12.4:110
10/02-12:18:09.554702  [**] Null scan! [**] 217.136.213.195:0 -> MY.NET.84.232:0
```

Table 24 lists the top source IPs in the "Null scan!" alerts. Given that the host 220.99.94.77 accounts for 87% of the "Null scan!" alerts, we'll examine its activities in more detail.

Top Source IP	Count
220.99.94.77	2,523
67.119.232.52	138
63.251.52.75	119
218.75.129.126	37
202.188.114.50	23
202.224.226.108	7
206.14.191.84	6
129.44.176.112	5
24.35.51.121	4

Table 24. Top Source IPs in the "Null scan!" Alerts

Table 25 shows the top destination IPs in "Null scan!" alerts generated by the host 220.99.94.77. This address resolves to host220099094077.cti-now.co.jp, which belongs to City Trust and Investment CO.,Ltd.

Top Destination IP	Count
MY.NET.66.2;0	2,271
MY.NET.66.2;19697	13
MY.NET.66.2;4634	10
MY.NET.66.2;9423	9
MY.NET.66.2;55678	8
MY.NET.66.2;1322	8
MY.NET.66.2;4672	6
MY.NET.66.2;20697	6

Table 25. Top destination IPs in "Null scan!" Alerts by 220.99.94.77

All of the 2,523 alerts that the host 220.99.94.77 triggered are targeted against the internal host MY.NET.66.2. The reason that host 220.99.94.77 is targeting this host is unknown but clearly this needs to be investigated and any vulnerabilities patched.

The host 220.99.94.77 scans destination ports that include both low and high ports. However it used a source and destination port of 0 in 78% of its probes of the internal host MY.NET.66.2. These alerts look as follows and are typically associated with OS fingerprinting by programs such as nmap:

```
10/04-15:39:48.677437 ;Null scan!;220.99.94.77;0;MY.NET.66.2;0
```

As well host 220.99.94.77 has triggered the following alerts:

1. Probable NMAP fingerprint attempts, e.g.

```
10/03-16:26:48.230070  [**] Probable NMAP fingerprint attempt [**] 220.99.94.77:3 -> MY.NET.66.2:38383
10/03-16:19:28.751726  [**] Probable NMAP fingerprint attempt [**] 220.99.94.77:31088 -> MY.NET.66.2:64144
```

2. Portscan alerts, e.g.

```
10/03-17:10:08.912714  [**] spp_portscan: portscan status from 220.99.94.77: 7 connections across 1 hosts: TCP(7), UDP(0) STEALTH [**]
10/03-17:10:14.321651  [**] spp_portscan: End of portscan from 220.99.94.77: TOTAL time(3s) hosts(1) TCP(10) UDP(0) STEALTH [**]
```

All the evidence considered, we can conclude that the host 220.99.94.77 is an attacker. The threat posed by it is not clear as we did not have a trace showing if the internal host MY.NET.66.2 responded. In any event, the following steps are recommended:

1. Determine whether the “Null scan” is getting into the internal network.

2. Contact the abuse address of cti-now.co.jp and notify them of the activity of the host 220.99.94.77.

3. Block the host 220.99.94.77 at the perimeter and log any denies for a period of several weeks to see if further malicious activity occurs.

3.6.7 Tiny Fragments - Possible Hostile Activity (2,375 hits)

The “Tiny Fragments - Possible Hostile Activity” rule is probably similar to the following current rule, which looks for packets with a payload less than 25 bytes, i.e. dsize: < 25:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Tiny Fragments"; fragbits:M; dsize: < 25; classtype:bad-unknown; sid:522; rev:1;)
```

So typically this event is generated when an IPv4 fragment of dubiously small nature is detected. Many IDS's are known to have issues regarding the reassembly of IP fragments, and could miss an attack carried over such means. Some firewalls suffer from the same issues and can be tricked into allowing packets through that should normally be rejected. Furthermore, there is a small history of OS issues related to unorthodox fragmentation. There is no piece of commercial network equipment that fragments packets in sizes smaller than 512 bytes[21]. However, tools have been written to trivially fragment traffic, e.g. Dug Song's fragrouter program is a well-known example[20].

In our case, these alerts look as follows:

```
10/01-17:23:04.963326 ;Tiny Fragments - Possible Hostile Activity;206.14.191.84;MY.NET.53.183
10/02-16:01:27.890324 ;Tiny Fragments - Possible Hostile Activity;66.68.188.86;MY.NET.84.232
10/03-20:23:47.373552 ;Tiny Fragments - Possible Hostile Activity;202.188.114.50;MY.NET.70.197
```

Table 26 shows the top source and destination IPs for the “Tiny Fragments” alert. The host 202.188.114.50 accounts for 90% of these alerts and its destination IP is exclusively MY.NET.70.197 (see the example alerts shown above). The threat against the host MY.NET.70.197 needs to be examined in greater depth.

Top Source IP	Count	Top Destination IP	Count
202.188.114.50	2139	MY.NET.70.197	2139
24.191.73.135	107	MY.NET.84.232	116
64.147.47.20	71	MY.NET.84.180	71
206.14.191.84	29	MY.NET.53.183	29
66.68.188.86	15	82.64.27.207	9
MY.NET.84.180	9	MY.NET.97.103	6
67.168.67.44	2	MY.NET.82.86	2
196.41.30.38	1	213.76.44.45	1
80.109.89.222	1	MY.NET.12.6	1
MY.NET.84.232	1	MY.NET.112.164	1

Table 26. Top Source and Destination IPs for the “Tiny Fragments” Alert

Examining the activity against the host MY.NET.70.197, we find the following correlated alerts that show the host 202.188.114.50 is targeting it:

```
10/03-20:17:14.032492  [**] Null scan! [**] 202.188.114.50:0 -> MY.NET.70.197:0
… 14 other similar alerts
10/03-20:19:24.568941  [**] Null scan! [**] 202.188.114.50:0 -> MY.NET.70.197:0
10/03-20:20:17.281509  [**] Probable NMAP fingerprint attempt [**] 202.188.114.50:0 -> MY.NET.70.197:0
10/03-20:20:21.740292  [**] High port 65535 tcp - possible Red Worm - traffic [**] 202.188.114.50:6257 ->
MY.NET.70.197:65535
10/03-20:20:24.271163  [**] Null scan! [**] 202.188.114.50:0 -> MY.NET.70.197:0
10/03-20:20:35.217413  [**] Null scan! [**] 202.188.114.50:0 -> MY.NET.70.197:0
10/03-20:20:39.365149  [**] Null scan! [**] 202.188.114.50:0 -> MY.NET.70.197:0
10/03-20:21:29.762047  [**] Probable NMAP fingerprint attempt [**] 202.188.114.50:0 -> MY.NET.70.197:0
10/03-20:21:46.126118  [**] Null scan! [**] 202.188.114.50:0 -> MY.NET.70.197:0
10/03-20:22:03.818905  [**] Null scan! [**] 202.188.114.50:12 -> MY.NET.70.197:5583
10/03-20:22:11.647656  [**] Null scan! [**] 202.188.114.50:0 -> MY.NET.70.197:0
10/03-20:23:41.721376  [**] Null scan! [**] 202.188.114.50:0 -> MY.NET.70.197:0
```

The host 202.188.114.50 does not have a reverse lookup record, nor does it appear in the DShield database. The APNIC Whois Database reports that this IP address belongs to Telekom Cellular Sdn. Bhd. - Kuala Lumpur.

The following are the recommended actions:

1. Set a packet capture on this host to examine in more detail what it is doing.

2. Contact the abuse address of Telekom Cellular Sdn. Bhd. and notify them of the activity of the host 202.188.114.50.

3. Block the host 202.188.114.50 at the perimeter and log any denies for a period of several weeks to see if further malicious activity occurs.

4. Ensure that the IP stacks of all publicly exposed hosts are not vulnerable to DoS due to the handling of tiny fragments.

3.6.8 EXPLOIT x86 NOOP (1,462 hits)

This signature is presumably based on the detection of the Intel X86 no operation code of 0x90 in the data payload of a packet. The "EXPLOIT x86 NOOP" rule is probably similar to the following current rules involving NOOPs:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"EXPLOIT ssh CRC32 overflow NOOP"; flow:to_server,established;
content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; reference:bugtraq,2347; reference:cve,CVE-2001-0144;
classtype:shellcode-detect; sid:1326;  rev:3;)

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;)
```

These alerts look as follows:

```
10/01-00:32:42.610108  [**] EXPLOIT x86 NOOP [**] 4.47.141.115:4685 -> MY.NET.190.102:135
10/01-14:49:56.881927  [**] EXPLOIT x86 NOOP [**] 217.230.76.18:63922 -> MY.NET.29.19:80
10/01-14:49:28.075247  [**] EXPLOIT x86 NOOP [**] 217.230.76.18:63916 -> MY.NET.5.55:80
```

The function of the NOOP bytes in a buffer overflow attack is described as follows[14]: "If the attacker wants to overflow a certain buffer in a program, he needs to know the exact address of that buffer inside the stack segment of the process memory. In fact, trying to guess the exact address of that buffer is nearly impossible. Therefore a trick is used to increase the chances of getting the address by putting a bunch of NOP bytes, which actually do nothing, in front of the buffer. With this approach, only an address that resides inside the frame of NOP commands needs to be guessed. To give an example, if we would add 100 NOP commands, we would increase our chance to guess a good address by the factor 100."

Table 27 shows the top Source IPs causing the "EXPLOIT x86 NOOP" alert. As well it shows the top Destination IPs and associated ports for this signature. The traffic to tcp/135 (Microsoft RPC service) from external hosts is disconcerting but it is not clear if it actually gets through the security devices to its intended targets. If it is getting through then the access to these internal hosts from the Internet needs to be reviewed and tightened up. If the traffic is not getting through then this signature needs to be tuned to reduce the false-positives.

If the organization is not offering news (tcp/119 - nntp) then again this traffic should be blocked.

Top Source IP	Count	Top Destination IP and Ports	Count
194.199.203.7	455	MY.NET.150.6;80	380
62.45.86.249	262	MY.NET.189.62;80	281
129.142.207.186	118	MY.NET.190.102;135	93
66.98.160.20	78	MY.NET.190.101;135	81
131.118.254.130	58	MY.NET.190.97;135	72
217.230.67.170	45	MY.NET.24.8;119	61
217.230.76.18	34	MY.NET.29.21;80	29
62.194.28.184	21	MY.NET.5.55;80	25
80.126.85.198	18	MY.NET.29.12;80	25

199.184.165.136	16	MY.NET.111.21;80	24

Table 27. Top Source IPs, and Destination IPs and Ports for the "EXPLOIT x86 NOOP" Alert

Since 194.199.203.7 is the top source IP address triggering the "EXPLOIT x86 NOOP" with its alerts accounting for 31% of the total, we'll look at its activities in more detail. Table 28 shows the top destination IP and associated ports of the traffic from 194.199.203.7. Given the nature of the signature and the destination port of tcp/80, it is reasonable to assume that this traffic is meant as buffer overflow attempts against the web servers. For example Windows 2000 IIS 5.0 had a Remote Buffer Overflow Vulnerability that permitted remote system level code execution[15].

Top Destination IP and Ports	Count
MY.NET.189.62;80	281
MY.NET.111.21;80	24
MY.NET.5.44;80	20
MY.NET.5.20;80	20
MY.NET.5.15;80	14
MY.NET.29.18;80	13
MY.NET.29.12;80	12
MY.NET.5.92;80	10
MY.NET.5.46;80	9
MY.NET.5.95;80	8

Table 28. Top Destination IPs and Ports for Source IP of 194.199.203.7

The following actions are recommended:

1. The legitimacy of the traffic causing these alarms needs to be examined by viewing a trace. If this traffic is malicious then the abuse address of the owner of the appropriate IP address block ought to be contacted to notify them of the incidents.

2. Review the patch levels of publicly accessible web servers.

3. Determine the need to block traffic from IPs such as 194.199.203.7.

4. Block the traffic to tcp/135 (Microsoft RPC service) from external hosts.

5. Examine the requirement for traffic to tcp/119 (nntp) from external hosts.

3.6.9 Connect to 515 from outside (1,198 hits)

This signature is based on the detection of the traffic to port tcp/515 which is assigned to the printer spooler, i.e. lpr. The importance of this traffic is that there is a vulnerability in certain lprd implementations that allows remote attackers to cause a buffer overflow and to then execute arbitrary commands[18].

The “connect to 515 from outside” rule is probably a simpler version of the following current rules involving lpr traffic:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 515 (msg:"EXPLOIT LPRng overflow"; flow:to_server,established; content:
"|43 07 89 5B 08 8D 4B 08 89 43 0C B0 0B CD 80 31 C0 FE C0 CD 80 E8 94 FF FF FF 2F 62 69 6E 2F 73 68 0A|";
reference:cve,CVE-2000-0917; reference:bugtraq,1712; classtype:attempted-admin; sid:301;  rev:4;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 515 (msg:"EXPLOIT Redhat 7.0 lprd overflow"; flow:to_server,established;
content:"|58 58 58 58 25 2E 31 37 32 75 25 33 30 30 24 6E|"; classtype:attempted-admin; sid:302;  rev:4;)
```

These alerts look as follows:

```
10/04-20:49:18.875478 ;connect to 515 from outside;68.32.127.158;672;MY.NET.24.15;515
10/01-12:49:00.646805 ;connect to 515 from outside;131.118.229.7;721;MY.NET.24.15;515
```

Table 29 shows the statistics for “connect to 515 from outside” alerts. As can be seen, both hosts 131.118.229.7 and 68.32.127.158 connect exclusively to host MY.NET.24.15. It seem likely that this traffic from these two hosts is allowed through the security devices to MY.NET.24.15 for a business reason.

Source IP connecting to tcp/515	Alert Count	Top Destination IP
131.118.229.7	795	MY.NET.24.15
68.32.127.158	403	MY.NET.24.15

Table 29. Statistics for “connect to 515 from outside” Alerts

The following actions are recommended:

1. Examine the requirement for traffic to tcp/515 (printer) from external hosts and blocked at the perimeter or at least restricted to authorized hosts, which is likely the case here.

2. Review the patch levels of all publicly accessible hosts for vulnerabilities such as lprd overflows.

3.6.10 Possible trojan server activity (489 hits)

While this signature is not in the overall top ten signatures, it is included because of the potential danger posed to the organization if Trojans are operating inside the network.

These alerts look as follows:

```
10/05-08:03:53.468641 ;Possible trojan server activity;141.157.8.192;27374;MY.NET.24.34;80
10/05-10:36:06.652914 ;Possible trojan server activity;MY.NET.24.34;80;69.140.135.254;27374
10/01-03:06:10.742309 ;Possible trojan server activity;217.85.235.152;1385;MY.NET.16.90;27374
10/02-02:42:41.599262 ;Possible trojan server activity;68.81.83.92;3680;MY.NET.190.252;27374
10/02-02:47:20.615167 ;Possible trojan server activity;24.60.194.13;2136;MY.NET.190.4;27374
```

Table 31 shows the basic statistics for “Possible trojan server activity” alerts. The obvious port of concern is tcp/27374 which is typically associated with the SubSeven Trojan or other Trojans such as Ramen[14].

Top Source IP	Count	Top Source Port	Count	Top Destination IP	Count	Top Destination Port	Count
24.62.71.91	97	27374	158	68.55.242.239	35	27374	458
68.81.83.92	77	80	102	MY.NET.100.165	35	80	101
24.60.194.13	57	25	20	68.55.195.148	29	25	25
172.151.131.58	48	443	10	MY.NET.24.34	29	443	7
217.85.235.152	42	2417	3	MY.NET.24.44	25	3536	3
MY.NET.24.44	40	2436	2	MY.NET.12.6	22	3255	2
68.55.195.148	33	2653	2	65.114.173.132	15	2417	2
MY.NET.100.165	31	2654	2	141.157.8.192	7	1998	2
MY.NET.24.34	25	3255	2	MY.NET.6.15	7	143	2
68.55.242.239	21	143	1	24.60.194.13	6	4697	1

Table 30. Statistics for “Possible trojan server activity” Alerts

The first two sample alerts shown above at likely noise, as they appear to be web traffic. The next three sample alerts are of concern since they show traffic from high ports on external hosts to tcp/27374 on internal hosts.

Of the 489 hits on this signature, 325 are of concern. Table 31 shows a breakdown of these alerts over the five-day reporting period. The reasons why a number of external hosts are scanning for the same hosts, e.g. MY.NET.5.5, and networks, e.g. MY.NET.190.x, needs to be investigated. Perhaps these hosts have a Trojan.

It should be noted that there is no traffic from the inside hosts that looks like Trojan scanning traffic. As well none of the “Source IP scanning for Trojans” or “Scanned Destination IP” appears in the OSS report files nor does any traffic to tcp/27374 appear.

Source IP scanning for Trojans	Alert Count	Scanned Destination IP
24.62.71.91	97	MY.NET.6.15
		MY.NET.190.x
68.81.83.92	77	MY.NET.5.5
		MY.NET.6.15
		MY.NET.190.x
24.60.194.13	52	MY.NET.190.x
172.151.131.58	48	MY.NET.6.15
		MY.NET.16.90
		MY.NET.16.114
		MY.NET.190.x
217.85.235.152	42	MY.NET.5.5
		MY.NET.16.90
		MY.NET.190.x
24.60.194.13	5	MY.NET.5.5
		MY.NET.6.15
		MY.NET.16.90
		MY.NET.16.106
		MY.NET.16.114

Source IP scanning for Trojans	Alert Count	Scanned Destination IP
141.156.176.107	4	MY.NET.5.5
		MY.NET.6.15

Table 31. “Possible Trojan server activity” Alerts of Concern

As shown in Section 3.8 , this traffic to tcp/27374 actually gets through the security devices to some of the intended targets. Therefore, the policy enforced by the security devices needs to be reviewed and tightened up.

3.7 Correlations from other sources

The GCIA Practical by Joe Bowling submitted on 20 September 2003 was consulted. It provided correlation in the general sense but not the specific sense since difference logs were used. He found the following 13 tops detects over the period of 27-31 July 2004:

- CS Webserver
- SMB Wildcard
- IIS unicode attack
- Queso fingerprint
- CGI NULL Byte
- Exploit x86 NOOP
- TCP High port
- UDP High port
- tiny fragments
- connect to 515 from outside
- SUNRPC highport access
- IDS552/web-iis_IIS ISAPI
- Null scan

However over the period of this audit, CS Webserver, IIS unicode attack, Queso fingerprint, CGI NULL Byte, SUNRPC highport access and IDS552/web-iis_IIS ISAPI did not appear amongst the top detects. In short during the period of this report, web servers detects were not prominent.

There is strong correlation between Joe’s audit and this audit, for example consider the following:

1. Joe's Top Talker for scans was 130.85.1.3 (1,942,362), which was the same one in this audit with a hit count of 2,753,737.

2. Joe's Top Talker for the Alerts was 68.48.217.68 which also appears in this audit.

3. A number of the external hosts from the same networks listed in Joe' audit were found in this audit, e.g. *.client2.attbi.com and *.sndg02.pacbell.net.

4. Joe found external hosts trying to connect to the well-known Trojan port tcp/27374 on the internal network. This was also seen in this audit and forms the basis of this audit's Link Graph Analysis (see Section 3.8).

3.8 Link Graph Analysis

This analysis will show a relationship among hosts that are targeting the host MY.NET.6.15 that is not readily apparent from looking at the log traces themselves. As seen in Table 32, there are a total of 74 events (alerts, scans and OOS) with a destination of MY.NET.6.15. Of these events, most appear to be related to probes for listening Trojans.

Destination Port	IANA Port Assignment	Possible Service on Port	Event Count
22	ssh	–	2
80	http	–	10
111	sunrpc	–	18
137	Netbios-ns	–	1
554	Unassigned	Real Time Stream Control Protocol	5
1080	socks	Trojan SubSeven 2.2	1
1524	ingreslock	Trojan (Trinoo)	1
2417	Unassigned	Composit Server	2
2540	Unassigned	lonworks	1
3389	Unassigned	MS Terminal Services	1
3625	Unassigned	–	1
3685	Unassigned	–	1
4000	Unassigned	Trojan (Connect-BackBackdoor)	3
4372	Unassigned	–	1
4898	Unassigned	–	1
4899	Unassigned	Radmin (Remote Administrator default port)	1
6112	dtspcd	CDE subprocess control	1
7070	Unassigned	arcp (legacy RealServer port)	2
13240	Unassigned	–	2
17300	Unassigned	Kuang2TheVirus	11
27374	Unassigned	Subseven, Ramen	7
34816	Unassigned	–	1

Table 32. Destination Ports targeted on MY.NET.6.15

From the alerts, scans and OOS_Report files, we see in Table 33 the events triggered by traffic to and from the host MY.NET.6.15, a.k.a 130.85.6.15. Of particular concern is the traffic to destination port tcp/27374 that is responded to by the host MY.NET.6.15. This is high port to high port traffic that could be associated with the SubSeven Trojan or other Trojans such as Ramen[13].

The host MY.NET.6.15 needs to be checked for a Trojan server or other cause of the traffic to and from port tcp/27374. If the system is running a legitimate program on port

tcp/27374, then either the port number of the application should be changed or else the "Possible Trojan server activity" signature tuned to avoid generating false-positives.

Event Details
10/01-03:12:43 194.133.18.72:2750 -> 130.85.6.15:4000 SYN ******S*
10/01-05:48:10 62.72.110.178:1540 -> 130.85.6.15:554 SYN ******S*
10/01-09:44:00 62.254.138.158:1119 -> 130.85.6.15:80 SYN ******S*
10/01-13:37:41 202.125.103.69:1855 -> 130.85.6.15:22 SYN ******S*
10/01-14:18:33 212.182.10.6:3013 -> 130.85.6.15:80 SYN ******S*
10/01-19:58:25 64.152.251.77:3052 -> 130.85.6.15:80 SYN ******S*
10/01-21:45:01 61.143.160.161:4170 -> 130.85.6.15:554 SYN ******S*
10/01-21:59:19.309100 [**] Possible trojan server activity [**] 141.156.176.107:2417 -> MY.NET.6.15:27374
10/01-21:59:20.268429 [**] Possible trojan server activity [**] 141.156.176.107:2417 -> MY.NET.6.15:27374
10/01-21:59:20.268533 [**] Possible trojan server activity [**] MY.NET.6.15:27374 -> 141.156.176.107:2417
10/01-21:59:20.742513 [**] Possible trojan server activity [**] 141.156.176.107:2417 -> MY.NET.6.15:27374
10/01-21:59:20.742593 [**] Possible trojan server activity [**] MY.NET.6.15:27374 -> 141.156.176.107:2417
10/02-01:27:46.118585 [**] Possible trojan server activity [**] 24.60.194.13:4372 -> MY.NET.6.15:27374
10/02-01:27:46.118842 [**] Possible trojan server activity [**] MY.NET.6.15:27374 -> 24.60.194.13:4372
10/02-02:02:47.198725 [**] Possible trojan server activity [**] 68.81.83.92:3685 -> MY.NET.6.15:27374
10/02-02:02:47.198760 [**] Possible trojan server activity [**] MY.NET.6.15:27374 -> 68.81.83.92:3685
10/02-02:27:21 212.38.180.170:1818 -> 130.85.6.15:80 SYN ******S*
10/02-02:40:26 147.83.107.173:4882 -> 130.85.6.15:13240 SYN ******S*
10/02-02:40:27 147.83.107.173:4882 -> 130.85.6.15:13240 SYN ******S*
10/02-03:33:40 81.17.102.4:2368 -> 130.85.6.15:4898 SYN ******S*
10/02-07:04:05.306226 [**] External RPC call [**] 63.203.91.212:56637 -> MY.NET.6.15:111
10/02-07:04:05.389307 [**] External RPC call [**] 63.203.91.212:56637 -> MY.NET.6.15:111
10/02-07:04:05.553865 [**] External RPC call [**] 63.203.91.212:964 -> MY.NET.6.15:111
10/02-07:04:05.556598 [**] External RPC call [**] 63.203.91.212:964 -> MY.NET.6.15:111
10/02-07:04:05.638959 [**] External RPC call [**] 63.203.91.212:964 -> MY.NET.6.15:111
10/02-07:04:05.641095 [**] External RPC call [**] 63.203.91.212:964 -> MY.NET.6.15:111
10/02-07:04:05.661318 [**] External RPC call [**] 63.203.91.212:966 -> MY.NET.6.15:111
10/02-07:04:05.828587 [**] External RPC call [**] 63.203.91.212:966 -> MY.NET.6.15:111
10/02-07:04:05.830577 [**] External RPC call [**] 63.203.91.212:966 -> MY.NET.6.15:111
10/02-07:04:05.832768 [**] External RPC call [**] 63.203.91.212:56637 -> MY.NET.6.15:111
10/02-07:04:05.912352 [**] External RPC call [**] 63.203.91.212:966 -> MY.NET.6.15:111
10/02-10:55:07 68.159.36.251:53740 -> 130.85.6.15:6112 SYN ******S*
10/02-12:29:19 133.81.136.82:2766 -> 130.85.6.15:22 SYN ******S*
10/02-13:07:54 213.184.162.18:1823 -> 130.85.6.15:4899 SYN ******S*
10/02-13:33:41 218.38.24.83:1793 -> 130.85.6.15:554 SYN ******S*
10/02-22:39:17 65.198.127.106:137 -> 130.85.6.15:137 UDP
10/03-00:10:07 68.12.154.233:3994 -> 130.85.6.15:1080 SYN ******S*
10/03-00:25:04 68.88.212.132:3023 -> 130.85.6.15:3389 SYN ******S*
10/03-01:42:54.934832 [**] Possible trojan server activity [**] 172.151.131.58:3625 -> MY.NET.6.15:27374
10/03-01:42:54.935014 [**] Possible trojan server activity [**] MY.NET.6.15:27374 -> 172.151.131.58:3625
10/03-07:01:41.489349 [**] External RPC call [**] 203.253.204.51:111 -> MY.NET.6.15:111
10/03-07:01:44.725225 [**] External RPC call [**] 203.253.204.51:3442 -> MY.NET.6.15:111
10/03-07:01:44.961183 [**] External RPC call [**] 203.253.204.51:996 -> MY.NET.6.15:111
10/03-07:01:44.979090 [**] External RPC call [**] 203.253.204.51:3442 -> MY.NET.6.15:111
10/03-13:37:11 159.242.13.10:37433 -> 130.85.6.15:80 SYN ******S*
10/03-15:01:08 217.227.109.91:2189 -> 130.85.6.15:80 SYN ******S*
10/03-19:16:37 24.241.96.134:1837 -> 130.85.6.15:17300 SYN ******S*
10/03-23:40:21 194.3.174.112:3162 -> 130.85.6.15:554 SYN ******S*
10/04-10:53:50 203.253.204.51:1524 -> 130.85.6.15:1524 SYN ******S*
10/04-12:55:54 65.213.110.86:1763 -> 130.85.6.15:80 SYN ******S*
10/04-18:36:19 212.202.30.102:1317 -> 130.85.6.15:80 SYN ******S*
10/05-00:42:34 66.171.157.127:3612 -> 130.85.6.15:4000 SYN ******S*
10/05-00:42:35 66.171.157.127:3612 -> 130.85.6.15:4000 SYN ******S*
10/05-01:14:12 130.13.66.175:1853 -> 130.85.6.15:17300 SYN ******S*
10/05-03:11:47 24.161.109.219:4860 -> 130.85.6.15:17300 SYN ******S*
10/05-03:11:48 24.161.109.219:4860 -> 130.85.6.15:17300 SYN ******S*
10/05-03:15:51 24.174.178.167:3765 -> 130.85.6.15:17300 SYN ******S*
10/05-04:02:01 217.34.34.142:3783 -> 130.85.6.15:17300 SYN ******S*
10/05-07:22:51 63.122.16.9:2086 -> 130.85.6.15:80 SYN ******S*
10/05-10:02:37 195.136.250.51:1483 -> 130.85.6.15:554 SYN ******S*
10/05-11:26:03 24.103.56.151:4788 -> 130.85.6.15:7070 SYN ******S*

Event Details
10/05-11:26:04 24.103.56.151:4788 -> 130.85.6.15:7070 SYN ******S*
10/05-11:30:32.623008 [**] External RPC call [**] 24.207.141.186:798 -> MY.NET.6.15:111
10/05-11:30:32.799925 [**] External RPC call [**] 24.207.141.186:798 -> MY.NET.6.15:111
10/05-11:30:32.803970 [**] External RPC call [**] 24.207.141.186:798 -> MY.NET.6.15:111
10/05-13:15:57 194.199.203.7:4353 -> 130.85.6.15:80 SYN ******S*
10/05-16:23:31 12.226.188.27:3832 -> 130.85.6.15:17300 SYN ******S*
10/05-16:23:54.512263 [**] Possible trojan server activity [**] 24.62.71.91:2540 -> MY.NET.6.15:27374
10/05-16:23:54.512370 [**] Possible trojan server activity [**] MY.NET.6.15:27374 -> 24.62.71.91:2540
10/05-17:25:27 68.106.40.188:1149 -> 130.85.6.15:17300 SYN ******S*
10/05-18:57:47 66.24.131.54:4447 -> 130.85.6.15:17300 SYN ******S*
10/05-22:29:48 192.160.131.12:55868 -> 130.85.6.15:17300 SYN ******S*
10/05-22:36:53 80.247.76.117:2599 -> 130.85.6.15:34816 SYN ******S*
10/05-23:23:16 64.30.211.151:1472 -> 130.85.6.15:17300 SYN ******S*

Table 33. Events triggered by traffic to\from host MY.NET.6.15 (130.85.6.15)

Figure 10 shows a link graph of activity associated with targeting of host MY.NET.6.15. The stimulus traffic to MY.NET.6.15 is shown in black, while the response traffic if any is shown in red. This graph clearly shows that except for the traffic to port tcp/27374, the remaining traffic to MY.NET.6.15 is uni-directional, i.e. MY.NET.6.15 does not respond to the stimulus. The link graph also shows more information about the activities of three of the hosts that targeted MY.NET.6.15. This is shown to illustrate that many of the hosts that targeted MY.NET.6.15 also targeted other hosts on the MY.NET.6.0 network.

Table 34 shows information about the five hosts that targeted port tcp/27374 on MY.NET.6.15 and one of the hosts that targeted port tcp/111. The events count column shows the total number of events generated by traffic to/from these hosts over the five-day period covered in this report.

Source IP	Resolved Name	Event Count	Remarks
203.253.204.51	physics.cheju.ac.kr	9603	• Owner: Cheju National University • Contact: sbyoon@cheju.ac.kr • not in DShield database
63.203.91.212	adsl-63-203-91-212.dsl.snfc21.pacbell.net	292	• Owner: Pac Bell Internet Services • Contact: abuse@pacbell.net • not in DShield database
24.62.71.91	h008019441d6b.ne.client2.attbi.com	203	• Owner: Comcast Cable Communications Holdings, Inc • Contact: abuse@comcast.net • not in DShield database
24.60.194.13	h001095d776d4.ne.client2.attbi.com	96	• as above
172.151.131.58	AC97833A.ipt.aol.com	51	• Owner: America Online • Contact: abuse@aol.net • not in DShield database
141.156.176.107	pool-141-156-176-107.esr.east.verizon.net	6	• Owner: Verizon Internet Services • Contact: abuse@verizon.net • not in DShield database

Table 34. Information about some hosts that targeted tcp/27374 and tcp/111 on MY.NET.6.15

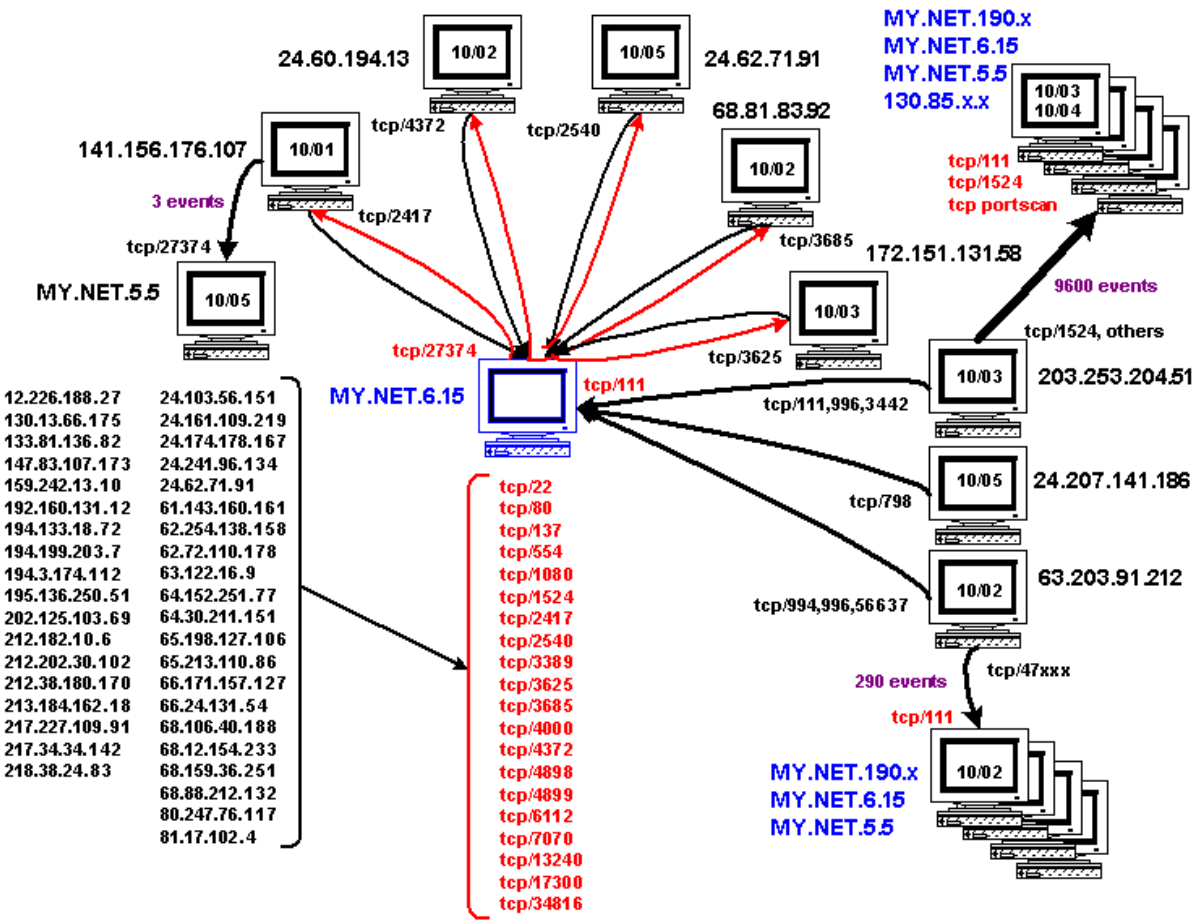

Figure 10. Link graph of activity associated with targeting of host MY.NET.6.15

3.9 Insights into internal machines

This security audit revealed a number of signs of compromised systems as well as possible dangerous or anomalous activity. These signs are consolidated into this section, although there is additional information available in the sections dealing with the analysis of the important detects.

1. The potentially infected hosts identified in Table 9 need to be examined.

2. MY.NET.162.118 and MY.NET.150.133 accounted for 90% of the “non-ICMP (non-spp_portscan)” alerts (see Section 3.4). The cause of this traffic needs to be eliminated.

3. The use of the internal hosts listed in Table 15, e.g. MY.NET.162.118, that are responsible for generating an excessive number of alerts need to be examined.

4. The internal hosts listed in Table 15, e.g. MY.NET.30.4, that are the target in an excessive number of alerts need to have their patch levels reviewed and examined for signs of compromise.

5. The hosts listed in Table 17 should be scanned for worms and both outbound and inbound traffic to the Microsoft ports of 137-139 and 445 both udp and tcp should be blocked.

6. If host MY.NET.30.4 is special then the access to this server from the Internet be reviewed and tightened up rather then leaving it so open.

7. If host MY.NET.30.3 is special then the access to this host from the Internet be reviewed and tightened up.

8. Some of the internal hosts may be infected with the Adore worm so the following steps are recommended:

    a. The internal hosts that need to be checked are those indicated in Table 22 and Table 23 by a "Y" in the "Check?" column if they are running Linux.

    b. Patch all Linux hosts against the Adore worm.

    c. Block access to any of the destination hosts in Table 22 and Table 23 if investigation shows that they are associated with the Adore worm.

9. The patch level and accessibility of the internal host MY.NET.66.2 are to be examined since it was heavily target by the "Null scan!".

10. For all publicly exposed hosts:

    a. Ensure that the IP stacks are not vulnerable to DoS due to the handling of tiny fragments.

    b. Review the patch levels of all publicly accessible hosts for vulnerabilities such as lprd overflows.

11. The host MY.NET.6.15 needs to be checked for a Trojan server or other cause of the traffic to and from port tcp/27374. If the system is running a legitimate program on port tcp/27374, then either the port number of the application should be changed or else the "Possible Trojan server activity" signature tuned to avoid generating false-positives.

3.10 Defensive recommendations

Defensive recommendations should be based on both the results of this audit and the organization's security policy. Currently it appears that this organization is operating in a manner that allows much more types of traffic both from the inside to outside and from the outside to inside than is prudent. It is reasonable to conclude that the current security policy, not available for this Practical Assignment, results in higher than

necessary operational costs. These costs would be incurred for both log analysis, investigation of potential intrusion events and excessive hardening requirement.

The study of detects over the study period showed that a handful of student systems account for an inordinate number of the events reported by the IDS sensor. Action is required to ensure that these systems stop generating such traffic. To this end, the university's Acceptable User Policy (UAP) needs to be strengthened so that malicious traffic is clearly defined and systems generating such traffic can be removed from the network until it stops.

The major recommendation for management is that this organization needs to adopt the following two pillars for sound security:

1. A security policy based on an "only allow what is explicitly permitted and deny everything else" approach.

2. A defense in depth approach to reduce the chance of an intrusion.

The first pillar is based on the belief that it is easier and more secure to define what types of traffic, both from the inside to outside and from the outside to inside, are required than it is to define a definitive list of bad traffic. Defining a list of bad traffic to block is very labour intensive and it's a losing proposition since hackers are always investigating new ways to exploit any exposed service. With the adoption of this approach, the IDS tuning could be significantly improved to increase the sensor's effectiveness by reducing the number of false-positives.

The second pillar is a defense in depth approach to reduce the chance of an intrusion. It's a classical approach to security since it presents the intruder with a number of security devices to get through and worry about their intrusion reporting capability. As shown in Figure 11, on both the route to the inside network or the DMZ, an intruder is subjected to the traffic management and traffic reporting capabilities of three or four security devices, i.e. the two routers, firewall and IDS sensor. As well the publicly exposed hosts are protected by host-based IDS (HIDS). Tying all these security devices together is the central Security Information Management (SIM) system. The SIM acts like a syslog server but has enhanced reporting and alert generation capabilities.

Of course there is a capital cost required to implement this recommended approach. However with a business case, it should be possible to show that the operational cost savings and the value of the foregone security breaches would offset capital cost of this approach.

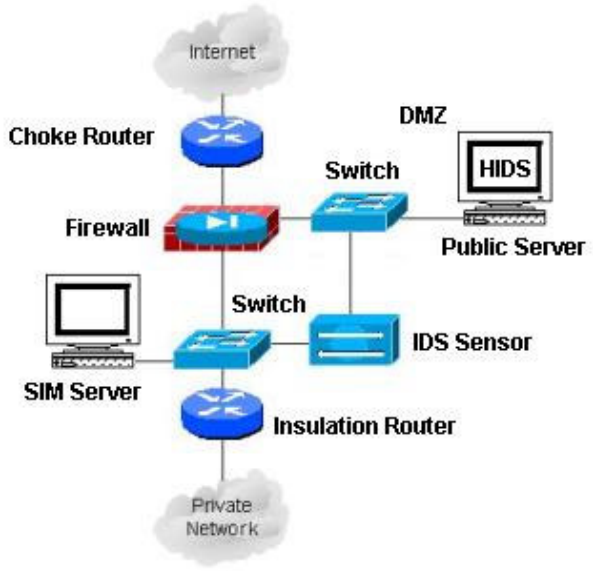

Figure 11. Recommended Defence in Depth Approach

On the practical side, there are a number of Snort modifications recommended in this audit that need to be actioned so as to make the sensor output more useful and thereby enhancing the university's security posture. Some of these recommendations are listed below (the reader is referred to sections above for more recommendations and further explanations):

1. The scans contain a number of alerts with "SYN 12****S* RESERVEDBITS" which is due to a bug that required modifications to the TOS plugin according to Martin Roesch. The TOS plugin should be updated.

2. The purpose of the "ICMP (non-spp_portscan)" alerts needs to be reexamined since all of these alerts involved ICMP traffic to and from hosts outside the internal network.

3. The SMB Name Wildcard Signature (902,224 hits) had no detects caused by traffic from an external host to an internal host. The purpose of this signature needs to be revisited.

4. The Incomplete Packet Fragments Discarded signature is buggy since according to Martin Roesch the type of alert that is being seen is because the sensor is using the defrag preprocessor instead of the newer frag2 preprocessor.

3.11 Description of Analysis Process Used

The events covering the period of October 1-5, 2003 were found in the files listed in Table 8. The standard Unix text manipulation commands, i.e. cut, awk, sed and grep were used to extract the data presented in this Practical Assignment.

As a first step the top talkers, top listeners, top signatures, top source ports and top destination ports were extracted. Based on this information, more specific information was extracted from the log files.

Given the size of the files, e.g. the scans, and the shortage to disk space available, the files were mainly left in their zipped format. The following is a sample command used to operate on the scans file when looking for a specific IP address:

```
for fname in scans.03100*.gz
do
gunzip -cd $fname | grep "24\.62\.71\.91" >> 24.62.71.91
done
```

then extracting the destination IP addresses targeted by the specific IP address:

```
cat 24.62.71.91 | cut -d' ' -f8 | cut -d':' -f1 > temp
awk -f counter.awk < temp | sort -rn > 24.62.71.91-ip_out
```

This analysis process was adequate for the purposes of this Practical Assignment but for ongoing operational analysis a much more highly automated process would be required. However this automated process is beyond the scope of the Practical Assignment.

References

6. SANS - What port numbers do well-known trojan horses use?, http://www.sans.org/resources/idfaq/oddports.php, updated 2/9/01
7. arachNIDS - The Intrusion Event Database - IDS203 "TROJAN-ACTIVE-Q-TCP", http://whitehats.com/cgi/arachNIDS/Show?_id=ids203&view=research, updated 2/9/01
8. Novell - Technical Information Document “NetWare 6.0 Support Pack 2 - TID2963227", http://support.novell.com/servlet/tidfinder/2963227, last modified 11SEP2002
9. Symantec - W32.Nimda.A@mm, http://securityresponse.symantec.com/avcenter/venc/data/w32.nimda.a@mm.html, Last Updated on: July 10, 2003 07:46:47 PM
10. LURHQ Threat Intelligence Group - Intrusion Detection: In-Depth Analysis, http://www.lurhq.com/idsindepth.html, undated
11. Re: [Snort-users] Incomplete Packet Fragments Discarded, http://www.mcabee.org/lists/snort-users/Nov-01/msg00820.html, 26 Nov 2001
12. Internet Storm Center - Port 17300, http://isc.incidents.org/port_details.html?port=17300, undated
13. ICSA Labs Firewall Lab Report For Novell BorderManager, http://www.icsalabs.com/html/communities/firewalls/certification/rxvendors/novellbordermanager36/labreport_cid1466.shtml, Last updated: September 9, 2002
14. Internet Storm Center - Port 27374, http://isc.incidents.org/port_details.html?port=27374, undated
15. Project Honeynet, Scan of the month (SCAN 20), http://project.honeynet.org/scans/scan20/sol/24.txt, April 2002
16. eEye Digital Security, Windows 2000 IIS 5.0 Remote Buffer Overflow Vulnerability (Remote SYSTEM Level Access), http://www.eeye.com/html/Research/Advisories/AD20010501.html, May 01, 2001
17. arachNIDS - The Intrusion Event Database, http://www.digitaltrust.it/arachnids/IDS4/event.html, undated
18. Project Honeynet, Scan of the month (SCAN 23), http://project.honeynet.org/scans/scan23/sol/Neil.html, undated
19. SecurityFocus HOME Vulns Info: Multiple Vendor lpr Format String Vulnerability, http://www.securityfocus.com/bid/1711, Sep 26, 2000
20. Snort Signature Database: SID 522 (MISC Tiny Fragments), http://www.snort.org/snort-db/sid.html?id=522, undated
21. Re: Newbie and snort results, http://www.incidents.org/archives/intrusions/msg10000.html, Mon, 6 May 2002
22. SANS Institute: Adore Worm, http://www.sans.org/y2k/adore.htm, April 12, 2001
23. F-Secure Virus Descriptions : Adore, http://www.europe.f-secure.com/v-descs/adore.shtml, April 2001
24. Neohapsis Archives - Snort discussion - Re: [Snort-users] Is there a problem with Linux 2.4.0? - From roesch, http://archives.neohapsis.com/archives/snort/2001-01/0198.html, Jan 12 2001

25. DShield.org - Port Report, http://www.dshield.org/port_report.php?, last update: 21/Feb/2004
26. IEEE OUI and Company_id Assignments, http://standards.ieee.org/regauth/oui/index.shtml, Modified: 30 January 2004
27. Snort Signature Database - SCAN Proxy \(8080\) attempt, http://www.snort.org/snort-db/sid.html?sid=620, undated
28. Snort Signature Database - SCAN Squid Proxy attempt, http://www.snort.org/snort-db/sid.html?sid=618, undated
29. Snort Signature Database - SNMP public access udp, http://www.snort.org/snort-db/sid.html?sid=1411
30. RFC 1514 (Host Resources MIB), http://rfc-1514.rfclist.net/rfc-1514.htm, September 1993
31. CERT Advisory CA-2002-03 Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP), http://www.cert.org/advisories/CA-2002-03.html, Last revised: Aug 18, 2003
32. Bugtraq Archive - Cert Advisory 2002-03 and HP JetDirect, http://securityfocus.com, Posted Tue, 19 Feb 2002 10:53:48
33. [Snort-users] HP Printers - SNMP Public Access udp, http://www.mcabee.org/lists/snort-users/Nov-03/msg00402.html, Posted Tue, 18 Nov 2003 12:29:43
34. GIAC Certified Intrusion Analyst (GCIA) Practical Assignment Version 3.3 by Peter H. Storm, http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf, November 15 2003
35. Snort Signature Database - BAD-TRAFFIC ip reserved bit set, http://www.snort.org/snort-db/sid.html?sid=523
36. Internet Protocol DARPA Internet Program Protocol Specification, http://www.ietf.org/rfc/rfc0791.txt, September 1981
37. Default TTL Values in TCP/IP, http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html, undated
38. GIAC Certified Intrusion Analyst (GCIA) Practical Assignment Version 3.3 by James Maher, http://www.giac.org/practical/GCIA/James_Maher_GCIA.pdf, dated 16/07/2003
39. Common Vulnerabilities and Exposures (CVE) database, http://cve.mitre.org/cve/, undated
40. RFC 793 - Transmission Control Protocol, http://www.faqs.org/rfcs/rfc793.html, September 1981
41. RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP, http://www.faqs.org/rfcs/rfc3168.html, September 2001
42. Snort - Chapter 2 Writing Snort Rules How to Write Snort Rules and Keep Your Sanity, http://www.snort.org/docs/writing_rules/chap2.html, undated
43. [Snort-users] TOS plugin modified (ECN mitigation) from Martin Roesch, http://archives.neohapsis.com/archives/snort/2001-01/0200.html, Fri Jan 12 2001 - 14:43:36 CST
44. Snort Signature Database - SCAN XMAS, http://www.snort.org/snort-db/sid.html?sid=625
45. HPING MAN PAGE, http://www.hping.org/manpage.html

46. North American Network Operators Group - Multiple vendors' TCP/IP implementation allow packets to bypass firewalls - VU#464113, http://www.merit.edu/mail.archives/nanog/2002-10/msg00519.html, Wed Oct 23 20:50:17 2002
47. SecurityFocus - Microsoft FrontPage Server Extensions MS-DOS Device Name DoS Vulnerability, http://www.securityfocus.com/bid/1608/discussion/, undated
48. insecure.org - Many, many, many security holes in the Microsoft Frontpage extensions, http://www.insecure.org/sploits/Microsoft.frontpage.insecurities.html, Thu, 23 Apr 1998 14:36:00
49. Snort Signature Database - WEB-IIS _vti_inf access, http://www.snort.org/snort-db/sid.html?sid=990
50. Snort Signature Database - WEB-MISC WebDAV propfind access, http://www.snort.org/snort-db/sid.html?sid=1079