Global Information Assurance Certification Paper

Intrusion Detection and Analysis:

An Investigation

GCIA Certification
Practical Assignment
v3.3

By Terry MacDonald

(Submitted 21 June 2003)

(page intentionally left blank)

Change Control and Typographical Conventions

Document

Title:	GIAC GCIA Certification Assignment v3.3
Version:	1.0
Date:	21 June 2003
Author:	Terry MacDonald
Printed:	04 August 2003

Distribution Restrictions

Name	Role(s)	Organisation(s)
None	N/A	N/A

Change Record

Version	Date	Comments	Updated By
0.1	27 April 2003	Initial Draft	Terry MacDonald
0.9	9 June 2003	Second Draft	Terry MacDonald
1.0	21 June 2003	Initial Release	Terry MacDonald

Typographical Conventions

Examples of Typographical Meaning
Normal Text
`Output of Commandline Program`
Quoted Text from a Webpage
Normal Text in a Table
Text in a Table (for a lot of data)

(page intentionally left blank)

Table of Contents

Paper Abstract

This paper was created in accordance with the guidelines given by the GIAC to fulfil the GCIA practical assignment criteria. The paper is divided into three sections.

In the first section, I discuss whether it is within our rights to retaliate against those who are attacking our networks. I show both sides of the argument and give my opinion on each point of view. Note this section has it own separate list of references.

The second section discusses three network detects. One network detect was sourced from a client network, and two were sourced from logs posted on http://www.incidents.org/logs/Raw. I discuss the logic and methodology used in arriving at my decisions, and explain my conclusions.

The third section is an analysis of IDS data that was given to me by a U.S. University. This University wanted a report produced which analysed the data, prioritised the incidents reported, and recommended any improvements to the Universities security configuration.

All appendices and references not included in assignment one are listed at the rear of the document along with all logs and information that is important to have, but which may have affected the readability of the paper itself.

Assignment 1. Describe the State of Intrusion Detection

1.1. The best form of defence is offence - but is it really ok to hack back?

1.1.1. Introduction

You are the Intrusion Analyst for a medium sized company that does a lot of business over the Internet - and you are stressed. For the past two weeks, your email systems have been inundated with viruses, all seeming to have originated from one single computer. You have tried emailing and phoning the owner, the attackers ISP and even the Cybercrime unit of your local police force but still no response. There seems like only one option open to you - to hack back.

But should you? What penalties might you incur if you or your organisation do hack back?

1.1.2. What is a Hack Back?

First, lets discuss terminology. When an Intrusion Analyst (IA) notices that their organisation is under attack, they attempt to classify the attack, and react to it. There tends to be three general types of reaction that IA's take.

Firstly, they can initiate a Passive Response. This is action taken by the IA or the organisation's ISP that affects only their local network. This type of action is the most common response. It is often used to combat portscans and other lower level exploit attempts. Things that come under this category are blocking the dangerous traffic from that ISP, or getting your mail server to ignore incoming messages from that email sender. This is the response that most IA's give for general alerts.

Secondly, they can initiate an Active Response. This is where the IA collects packet traces, and other information about the attacker for evidence, then contacts the relevant people to get the attacker stopped or apprehended. The relevant people could be the users ISP, the local authorities in the attackers country or your local authority if criminal investigations need to be made. Information like the attackers IP address, the types of packet sent, packet traces, connection times, the types of information sought. Honeypots can be a great tool for this type of deception, as they allow a great deal of information to be gathered about the way an attack was made. This is the level given for attacks that are more persistent and for exploits that are successful.

The third type of response the IA can initiate is the Invasive Response. This is where the IA performs some sort of reverse attack on the attackers machine. The IA of the initial target network actively takes matters into its own hands. It includes everything from performing a Denial of Service attack on the attackers machine, to sending and email bomb, to gaining full control of the remote computer. This is a hack back.

1.1.3. The Problem

Many people have been the victims of hacking. Malicious traffic is so prevalent on the Internet that computers have been exploited within 15 minutes of connecting to it [1]. Systems are scanned an average of 17 times a day. There are so many automated scanners trawling through the Internet looking for vulnerable machines that it is very likely you will be caught out. Some attackers install zombie programs to lie in sleep before they are woken by the attacker to be used for nefarious purposes [2]. Others utilise open proxy servers [3] to hide their real IP addresses to do their dirty work. Other release viruses, worms and trojans out into the Internet. They do this for various reasons - to increase their reputations, for political reasons, for criminal purposes and just for fun.

How do we combat this? Standard practises include installing a firewall, using boundary router to drop packets, and monitoring using NIDS systems. However, what if you are attacked persistently by a particular computer and you get no response to complaints to the attackers ISP? Wouldn't it be good if you could give the attacker a taste of their own medicine? If all you need to do was a little invasive response to disable the attack and stop it completely at it's source….

1.1.4. The Argument For

For some people enough is enough. They have spent too much time fixing problems caused by attacks and just want to cut down the number of problems that they have to deal with. What better way than to stop the problem at it's source. Why repetitively defend against external attacks, when you can attack them once and be done with it? It is something many analysts have thought of at one time or another. Even the U.S. military is attempting to get permission to strike back for attacks on its networks [4].

Most current discussion about hack backs seems to be about the unauthorised repairing of infected innocent computers. Sometimes a user's computer is being controlled by a third party and is unknowingly being involved in Denial of Service attacks [2] and other illegal tasks. The attacker would have infected the users machine through exploiting a known vulnerability, and would have installed a 'bot'* and/or a backdoor to gain access to the machine in the future [2]. There have been cases where some attackers have infected so many computers that they control networks of bots that number more than 10,000 computers [5]. This could be avoided if the user had patched their software to fix the vulnerabilities, or if they had been running firewall or anti-virus software, but most consumers are unaware the vulnerabilities exist, let alone the fixes. If a user's computer is infected with a bot, and they are unaware that they are attacking your network, then why not take matters into your own hands and fix their machines for them?

Jonathan Morton [6] decided to fight back against the Fizzer Virus. He created a Java IRC bot called FizzerKiller that attacks the Fizzer IRC bot. The Fizzer virus/trojan/bot installs itself through either email, KaZaA or IRC on to another users PC. FizzerKiller gains control of the bot, then uninstalls it. This gets rid of the

* A bot is a remotely controlled program that is installed by a malicious remote user.

problem. The user of the infected machine is unaware that anything has happened; yet the Internet at large is spared from clogged bandwidth.

Another company, Bindview, have released a tool called Zombie Zapper [7] that connects to the machines that are taking part in a DDoS against you and sends them a control packet containing commands that tell them to stop the attack. It halts attacks from the TFN, Stacheldraht, Trinoo and Shaft bots.

FutureVision, RSA, and other companies are also joining the bandwagon [8]. They all are producing tools that combat Denial of Service attacks.

Suppose an attacker is involved in a series of focused DoS attacks that are specifically aimed at one or two of your servers. These have been going on for a few days. You and your ISP have tried to mitigate these attacks but have been unable to. Would you take offensive action?

That is exactly what Conxion did when a group of hacktivists attempted to take down the WTO website that Conxion were hosting [8]. It was January 2000 during the WTO summit in Seattle. The hacktivists were from an online group called the Electrohippies. They launched a DoS against the WTO website. Instead of filtering out the attack, as is currently the norm, Conxion returned the mail bomb packets to the originating server, taking it offline for several hours, and stopping the original attack.

1.1.5. The Argument Against

The main problem with hack backs is the chance of hitting innocent bystanders. How do you know that the person who has the computer is really controlling the attack? The users computer could be infected with a bot, and could be under the control of a third party, with the user unknowingly being involved in a denial of service attack [2]. Traffic could appear to have been initiated by the user, when in fact it is the third-party attacker who is responsible.

Traceability is another major issue. The TCP/IP suite of protocols were developed for ease of connectivity, not security [9]. Consequently, it is easy for an attacker to spoof their IP address. You cannot be sure that a packet really did originate where it says it did, as there is no built in tamper-proof way of recording where a packet has been.

This lack of traceability has spawned a lot of research to try to overcome this [8]. Deterministic [10] and probabilistic [11] packet marking, router stamping [12], and sleepy watermark tracing [13] are all techniques being developed to help with traceability. If people gain the ability to trace messages, then they gain the ability to identify the sources of incoming threats. It would mean that the ability of attackers to hide behind a veil of open web proxies and infected hosts would be removed, making it easier for law enforcement agencies to catch those responsible, removing the need for cyber vigilantes and invasive response.

1.1.6. The Law

So what does the law say? The law is quite clear-cut about this. In the US, the Computer Fraud and Abuse Act of 1986 [14], the Unlawful Access to Stored Communications Act [15], and Electronic Communications Privacy Act of 1986 [16] all make no distinction between the original attacker and any retaliation attacks. This means that if you retaliated against an initial attack, then you would be accessing the attackers computer unauthorised with a malicious intent, and you could be prosecuted under the same laws as the attacker. This includes any attempt to remove viruses or to patch virus infected machines remotely. It is worth noting that the three U.S. Acts that pertain to computer misuse and fraud [14][15][16] have been amended with the enactment of the U.S. Patriot Act of 2001 [17]. It is now illegal to hack into a foreign computer, any damage is now considered enough to break the law, and victims of hacking can request law enforcement officers to monitor the communications of unauthorised users.

In the UK, the law is not so obvious. Some aspects of hacking are covered by the Computer Misuse Act 1990 [18]. The installation of Trojans/Viruses and backdoor are covered under the Act, but the actual use of these trojans in DDoS attacks against other machines does not seem to be covered [19][20]. The government is currently reviewing the Computer Misuse Act to rectify the DoS loophole [21], but no more has been heard from the UK government since this was made public in May 2002.

An example of the difficulty the legislation faces is demonstrated by another attempt to control the Fizzer virus, which was made by the Fizzer Task Force [22]. A particular analyst noticed that the virus was going to a particular website to download update code for itself. He and a friend developed some code that uninstalled the virus. They uploaded removal code to the update page in the hope it would trigger the removal of the virus. In essence, the virus would kill itself. In fact, the code did not work correctly and the virus did not uninstall itself.

Was this attempted virus kill against the law? The analyst did write code that was downloaded without the users knowledge and that would have changed things on the users computer. Yet, it was the users infected computer that initiated the connection and downloaded the code from the site. There was no 'invasive response' on the analyst's part. He did not instruct the machines to get the code and use it, but the virus did. Was what analyst attempt to do truly illegal?

1.1.7. Conclusion

There is no real place for cyber vigilantism. As the law says, anyone entering a computer unauthorised is breaking the law, and makes him or herself liable for prosecution. At present, the current law fails to take all of the different ways that damage can be done to a computer into account. It also fails to allow for the challenges that the anonymity of the Internet permits. But it is still the law.

Hacking back is illegal, and attempts to do so should not be made.

We should instead redirect our efforts to lobbying government to improve legislation, and help guide the government departments into improving the effectiveness of

controls. If the current structure of computer crime prevention is not working then we need to improve it - not take matters into our own hands.

Going back to our opening two-week attack scenario, you should react using passive and active responses – but not an invasive response. You should use all options available to you but you should not do anything illegal. If you are unsure if you actions are illegal then stay on the safe side and do not do it. Remember that you or your organisation could be held accountable for any consequences of your actions. Ultimately it is the law that sets the ethical standards that govern our actions, and that society judges us by, and as such, it is the law that should be followed.

1.2. References

[1] WebCentric. "The National Strategy to Secure Cyberspace". Webcentric Computer Services Website. URL: http://www.web-centric.net/files/cybersecurity2.pdf (9 June 2003)
[2] Anonymous. "All about Bots, Trojans and Worms". Swat -It Website. 2003. URL: http://swatit.org/bots/index.html (9 June 2003)
[3] Anonymous. "Openproxies FAQ". Openproxies.com Website. URL: http://www.openproxies.com/about.php (9 June 2003)
[4] Heuston, George. "NIPC Daily Report 13 November 2001". Email posting on NIPC Watch mailing list. 13 November 2001. URL: http://lists.jammed.com/crime/2001/11/0114.html (9 June 2003)
[5] Roberts, Paul. "Al-Jazeera hobbled by DDOS attack". ComputerWorld Website. 26 March 2003. URL: http://www.computerworld.com/securitytopics/security/story/0,10801,79743,00.html (9 June 2003)
[6] Morton, Johnathan. "FizzerKiller Readme". Chromatix Website. 31 May 2003. URL: http://www.chromatix.org.uk/antifizzer/README.fizzerkiller (9 June 2003)
[7] Simple Nomad. "Zombie Zapper Overview and Usage Notes". Razor Bindview Web site. 29 March 2000. URL: http://razor.bindview.com/tools/desc/ZombieZapper_readme.html (9 June 2003)
[8] Jayawal, Vikas; Yurcik, William; Doss, David. "Internet Hack Back: Counter Defense as Self-Defense or Vigilantism". SOS Research Website. June 2002. URL: http://www.sosresearch.org/publications/ISTAS02hackback.PDF (6 June 2003)
[9] Dimitrov, Ivan. "Network Security: Limitations Of The Internet Protocol And The IPSec Protocol Suite". McMaster University Website. http://www.cas.mcmaster.ca/~wmfarmer/SE-4C03-01/papers/Dimitrov-IPSec.html (9 June 2003)
[10] Belenky, Andrey; Ansari, Nirwan. "IP Traceback with Deterministic Packet Marking". New Jersey Institute of Technology. April 2003. URL: http://web.njit.edu/~ang/papers/COMM_Let03.pdf (9 June 2003)
[11] Park, Kihong; Lee, Heejo. "On the effectiveness of Probabilistic Packet Marking for IP Traceback under Denial Of Service Attack". SecurityFocus Website. April 2001. URL: http://www.silicondefense.com/research/itrex/archive/tracing-papers/park01effectiveness_of_marking.pdf (9 June 2003)
[12] Dean, Drew; Franklin, Matt; Stubblefield, Ada m. "An Algebraic Approach to IP Traceback". Silicon Defense Website. 6 April 2001. URL: http://www.silicondefense.com/research/itrex/archive/tracing-papers/dean01algebraic_approach.pdf (9 June 2003)
[13] Wang, Xinyuan. "Survivability through Active Intrusion Response". Cert.org Website. 2000. URL: http://www.cert.org/research/isw/isw2000/papers/9.pdf (9 June 2003)
[14] Anonymous. "Computer Fraud and Abuse Act of 1986". Cornell Law Website. URL: http://www4.law.cornell.edu/uscode/18/1030.html (9 June 2003)
[15] Anonymous. "Unlawful Access to Stored Communications Act". Cornell Law Website. URL: http://www4.law.cornell.edu/uscode/18/2701.html (9 June 2003)
[16] Anonymous. "Electronic Communication s Privacy Act of 1986". Cornell Law Website. URL: http://www4.law.cornell.edu/uscode/18/1367.html (9 June 2003)
[17] Anonymous. "Patriot Act of 2001". Cybercrime Website. URL: http://www.cybercrime.gov/PatriotAct.htm (9 June 2003)
[18] Anonymous. "Computer Misuse Act 1990". Her Majesty's Stationery Office Website. 20 September 2000. URL: http://www.hmso.gov.uk/acts/acts1990/Ukpga_19900018_en_1.htm (9 June 2003)
[19] JISC. "New Developments in UK Internet Law". APU Website. URL: http://www.apu.ac.uk/guidelines/smbp10.pdf (9 June 2003)
[20] Goodwin, Bill. "Call for cyber law review". Computer Weekly Website. 19 December 2002. URL: http://www.computerweekly.com/articles/article.asp?liArticleID=118351&liFlavourID=1 (9 June 2003)
[21] Goodwin, Bill. "Whitehall begins review of UK's cybercrime laws". Computer Weekly Website. 22 May 2002. URL: http://www.cw360.com/Article112682.htm (9 June 2003)
[22] Stewart, Joe. "Re: Fizzer self-destruct". Email posting on the icidents@intrusions.org mailing list. 16 May 2003. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/05/msg00142.html (9 June 2003)

1.3. Further Reading

[23] Radcliff, Deborah. "Can you hack back?". CNN Website. 1 June 2000. URL: http://www.cnn.com/2000/TECH/computing/06/01/hack.back.idg/#r (8 June 2003)
[24] Kabay, M. E. "Don't hack back". Network World Fusion Website. 6 May 2003. URL: http://www.nwfusion.com/newsletters/sec/2003/0505sec1.html (8 June 2003)
[25] Lyman, Jay. "When the Hacked Becomes the Hacker". News Factor Website. 19 November 2001. URL: http://www.newsfactor.com/perl/story/14874.html (7 June 2003)
[26] Yurcik, William. "Information Warfare: Legal & Ethical Challenges of the next Global Battleground". SOS Research Website. June 1997. URL: http://www.sosresearch.org/publications/ethics97.PDF (8 June 2003)
[27] Smith, Bryan; Yurcik, William; Doss, David. "Ethical Hacking: The Security Justification". SOS Research Website. 18 October 2001. URL: http://www.sosresearch.org/publications/eei21.pdf (7 June 2003)
[28] Smith, Bryan; Yurcik, William; Doss, David. "Ethical Hacking: The Security Justification Redux". SOS Research Website. June 2002. URL: http://www.sosresearch.org/publications/ISTAS02ethicalhack.PDF (7 June 2003)
[29] Yurcik, William. "Information Warfare Survivability: Is the Best Defense a Good Offense?". SOS Research Website. July 2000. URL: http://www.sosresearch.org/publications/ethics00.PDF (8 June 2003)
[30] Loomis, Chirstopher. "Defenders or Digilantes: an overview of the appropriate response debate". Security Focus Website. 19 July 2001. URL: http://www.securityfocus.com/guest/6247 (8 June 2003)

Assignment 2. Network Detects

2.1. Detect 1

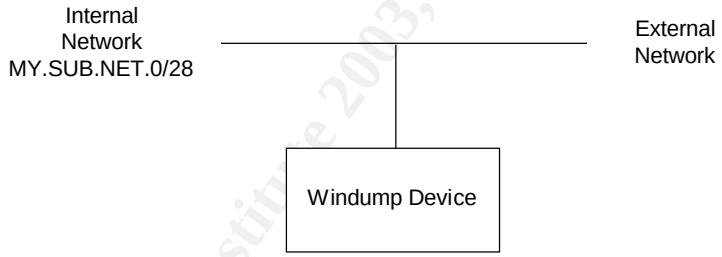
2.1.1. Source of Trace

The raw binary packet log used in for this detect was sourced from a client's network. The binary files were created on 22nd May 2003 with Windump using the commandline below:

```
windump -w 2002-05-22
```

This recorded all traffic into a raw binary file for use with snort. The original plan was to sample a full 24 hours of traffic. In the future this will be a service offered to our clients, but as this was a test situation, only a couple of hour's traffic was saved. Windump was selected to capture the binary packet data to provide the highest fidelity logs if further analysis was needed.

The windump instance sat on a small server outside the corporate network. It was connected to a mirrored port on the switch that led to the ISP's local router, and logically sat between the firewall and the ISP. This configuration was chosen so that the windump instance could see all of the traffic from and to the client network, but could not participate in any communication. To enhance security further the computer was not given an IP address, and all unnecessary networking protocols were removed.



2.1.2. Detect was generated by

To find the detect I used Snort Version 1.9.1-ODBC-MySQL-WIN32 (Build 231) [31] running the current snort rules [32] (as at 25 April 2003). The rules were slightly modified as I 'turned on' the rules that are normally off in the default installation (things like shellcode rules, etc.) to make the full range of possible detects available.

Similarly to the raw logs available from incident.org [33], the logs that I generated had the following features:

- All internal addresses have been changed to MY.SUB.NET.x
- Checksums have been changed as well to stop reverse engineering of IP Addresses
- External IP Addresses are unchanged from those captured.

The commandline I used to generate the logs is below:

```
snort -X -r c:\detect\2003-05-22 -l c:\detect\logs -c c:\snort
\rules\snort.conf -S HOME_NET=MY.SUB.NET.0/28 -S EXTERNAL_NET=!MY.SUB.NET.0/28
```

These commandline parameters were:

-r = read from file 2003.05.22
-X = dump the raw packet data starting at the link layer
-c = use the snort.conf configuration file
-l = log to the log directory
-S HOME_NET=MY.SUB.NET.0/28 = Set the home network variable to our MY.SUB.NET.0/28
-S EXTERNAL_NET=!MY.SUB.NET.0/28 = Set the external network variable to everything not on our internal network

Snort took a minute or so to churn through the binary packet file but returned with 83 alerts logged. I had a look through the alert.ids file that was generated and found one detect that caught my eye...

```
[**] [1:499:3] ICMP Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/22-12:33:29.516782 194.196.100.59 -> MY.SUB.NET.3
ICMP TTL:245 TOS:0x48 ID:24826 IpLen:20 DgmLen:1492 DF
Type:8 Code:0 ID:50483 Seq:0 ECHO
[Xref => arachnids 246]0
```

The internal IP address MY.SUB.NET.3 was our clients external facing website so I expected some alerts on it, but this looked a bit dangerous.

2.1.3. Probability the source address was spoofed

Next step was to see who sent us the large ICMP packet. I searched for the source address of 194.196.100.59 on DShield's IPInfo tool [34] and got the results below:

DNS Name: blueice1a.uk.ibm.com
inetnum: 194.196.100.0 - 194.196.100.255
netname: GB-AGNS-NET
descr: Network of AGNS
country: GB
status: ASSIGNED PA
remarks: Service: ICS
remarks: Please send SPAM reports to postmaster@ibm.net
remarks: Please send ABUSE reports to abuse@ibm.net
route: 194.196.0.0/16
descr: AT&T GNS Ranges
descr: For routing issues: noc@attglobal.net
descr: For NEW peering issues: peering@attglobal.net
descr: For SPAM: abuse@attglobal.net
descr: For addressing issues: euabsipa@emea.att.com
origin: AS2686
mnt-by: MAINT-AS2686
changed: drueegg@emea.att.com 20021223
source: RIPE
person: Anthony Michalakopoulos

address: AGNS Firewal
address: Mailpoint C2E, c/o
address: IBM North Harbour
address: Portsmouth PO6 3AU
address: GB
phone: +44 239 256 5327
nic-hdl: AM6759-RIPE
mnt-by: EU-IBM-NIC-MNT2
changed: minas@nl.i bm.com 19991125
source: RIPE

So, who were AGNS? They seemed to have something to do with IBM so I searched for both on Google [35] and found some news from AT&T on the subject [36]. It tells us that in May 2000 over 3,000 IBM staff transferred to AT&T to join the AGNS (AT&T Global Network Services). It seems they do have something to do with IBM.

Is this source address spoofed? ICMP traffic can be spoofed easily if the attacker wants to hide their real address. It really depends what this packet is part of. If it is part of an attempted server crash, then the source IP will likely be bogus. But if the packet is a load balancing timing packet, or is part of a control connection to a DDoS bot then the source address is not likely spoofed as the sender wants a reply.

As you will find out in the next sections I believe this to be a valid IP address as there are completed TCP connections that follow the ICMP echo request packet that caused the alert.

2.1.4. Description of t he attack

The next step was to try and find the rule that that triggered this alert. It was quickly found in the ICMP rules file and is listed below:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Large ICMP Packet";
dsize: >800; reference:ara chnids,246; classtype:bad -unknown; sid:499; rev:3;)
```

I investigated the SID 449 that was listed in the rule above by looking it up on Snort.Org [37]. This rule was fully documented and is listed below:

Summary
A large ICMP packet was sent to one of your systems.

Impact
Denial of service by system crash or bandwidth utilisation.

Detailed Information
Some implementations of the IP stack may result in a system crash or hang when a large ICMP packet is sent to them. Alternatively a large number of these packets may result in link saturation, especially on lower bandwidth links.

Affected Systems
Attack Scenarios A malicious individual may send a series of large ICMP packet to a host with the intention of either crashing or hanging the host, or saturate its available bandwidth.

False Positives A number of load balancing applications use 1500 byte ICM P packets to determine the most efficent route to a host by measuring the latency of multiple paths. HP -UX

systems configured with PMTU discov ery will send echo requests in response to several types of network connecti ons. PMTU Discovery is enabled in HP -UX 10.30 and 11.0x by default.

References arachnids,246

The snort rule was designed to trigger if the ICMP was larger than 800 bytes. In the Whitehats Database entry on this subject [38], Mixter says:

" *Stateful UDP sessions are normally using small UDP packets, having a payload of not more than 10 bytes. Normal ICMP messages don't exceed 64 to 128 bytes. Packets that are reasonably bigger are suspicious of containing control traffic, m ostly the encrypted target(s) and other options for the DDOS server*."

Matt Kettler did have a word of caution though, telling of the high rate of false positives on this rule in his post on the Snort Users mailing list [39] to back up the similar suggestions made in the snort rules database.

So the attack is either an attempted DoS, control connection, MTU discovery, network speed checking or possibly just sent from a broken TCP/IP stack. I needed a closer look.

2.1.5. Attack mechanism

To see if there was any information that I was missing, I decided to recheck the binary log files. I used windump to produce a list of all traffic sent by 194.196.100.59. I used the command line below:

```
windump -r 2003-05-22 "host 194.196.100.59" > icmppkts.txt
```

I checked the icmppkts.txt file and found things were very strange indeed. The flow of traffic is listed below:

```
12:33:29.516782 IP blueice1a.uk.ibm.com > MY.SUB.NET.3: icmp 1472: echo request
                  seq 0 (DF)
12:33:29.516792 IP  blueice1a.uk.ibm.com.52784 > MY.SUB.NET.3.80: S
                  1513137258:1513137258(0) win 16384 <mss 512>
12:33:29.517426 IP MY.SUB.NET.3.80 > blueice1a.uk.ibm.com.52784: R 0:0(0) ack
                  1513137259 win 0
12:33:33.888551 IP blueice1a.uk.ibm.com > MY.SUB.NET.3: icmp 1472: e  cho request
                  seq 2 (DF)
12:33:34.212139 IP blueice1a.uk.ibm.com.53491 > MY.SUB.NET.3.80: S
                  2843312759:2843312759(0) win 16384 <mss 512>
12:33:34.212811 IP MY.SUB.NET.3.80 > blueice1a.uk.ibm.com.53491: R 0:0(0) ack
                  2843312760 win 0
12:34:14.305776 IP blueice 1a.uk.ibm.com.60114 > MY.SUB.NET.3.80: S
                  1772090:1772090(0) win 16384 <mss 512>
12:34:14.306213 IP MY.SUB.NET.3.80 > blueice1a.uk.ibm.com.60114: S
                  3178643162:3178643162(0) ack 1772091 win 8192 <mss 1460> (DF)
```

Firstly, there is a large ICMP echo request sent to our external webserver from 194.196.100.59 (blueice1a.uk.ibm.com). This is followed closely by a TCP SYN packet. Our web server replies to the SYN packet with a RST packet to kill the connection. Four seconds later there is another large ICMP echo request sent, followed by a TCP SYN packet. Our web server replies again with a RST packet to

kill the connection. 40 Seconds later the Blueice1a tries again with a SYN packet, which is responded to by a SYN/ACK packet and a standard set of HTTP web client/server connections is started. It finishes when Blueice1a finishes downloading the main page and moves on to our client's vacancies page.

The ICMP echo request messages are padded with zeroes, so I can exclude the DDoS bot control message idea. There is more than one connection from blueice1a.uk.ibm.com, some of them involving a full three-way handshake, so that shows the IP address is not likely to be spoofed. There are only two large ICMP packets sent too, so that practically removes the possibility of it being a DoS attempt on the server.

That leaves MTU discovery, network speed checking or the broken TCP/IP stack.

I decided to have a look for more information about 'Blueice' in case that part of the hostname had some kind of functional meaning. I found a webpage on IBM's research site [40] that showed that Blueice was an IBM project with aims to create next generation software for web caching, efficient distribution of multimedia content, on-demand secure virtual private networks and massively multiplayer gaming and application performance monitoring. I found a webpage from a member of the BlueProject Team, Jim Giles [41], and he states in it that he is currently working on "web caching of dynamic content".

So it seems the source of the packets could be involved in some sort of load balancing or network performance monitoring. What could it be for?

One possible scenario is for MTU discovery. If you send an ICMP echo request packet of a certain size with the 'Don't Fragment' bit set to a target computer, and you get an ICMP echo response back then you know that your messages will not get fragmented along the way and you will get a fast throughput. If you get returned an ICMP "Fragmentation needed with don't fragment bit set" message then you need to reduce the MTU size. Marc Slemko has written an overview of this process in better detail [42].

Another scenario is network performance. By sending an ICMP echo request packet to a target machine and measuring the time taken for the ICMP echo response to be sent back you can see how well a network is performing.

The third scenario is a broken TCP/IP stack. The fact that there are no OS's that I know of that 'accidentally' send a 1472 byte sized ICMP echo request, and that there is quite a high number of successful connections around that same time from the same source means that this is unlikely.

In my mind, the most likely purpose would be MTU discovery. This could be used to improve overall Internet connection times as it avoids fragmentation.

2.1.6. Correlations

I could not find any related postings from GCIA Students. Pedro Bueno did give evidence that IBM machines seem to send large ICMP packets quite often [43]. Andrew Daviel also suggested that these large packets were part of a network latency test [44]. I also noted during a search of blueice1a on Google that there were a large number of times blueice1a.uk.ibm.com appeared on peoples website scan logs. This shows the source host was used in large scale website scanning.

2.1.7. Evidence of active targeting

The ICMP echo requests were definitely targeted specifically at the webserver. They seem to be part of a larger mapping scan that trawls across the Internet discovering MTU's or measuring network performance. The echo requests do not appear to be created with malicious intent.

2.1.8. Severity

The severity of this attack can be calculated by the following formula:

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

Criticality = 4/5
The packet was targeted at the main external web server. This server is used by a great number of people to do online transactions and to view important information.

Lethality = 1/5
This is a false alarm. The packet had no real malicious intent. This means this detect should be given a minimal level of lethality.

System Countermeasures = 5/5
This web server is always patched as soon as the manufacturer releases new patches. It is part of a ongoing monitoring procedure that has been implemented to keep our clients servers updated. It also has had a HTTP URL checker/firewall installed on the host to intercept, check and possibly ignore http traffic. This allows defence from undiscovered vulnerabilities even before patches are available from the manufacturer.

Network Countermeasures = 4/5
The system is behind a firewall but is in a DMZ. The firewall is kept up to date with patches and is installed with a 'block unless expressly allowed' style security policy. There is packet filtering outside the firewall.

Overall Severity rating = (4 + 1) - (5 + 4) = -4 (scale is from –10 to 10)

2.1.9. Defensive recommendations

A recommendation would be to restrict ICMP packets to 500 bytes maximum at the boundary router. This would alleviate the possibility of attacks affecting the operating systems of internal servers.

I would also recommend the testing of the firewall to see if it can handle large ICMP packets successfully to see if the firewall manufacturer needs to be informed of any potential vulnerabilities.

2.1.10. Multichoice question

You notice an 'ICMP Large ICMP packet' alert message in you alert.ids file when you check you Snort IDS system. There are different reasons why these large ICMP messages are sent. Which of the following is not a valid reason?

A) It is an attempt to crash the computer's TCP/IP stack.
B) The attacker wants to find your computers open UDP ports.
C) The attacker wants to find the MTU between your network and their network.
D) Your ISP wants to check your network speed.

Answer: B

The attacker wants to find your computers open UDP ports. An attacker cannot discern your open UDP ports just by sending a large ICMP packet. The other three answers are valid reasons for sending an ICMP packet.

2.1.11. Incidents.Org Email Discussion

This network detect was posted to intrusions@incidents.org mailing list [45] in order to gain feedback from the list members. One reply was received stating that the original posting was good except I needed to clarify two points – the evidence of active targeting, and the defensive recommendations.

2.2. Detect 2

2.2.1. Source of Trace

The raw binary packet log used for this detect was sourced from the selection of raw Incidents.org log files available from http://www.incidents.org/logs/Raw/[33]. I selected a file from the list at random and decided upon the log from 13th May 2002 (http://www.incidents.org/logs/Raw/2002.5.13)

The Incidents.Org raw logs that are available have the following features as specified in the readme file [46]:

- Captured by Snort running in binary mode
- Only captured packets that caused an alert
- All internal addresses have been changed
- Checksums have been changed as well to stop reverse engineering of IP Addresses
- Certain keywords within the packet have been replaced with X's
- All ICMP, DNS, SMTP and Web traffic has been removed
- External IP Addresses are unchanged from those captured.

Once the log to be used had been decided, it was time to do some snooping as to what the network topology was. A great demonstration of how to do this was shown by André Cormier in his "Strange Fragmented TCP Packets" analysis post to the intrusions@incidents.org discussion list (http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00121.html [47]). I will be following a similar network topology detection process in my detect methodology today.

My first step in network topology deduction was to find out how many different Ethernet addresses were communicating on the network segment that the snort instance (as specified in the Incidents.Org README file [46]) resided. To do this I used Windump v3.8 alpha (Win32 version) [48] to scan and report how many different Ethernet address it found. I used the commandline below:

```
windump -neqr 2002.5.13 > ethcards.txt
```

These commands were:

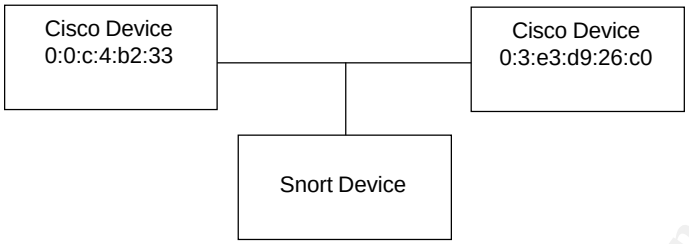
-n = disable hostname resolution
-e = show Ethernet addresses
-q = quick mode (prints less protocol information)
-r 2002.5.13 = read input from file 2002.5.13

More information on windump commands is available from http://windump.polito.it/docs/manual.htm [49]

Unfortunately, as I was using Windows 2000, I could not use the nice UNIX commands that André used, so instead I used Microsoft Excel (sorry!) to import and sort the data in the ethcards.txt file.

By sorting on source Ethernet address, I found that there were only two Ethernet addresses used - 0:0:c:4:b2:33 and 0:3:e3:d9:26:c0. I then sorted on the destination Ethernet addresses, and found only those same two addresses. This meant that the Snort instance was listening on the link between two network devices.

I then checked the Ethernet addresses against the list of Ethernet address ranges assigned by the IEEE standards organisation [50] and found that 0:0:c:4:b2:33 and 0:3:e3:d9:26:c0 both came from the address range used by CISCO SYSTEMS, INC. This meant that the network was at this point similar to this:

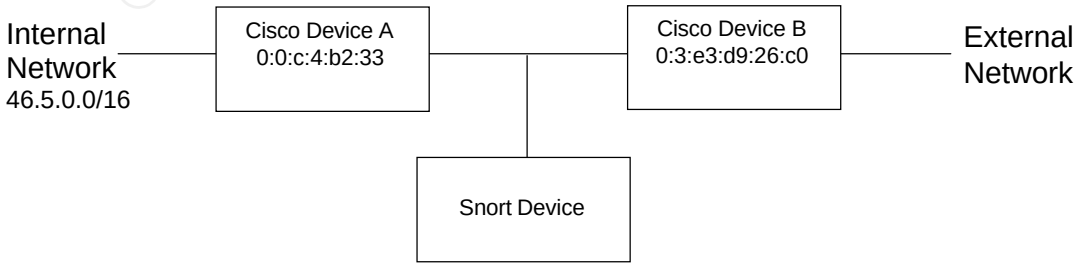


The next step was to try to discern which of the devices were attached to the internal network and which were attached to the external network. To do this we needed to look at which IP addresses were hidden behind which Ethernet addresses.

By using the ethcards.txt file generated earlier I again used Microsoft Excel to import and sort the data, and found that only two IP addresses were ever used in packets with 0:0:c:4:b2:33 source Ethernet address - 46.5.180.133 and 46.5.180.250. Checking the destination IP addresses sent to 0:3:e3:d9:26:c0 showed that no packets with IP addresses in the 46.5.x.x address range existed.

This was an initial indication that the Cisco device with MAC address of 0:0:c:4:b2:33 was the internal device. Further checking revealed that all packets generated with source Ethernet address 0:3:e3:d9:26:c0 had a destination IP address within the 46.5.x.x address range. It was also noted that there were no 46.5.x.x source addresses in any packets sent from 0:3:e3:d9:26:c0 source Ethernet address, but this information was less convincing as it is easy to spoof an IP address if wanted by using a tool such as hping2 [51] or sendip [52].

The 46.5.x.x addresses ranged from 46.5.0.78 to 46.5.253.225 inclusive, which indicated a netmask of 255.255.0.0 (or /16). Combined with the information previously discussed, it seemed that the internal network was 46.5.0.0/16. We can make an assumption about the network topology and draw a diagram:



Just as André did, I decided to check if there was any sort of incoming filtering on the Cisco Device B. I ran the following commandline to check what destination ports were allowed through:

```
windump -neqr 2002.5.13 "eth er src 0:3:e3:d9:26:c0" > ethcardports.txt
```

These commandline parameters were:

-n = disable hostname resolution
-e = show Ethernet addresses
-q = quick mode (prints less protocol information)
-r 2002.5.13 = read input from file 2002.5.13
"ether src 0:3:e3:d9:26:c0" = only select packets with the Ethernet address of 0:3:e3:d9:26:c0.

The text file was again imported into Excel and sorted to show what destination ports were being allowed through the boundary routers. They were:

Port Number	Common Usage
21	FTP control instructions
53	DNS
80	HTTP
137	NETBIOS Name Service
515	UNIX Print Spooling (LPR)
1024	Reserved by IANA, Common NetSpy Port
1080	SOCKS Proxy
40195	Unassigned by IANA
>6100	RPC, Various

At first glance, it seems that there was some form of packet filtering on the Cisco Device B, but as the readme file [46] states, only the packets that caused an alert were recorded. This means that it was more likely that the snort rules were configured to watch the common well-known ports and those higher dynamic ports often used for RPC communications.

2.2.2. Detect was generated by

To find the detect I used Snort Version 2.0.0-ODBC-MySQL-WIN32 (Build 72) [31] running the current snort rules [32] (as at 25 April 2003). I used this version because this version does not suffer from the stream4 pre-processor vulnerability [53] that earlier versions have. The commandline I used is below:

```
snort -d -e -r c:\snort\rawlogs\2002.5.13 -c snort.conf -l c:\snort\log -S
        HOME_NET=46.5.0.0/16 -S EXTERNAL_NET=!46.5.0.0/ 16
```

These commandline parameters were:

-d = dump the application layer
-e = show link-layer information
-r = read from file 2002.5.13
-c = use the snort.conf configuration file
-l = log to the log directory
-S HOME_NET=46.5.0.0/16 = Set the home network variable to our previously discovered 46.5.0.0/16

-S EXTERNAL_NET=!46.5.0.0/16 = Set the external network variable to everything not on our internal network

The alert.ids file that was generated contained many Nmap scans and DNS Named version attempts, but one strange thing I noticed was that the timestamps on the packets was different from the date in the filename. The date of the packet capture was 13 June 2002, not 13 May 2002 as the log was labelled on the Incident.Org website. When I had checked some other practicals that used the raw logs from about this time, I noted that they had reported the same issue. I assumed the date on the file was wrong (possibly a mistake in the script that manages the daily log file splitting) and that the date of the snort data as the correct date.

I decided to look at the two "BAD-TRAFFIC ip reserved bit set" alerts listed below more closely:

```
[**] [1:523:4] BAD-TRAFFIC ip reserved bit set [**]
[Classification: Misc activity] [Priority: 3]
06/13-17:30:26.004488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
192.1.1.188 -> 46.5.28.40 TCP TTL:239 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0x0014

[**] [1:523:4] BAD-TRAFFIC ip reserved bit set [**]
[Classification: Misc activity] [Priority: 3]
06/13-21:44:51.094488 0:3:E3:D9:26:C0  -> 0:0:C:4:B2:33 type:0x800 len:0x3C
192.1.1.188 -> 46.5.39.57 TCP TTL:239 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0x0014
```

This detect was generated by the bad-traffic.rules file by the rule listed below:

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

To understand why this rule triggered, I checked the Snort Rules Database to see if there was documentation about what this rules' purpose was. I searched using the SID 523 in the Snort database at Snort.Org [54] and got:

SID: 523
message: BAD-TRAFFIC ip reserved bit set
Signature: alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC ip reserved bit set";
fragbits:R; sid:523; classtype:misc-activity; rev:4;)
Summary: This event is generated when packets on the network have the reserved bit set.
Impact: Possible prelude to system compromise.
Detailed Information: Under normal circumstances IP packets do not use the reserved bit. This 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.
Affected Systems: All
Attack Scenarios: The attacker may send specially crafted packets with the reserved bit set.
Ease of Attack: Simple
False Positives: None Known
False Negatives: None Known
Corrective Action: Use a packet filtering device to reject packets with this bit set.
Contributors: Original rule writer unknown. Sourcefire Research Team - Nigel Houghton <nigel.houghton@sourcefire.com>

As we can see from the database excerpt above, the alert was raised because the IP Flags reserved bit was set for that packet. I checked the IP Protocol Specification (RFC 791) [55] and found that RFC 791 states that the IP Control Flags Bit 0 is "… reserved, must be zero ...". Which led me to the obvious question - why was it not in these two packets?

Just to make sure that it was not a false alert I ran Windump over the same 2002.5.13 raw log file using the commandline below:

```
windump -xvr 2002.5.13 "ip[6] &  0x80 != 0"
```

These commandline parameters were:

-x = show the Hex as well
-v = show verbose mode
-r 2002.5.13 = read input from the file 2002.5.13

The BPF filter at the end of the commandline was designed to select all the packets from the 202.5.13 file with the IP reserved bit set (IP offset byte 6). The output from the command was:

```
17:30:26.004488 IP (tos 0x0, ttl 239, len 40) 192.1.1.188 > 46.5.28.40: tcp
(frag 0:20@17184)bad cksum 3e88 ( ->3781)!
                          4500 0028 0000 8864 ef06 3e88 c0 01 01bc
                          2e05 1c28 0e64 0050 05d0 bef2 05d0 bef2
                          0004 0000 62c4 0000 0000 0000 0000

21:44:51.094488 IP (tos 0x0, ttl 239, len 40) 192.1.1.188 > 46.5.39.57: tcp
(frag 0:20@17184)bad cksum 3377 ( ->2c70)!
                         4500 0028 0000 8864 ef06 3377 c001 01bc
                         2e05 2739 098f 0050 06b9 ad7e 06b9 ad7e
                         0004 0000 7d9e 0000 0000 0000 0000
```

The reserved flag is found in the sixth byte offset (88). The IP Flag structure in both these two packets are listed below:

Res	DF	MF
1	0	0

Where: Res = Reserved bit (Must be Zero)
DF = Don't Fragment
MF = More Fragments to follow

As you can see the reserved flag was the only IP flag set, and it's the only flag that was not supposed to be! Interestingly when I viewed these packets using Ethereal v0.9.12 on Windows 2000, I saw that although the raw data showed that there was a Reserved Bit set, the GUI only decoded that the DF and MF flags were not set, and did not mention the fact the reserved bit was set. This shows the danger of relying on the logic built into IDS systems, and the benefit of raw packet capture for future analysis and correlation.

2.2.3. Probability the source address was spoofed

To check if the source address was spoofed, I searched on both InterNIC [56] and ARIN Whois Searches [57]. The ARIN search provided the following:

Search results for: 192.1.1.188

BBN Communications BBN-CNETBLK (NET-192-1-0-0-1)
192.1.0.0 - 192.1.255.255
Bolt Beranek and Newman Inc. BBN-WAN (NET-192-1-1-0-1)
192.1.1.0 - 192.1.1.255

A lookup on the BBN-WAN networks showed this information:

OrgName: BBN Communications
OrgID: BBNP
Address: 150 CambridgePa rk Drive
City: Cambridge
StateProv: MA
PostalCode: 02140
Country: US

Doing a search using Yahoo Search Engine [58] for BBN Communications brought up RFC1141 [59], in which T. Mallory and A. Kullberg both work for BBN Communications and have bbn.com email addresses and work at 50 Moulton Street, Cambridge MA. The company seemed to have moved. By visiting Bbn.Com, I found that it was a company that was involved in the original ARPANET construction and design [60]. I wondered if the 192.1.1.0 addresses were used for testing or for private networks, as it seemed like their assigned addresses were quite close to the 192.168.x.x private network addresses. I tried to delve a bit deeper.

A search on Yahoo for "192.1.1" found an interesting article by Jon Udell on Byte.Com [61], which seemed to answer my question. In it he states that use of the 192.1.1.x address for private IP network addressing is like an "internet folk tradition".

"Why isn't 192.1.1 one of the Class C networks that RFC 1597 proposes t o reserve? Well, you can't always tr ust folk traditions. As it turns out, 192.1.1 is a registered network. Walter "Doc" Urbaniak, a network engineer with BBN Corporate Telecommunications, contacted me to set the record straight.
For years, 192.1.1 has bee n registered to Bolt, Beranek, and Newman. Urbaniak says that because 192.1.1 is widely but wrongly believed to be available for testing, BBN doesn't even try to use it anymore. "We even found some instances in which vendors were shipping hardware preconfigured for the 192.1.1 network," he adds."

To confirm this, I searched the Internet RFC archives and found two articles. RFC1166 [62] had a listing of the early assigned networks, one of which was the network that we wanted:

R 192.1.0.rrr-192.1.1.rrr BBN Local Nets [SGC]

This is definitely not a private network range IP address as per RFC1918 [63]. Considering that BBN admits that it does not even use those addresses any more, it seems unlikely that this packet originated from BBN Communications. As John Udell's article was written over 7 years ago, BBN may have changed its personnel and it's policy, and could have begun utilising the 192.1.1.x IP address range again.

It is possible that some older routing equipment that had this address shipped as standard was being used somewhere else on the local private network but this is unlikely.

It is also possible that the source IP address of 192.1.1.188 was spoofed. This choice may have been made so that the chances of being let into a network that used the same (unofficial) private addressing scheme would be improved. It could be that the choice of this address would guarantee that no machine at 192.1.1.188 would respond. Alternatively, it could be that this choice was just random. It is nearly impossible to say.

There are three possibilities:

- The IP address is not spoofed and the packet originates from 192.1.1.188 at BBN Communications.
- The IP address is not spoofed and it originates from a router that has resorted back to it's original default configuration.
- The IP address is spoofed and just has selected the IP address for its increased likeliness of being overlooked.

It is my belief that the IP address is probably spoofed, as it is unlikely that BBN Communications is using the IP address range at all anymore. To be fully sure we would need to contact the system administrators at BBN.

2.2.4. Description of the attack

I believe it is impossible to give a conclusive analysis of what the purpose of this packet is due to the lack of availability of all the data. As the Incident.Org Raw log readme states *"...only the packets that violate the ruleset will appear in the log."* This means we are missing some of the packets needed for a full analysis.

As you will see in the next section, I have speculated on six possible purposes of these crafted packets. Firstly, that this could be a tiny fragments attack designed to evade the NIDS. Secondly, that it might be an OS fingerprinting reconnaissance scan. Thirdly, that it is a DoS attempt. Fourthly, that it possibly is a control packet for a Trojan. Fifthly that is it a misconfigured router. Lastly, that it is a response to stimuli sent from internal networked devices, so didn't originate from the local network. My feeling though is that it was most likely an OS fingerprinting scan or a trojan control packet using tiny fragments in an attempt to evade the NIDS.

2.2.5. Attack mechanism

In order to try to establish a purpose for these packets we need to ask some questions about the design of the packets. What exactly can we find out? Some questions spring to mind....

Are the packets part of a fragment train?

RFC791 [55] says that *"The fragmentation strategy is designed so than an unfragmented datagram has all zero fragmentation information (MF = 0, fragment offset = 0)."* Both these packets have a non-zero fragment offset, so they must be part of a fragment train (or at least pretending to be).

Are the packets the initial fragment, part of the middle of a fragment train, or the last fragment?

- To qualify as an initial fragment according to the IP specification you must have the More Fragments (MF) bit set, and a fragment offset of zero. This is not the case in our packets so they cannot be the initial fragments in the fragment train.
- To qualify as a middle fragment it needs to have the MF bit set, and a non-zero fragment offset. These packets do not have the MF bit set but do have a non-zero fragment offset. The packets cannot therefore be the middle fragments of the fragment train.
- To qualify as the last fragments in a fragment train they must have a non-set MF bit and a non-zero fragment offset. This is exactly what these packets have.

What is the packets' purpose?

This is difficult to say without a full packet trace but there are some possibilities.

- **NIDS evasion attempt.**
  By sending tiny fragments through the network, it may perhaps bypass the rules used in the NIDS. McAfee ASaP site discusses some vulnerabilities based on this [64], of which "12042 IP - fragmentation - tiny fragment with reserved bit set" most closely matches our case. Further investigation of RFC1858 [65] found that:

  > *"With many IP implementations it is possible to impose an unusually small fragment size on outgoing packets. If the fragment size is made small enough to force some of a TCP packet's TCP header fields into the second fragment, filter rules that specify patterns for those fields will not match. If the filtering implementation does not enforce a minimum fragment size, a disallowed packet might be passed because it didn't hit a match in the filter."*

  So it could be that this is a tiny fragments attack, specifically designed to avoid detection. There may have been some other attack code in the previous fragments in the fragment train that were not logged which were part of the exploit. To know this for sure we would need to examine all traffic from the hosts 46.5.28.40 and 46.5.39.57 so see if they sent any responses.

- **OS fingerprinting scan.**
  Ofir Arkin posted an email on the bugtraq@securityfocus.org mailing list late 2000 discussing how to use the reserved bit in the IP packet to do OS fingerprinting. Apparently, if you send an ICMP echo request to a Solaris 8.0 or HPUX 11.0 computers they would echo the reserved bit back. This information can then be used with other attempts to send incorrectly formatted packets to 'fingerprint' the Operating System.

  This argument is lent extra weight by the fact that the TTL is set at 239 in both packets. Based on information in the GCIA course, the Passive Fingerprinting article on incidents.org [67] and the default TTL list on Nerim.Net [68], this packet has likely come from an operating system with an initial TTL value of 255. By comparing the TTL value with the TOS value and even the IP ID value we can see where this has likely come from [69]. We see that the TTL value is 255. The TOS is 0. This is probably either a Solaris 2.x machine or a Cisco 12.0. The

decisive factor for me though is the Initial IP ID of 0 that we have. This correlates well with the initial IP ID that Cisco IOS 12.0 uses on an initial connection.

I doubt that the Cisco IOS 12.0 device is really being used for OS Fingerprinting. I believe the packets are crafted and as such may contain values that just happened to match the values in the OS fingerprinting article.

- **Denial of service attack**
  This idea was due to an article I found on NetBSD website [70] which explained a vulnerability in testing with the defragmentation code in the TCP/IP stack. My thinking was that the attacker could be running a NetBSD machine or another machine that has a problem with processing fragmented packets and would consume system resources:

  > *"The code masks out the DF bit and then, if any of the rest of the bits in the structure are set, the packet is considered a fragment. But this test will pass, and the fragment code run, even if the packet is not a fragment and just the "must be zero" reserved flag is set."*

  To check I scanned through the logs from the 10 May 2002 to 16 May 2002 (see Appendix B) and found that the maximum packet captures of this nature in a day was 14. This of course was way below the levels expected of a DoS attack and the hypothesis was dismissed immediately.

- **Trojan control packet**
  It could be pretending to be the last fragment in the fragment train, yet really be the only fragment sent. By sending with the reserved IP flag set it could be attempting to bypass the NIDS and control the Trojan. I checked some Trojan port sites [71] and found that it would be nearly impossible to research this further without the original full packet traces, as we would need to examine all traffic from the hosts 46.5.28.40 and 46.5.39.57 to see if they sent any suspicious traffic to other external addresses. It would also be a good idea to check the two computers involved to see if there were any new software installed on these machines.

- **Misconfigured router on the local network**
  As I had learnt earlier, the 192.1.1.x address range had been used in the past (incorrectly) as a private internal IP addressing range. It was possible that this addressing schema was still in use in one router within this organisation or that the router had reverted back to it's default settings. This is unlikely though, as the address 192.1.1.188 was only seen coming from 0:3:e3:d9:26:c0, which we had previously established was the external router. This meant there was only a slim chance that this was the case so I ruled it out.

  It could also be that there was a problem with a Cisco 12.0 router, which was setting the reserved bit to one and fragmenting all the data. While possible, this is not plausible, as it would have meant that all traffic sent through that router would experience the same problem, and each connection would not be established. I expect this would be detected fairly quickly - be it by the network administrator or even the users on the other side of the router. I ruled this out as well.

- **Response to stimuli sent from internal hosts**
  This would mean that the packets would have to be to the RFC specifications, which we know is not true. Packets that contain a set IP Reserved Flag are not legal.

My belief is that it is most probably either an OS fingerprinting attempt or a trojan control packet using a covert communications channel.

2.2.6. Correlations

The closest item I could find on Google was a detect completed previously by GIAC students. I agreed with the findings by T. Brian Granier [72] that this detect could be an evasive OS fingerprinting attempt, but I also thought it could be a covert trojan control communication attempt. A second detect by Brent Wrisley [73] came to the same conclusion as T Brian Granier. A third detect by Ron Shuck [74] was similar in nature as it was triggered by a similar rule, and is worth further reading if desired.

2.2.7. Evidence of active targeting

Based on the random nature of the packets that were captured, I believe that these packets were part of an evasive attack using tiny fragments to evade NIDS detection. As such, I do not believe that the destination IP addresses were selected from prior reconnaissance but instead were targeted randomly.

2.2.8. Severity

The severity of this attack can be calculated by the following formula:

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

Criticality = 2/5
It is very hard to say how critical the systems are that were targeted. As such, I will assume that they were mostly user systems, and rate that accordingly.

Lethality = 3/5
My initial consideration was that this rating should be given a four due to the evasion techniques employed. If this much work has gone into designing an evasion technique then it would be sensible to assume the same care went into devising a system of covert communication or an OS fingerprinting technique. This rating was deemed too high though because of the untargeted nature of the scan. This is an attack that is likely to be polling the Internet looking for open systems, and probably notifies the attacker in some way when the attack succeeds. I arrived at the figure of three by assuming that what we were viewing the end fragments of a scan or covert communication fragment train. Although the attacks were not necessarily successful, they still were tried against some of the computers on the local network.

System Countermeasures = 1/5
This figure takes into account the fact that we do not know what services were being targeted, and what level of patching was reached on the network devices. I need to assume a minimal level.

Network Countermeasures = 3/5
This figure takes into consideration the fact that we could not prove ingress filtering. We have to assume though that adequate protection has been made however, as the fact we have logs to view is an indication that the system administrator knows the value of an NIDS. For this reason I gave the figure of 3 for network countermeasures.

Overall Severity rating = (2 + 3) - (1 + 3) = 1 (Scale is from –10 to 10)

2.2.9. Defensive recommendations

My first defensive recommendation would be to contact the systems administrators at BBN Communications to ask if they still use the IP address 192.1.1.188. If they do not, then the IP address is spoofed, and the targeted machine needs to be checked for possible trojans. If they do still use that address, then it is likely that the IP address was not spoofed and they need to check if 192.1.1.188 is infected with a trojan. Next, I would recommend that they put in a firewall if there is not one already. It would need to be configured to make sure that the complete TCP header was encapsulated in the first fragment, as per the recommendations in RFC1858 3.2.1 Direct Method [65]. I would also make sure that the firewall could handle other issues like overlapping fragments as well. I would enlarge the IDS rules to log any traffic to or from any of the local IP addresses listed in Appendix B. This would give me more traffic to check, without overloading the NIDS (if it is a high traffic environment). I would also make sure the computers on the local network were patched to the latest OS and application patches in case they are attacked with a similar attack.

2.2.10. Multichoice question

```
17:30:26.004488 IP (tos 0x0, ttl 239, len 40) 192.1.1.188 > 46.5.28.40: tcp
(frag 0:20@17184)bad cksum 3e88 ( ->3781)!
                        4500 0028 0000 8864 ef06 3e88 c001 01bc
                        2e05 1c28 0e64 0050 05d0 bef2 05d 0 bef2
                        0004 0000 62c4 0000 0000 0000 0000
```

The above packet has the following characteristics:

A) The IP ID of 0x0028
B) The Don't Fragment bit is set
C) The IP Reserved bit is set
D) It contains a TCP packet that is larger than standard

Answer: C

The IP Reserved bit is set. RFC791 states it should never be set.

2.3. Detect 3

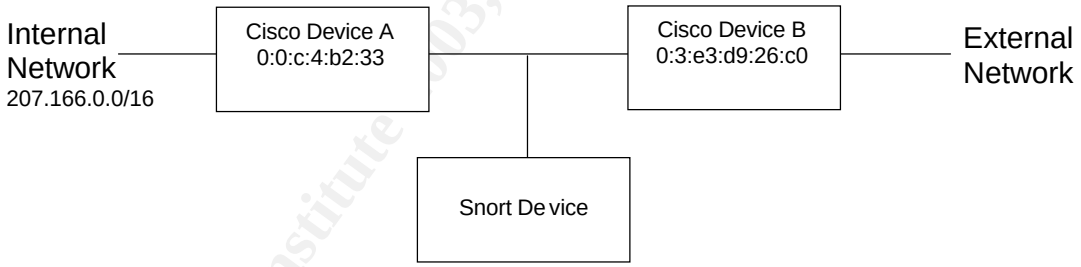
2.3.1. Source of Trace

The raw binary packet log used in for this detect was sourced from the selection of raw Incidents.org log files available from http://www.incidents.org/logs/Raw/[33]. I selected a file from the list that was closer to the end of the 2002 year and decided upon the log from 10th October 2002 (http://www.incidents.org/logs/Raw/2002.10.10)

The Incidents.Org raw logs that are available have the following features as specified in the readme file [46]:

- Captured by Snort running in binary mode
- Only captured packets that caused an alert
- All internal addresses have been changed
- Checksums have been changed as well to stop reverse engineering of IP Addresses
- Certain keywords within the packet have been replaced with X's
- All ICMP, DNS, SMTP and Web traffic has been removed
- External IP Addresses are unchanged from those captured.

Now that I had a log selected, I needed to find a little about the network topology. Following the same process that I had used in my previous detect #2, I came up with the following topology:



Just as a side point, the 2002.10.10 and 2002.5.13 files seemed to have the same network topology used, with the real IP address of the Internal Network obfuscated with different numbers to protect the real network identity.

Also, note that although the file was called 2002.10.10 they contained the binary logs from 10 November 2002.

2.3.2. Detect was generated by

To find the detect I used Snort Version 2.0.0-ODBC-MySQL-WIN32 (Build 72) [31] running the current snort rules [32] (as at 25 April 2003).

The commandline I used is below:

```
snort -d -e -X -r c:\snort\rawlogs\2002.10.10 -l c:\snort\logs -c
             c:\snort\rules\snort.conf -S HOME_NET=207.166.0.0/16 -S
             EXTERNAL_NET=!207.166.0.0/16
```

These commandline parameters were:

-d = dump the application layer
-e = show link-layer information
-r = read from file 2002.10.10
-X = dump the raw packet data starting at the link layer
-c = use the snort.conf configuration file
-l = log to the log directory
-S HOME_NET=207.166.0.0/16 = Set the home network variable to our previously discovered 46.5.0.0/16
-S EXTERNAL_NET=!207.166.0.0/16 = Set the external network variable to everything not on our internal network

The output generated was:

```
Running in IDS mode
Log directory = c: \snort\logs
TCPDUMP file reading mode.
Reading network traffic fr om "c:\snort\rawlogs\2002.10.10" file.
snaplen = 1514

        -=== Initializing Snort == --
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug -ins!
Parsing Rules file c: \snort\rules\snort.conf

++++++++++++++++++++++++++++++++++++ ++++++++++++
Initializing rule chains...
---[SNIP]---
        -=== Initialization Complete == --

-*> Snort! <*-
Version 2.0.0 -ODBC-MySQL-WIN32 (Build 72)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
1.7-WIN32 Port By Michael Davis (mike@datanerds.net, www.datanerds.net/~mike)
1.8 - 2.0 WIN32 Port By Chris Reid (chris.reid@codecraftconsultants.com)
Run time for packet processing was 137.404000 seconds


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

Snort processed 12503 packets.
Breakdown by protocol:                Action Stats:

    TCP: 12498      (99.960%)         ALERTS: 12145
    UDP: 0          (0.000%)          LOGGED: 12150
   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: 29         (0.232%)
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
===============================================================================

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

Snort exiting
```

After checking out the alert.ids file there was a pair of detects that caught my eye...

```
[**] [1:1394:3] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/10-17:12:47.636507 0:3:E3:D9:26:C0  -> 0:0:C:4:B2:33 type:0x800 len:0x5EA
130.94.22.249:80  -> 207.166.87.157:64741 TCP TTL:52 TOS:0x0 ID:23339 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0xD454C3BC  Ack: 0xAC7EEF02  Win: 0x1920  TcpLen: 20

[**] [1:1394:3] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/10-17:13:13.236507 0:3:E3:D9:26:C0  -> 0:0:C:4:B2:33 type:0x800 len:0x5EA
130.94.22.249:80  -> 207.166.87.157:64741 TCP TTL:52 TOS:0x0 ID:23403 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0xD456219D  Ack: 0xAC7EF2EC  Win: 0x2180  TcpLen: 20
```

The rule that triggered this alert was found in the shellcode.rules file and is listed below:

```
alert ip $EXTERNAL_NET any  -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86
NOOP"; content:"|61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61|"; classtype:shellcode-detect; sid:1394; rev:3;)
```

A quick lookup on Snort.Org [75] for SID 1394 showed that this was one of the non-documented rules. So I did a search on Google [35] and found a couple of good posts from Bryan Burns [76] and Robert David Graham [77] about possible false positives on these rules. Bryan said:

> *"The "Shellcode" set of signatures are trying to look for generic attacks that have not been discovered yet by looking for patterns in network traffic that appear to be dangerous or common CPU instructions used in hax0ring attempts. These signatures are prone to a high false-positive rate though, and often get triggered by perfectly harmless data."*

Just to make sure I was not doing something wrong I viewed these packets using Ethereal v0.9.12 on Windows 2000. Ethereal decoded the packets as a HTTP continuation - meaning it thought they were from a http connection already in process.

2.3.3. Probability the source address was spoofed

To check if the source address was spoofed, I searched on both InterNIC [56] and ARIN Whois Searches [57]. The ARIN search provided the following:

Search results for: 130.94.22.249

OrgName: Verio, Inc.
OrgID: VRIO
Address: 8005 South Chester Street
Address: Suite 200
City: Englewood
StateProv: CO
PostalCode: 80112
Country: US
NetRange: 130.94.0.0 - 130.94.255.255
CIDR: 130.94.0.0/16
NetName: VRIO-130-094
NetHandle: NET-130-94-0-0-1
Parent: NET-130-0-0-0-0
NetType: Direct Allocation
NameServer: NS0.VERIO.NET
NameServer: NS1.VERIO.NET
NameServer: NS2.VERIO.NET
*Comment: **********...*
Comment: Reassignment information for this block is
Comment: available at rwhois.verio. net port 4321
*Comment: **********...*
RegDate: 2000-07-11
Updated: 2001-09-26

Doing a search using Yahoo Search Engine [58] for "Verio, Inc" gave me a link to a NTT Communications subsidiary called NTT/Verio [78]. This is an Internet Service Provider with a worldwide coverage. I checked their Whois Service using the GEEKTOOLS Whois proxy [79] to see who was using the range that we were interested in. I got this answer back:

Rwhois server data:
---[SNIP]---
network:IP-Network-Block:130.94.22.248 - 130.94.22.251
network:Org-Name:Drive Savers Data Recover, Inc.
network:Street-Address:400 Bel Marin Keys Boulevard
network:City:Novato
network:State:CA
network:Postal-Code:94949
network:Country-Code:US
---[SNIP]---
network:IP-Network-Block:130.94.16.0 - 130.94.31.255
network:Org-Name:Verio Advanced Hosting - Dulles
network:Street-Address:22451 Shaw Rd
network:City:Sterling
network:State:VA
network:Postal-Code:20166
network:Country-Code:US
network:Tech-Contact;I:IA17312-VRIO.127.0.0.1/32
network:Created:2001-06-15 18:07:22+00
network:Updated:2001-06-15 18:07:22+00

As you can see from the above, the web address range was assigned to Drive Savers Recover, Inc. They were being hosted on servers run by Verio Advanced Hosting. To check this I went to InfoBear.Com's NSLookup Webpage [80] and

queried 130.94.22.249. I got:

Output of:
nslookup -q=A www.drivesavers.com ns1.worldnet.att.net
Server: ns1.worldnet.att.net
Address: 204.127.129.1

Name: www.drivesavers.com
Address: 130.94.22.249

I visited the Drivesavers webpage on port 80 and found a valid webpage.

In summary, I do not believe this IP address was spoofed at all. Both the packets triggered in the alert had the acknowledgement flag set, which means they had performed the three-way handshake. Both packets also had a source port of 80 which we later found was confirmed as the 130.94.22.249 address was confirmed as a website address.

2.3.4. Description of the attack

In an attempt to find out more about these shellcode rules I entered "SHELLCODE x86 NOOP" into a search on Google [35] and added the page from Dragos Ruiu [81] to the two pages from Bryan Burns [76] and Robert David Graham [77] about possible false positives on these rules.

The gist of these three posts are that all these records are searching for are sequences of hexcodes that match common NOOP (no operation) hexcodes. These hexcodes are often used in buffer overrun exploits so that the attacker does not need to worry about exact memory locations when they attempt to change program execution. By putting a NOOP sled in the packet, the execution jump can be a little off, and the execution will just 'skip over' the NOOP commands until it reaches the code the attacker wishes to run.

All posts also make another point very clear. The "SHELLCODE x86 NOOP" rule has a very high false positive rate. All it requires is a datagram payload that happens to contain the same content, and the rule will fire. JPG, PNG, GIF, PICT or Document (MSWord etc.) are commonly affected by this rule. This is one of the reasons that this rule is commented out by default in the standard snort rule distribution.

When scanning through the hex display of the packet captured by snort, I noticed the JFIF marker in the file. I know from past experience that this indicates a JPEG file (or derivative). I decided to see if I could extract the JPEG from the datagram and view it.

First step was to find out the JPEG file structure. Using Google I found the official JPEG specification [82], and an article called "How do I recognize which file format I have, and what do I do about it?" by Tom Lane [83]. Using these articles, I attempted to retrieve the data. I was unsuccessful. It looked like only the JPEG header was captured in both packets, and that the rest of the JPEG file was in following packets

that weren't captured by the snort instance that created the 2002.10.10 raw log file. As you will see in the next section, I managed to decode some parts of the packet, and concluded that this was indeed a false alarm.

2.3.5. Attack mechanism

A full listing of both packets that match the SHELLCODE x86 NOOP alert rule is available in Appendix C. I will go through the first packet to decode it:

```
The Ethernet Header [19]
------------------
Destination MAC Address: 00 00 0C 04 B2 33
Source MAC Address: 00 03 E3 D9 26 C0
Ethernet Type Field: 08 00 (IP Protocol)

The IP Packet Header [20]
--------------------
IP Version: 4
Header Length: 5 words = 20 bytes
Type of Service: 00
IP Total Length: 05 DC
IP Identification: 5B 2B
40 00
Time To Live: 34
Protocol: 06 = TCP
Checksum: 6F A0
Source IP Address: 82 5E 16 F9
Destination IP Address: CF A6 57 9D

The TCP Packet Header [21]
----------------------
Source Port: 00 50
Destination Port: FC E5
TCP Sequence Number: D4 54 C3 BC
TCP Acknowledgement Number: AC 7E EF 02
Header Length: 5 words = 20 bytes
Reserved: 0
TCP Flags: Acknowledgement Flag
TCP Window Size: 19 20
Checksum: 14 69
TCP Urgent Pointer: 00 00

The Payload
-----------
5B AD D5 58 40 00 FF 04 04 00
00 00 00 00 00 00 40 00 FF 04
----[SNIP]----
2E 19 19 2E 61 41 37 41 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 6 1 61 61 61 61 61 61 61 61
61 61 61 61 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00
----[SNIP]----

Jpeg File [17,18]
----------
JPEG Marker: FF D8 FF E0
Length: 00 10
JFIF Marker: 4A 46 49 46 00
JPEG Version: 01 01
Units : 00 = Pixels
X density: 00 01
Y density: 00 01
X Thumbnail: 00
Y Thumbnail: 00
```

```
Section Marker: FF C0
Picture Data: 00 11 08  ---[etc until packet end]---
```

As you can see from the above, the first packet contained a JPEG file embedded in the payload. After an extreme amount of searching, I was unable to find any other matching parts to fully decode this payload.

So now, for the million-dollar question - Is this an attack?

It is my belief that this not an attack. This is a false alarm generated by the hex 61's in the jpeg file embedded in the datastream. Several things made me come to this conclusion. Firstly, the flow of traffic is from port 80 on the source, which is a well known port used by http. The traffic is going to a high ephemeral destination port (64741) on 207.166.87.157. This is the action of a response to a connection already made by 207.166.87.157.

Can we be sure this is a web style connection? We can check host 207.166.87.157's role by investigating what connections are being made and received by it. I did this by running the command:

```
windump38a -vr 2002.10.10 "host 207.166.87.157"
```

This lists out all the connections made to and from 207.166.87.157. It turned out there were 288 alerted connections from an ephemeral port to port 80, yet only 8 alerted connects back in the other direction. This makes me sure that host 207.166.87.157 is a http proxy server, and is protecting the internal network computers from having direct access to the Internet.

So the facts so far:

- The source of 130.94.22.249:80 is a valid web site
- The destination of 207.166.87.157:64741 is a likely http proxy server
- The payload contains a JPEG file
- In the middle of the JPEG is a sequence of NOOP instructions

This therefore is a response to earlier stimuli sent by the http proxy server requesting some data, possibly a JPG, PNG, PICT or Document with embedded pictures, of which we are seeing only one datagram. This can be lent extra weight by the fact that both packets are the maximum Ethernet datagram size of 1514 bytes, which shows there is probably more data to follow.

2.3.6. Correlations

The closest item I could find from any GCIA Students was a posting by Tyler Hudak [84]. Apart from the aforementioned Dragos Ruiu [81], Bryan Burns [76] and Robert David Graham [77] I could not find anything further of note.

2.3.7. Evidence of active targeting

These packets were definitely targeted at the internal http proxy server by the external web server, but this is correct behaviour. The detect is a false positive.

2.3.8. Severity

The severity of this attack can be calculated by the following formula:

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

Criticality = 4/5
The packet was targeted at the http web proxy server. This is quite an important piece of kit, and should be rated as such.

Lethality = 1/5
As this is a false alarm this detect should be given a minimal level of lethality.

System Countermeasures = 1/5
This figure takes into account the fact that I do not know what patches had been applied on the http web proxy server. I need to assume a minimal level.

Network Countermeasures = 3/5
This figure takes into consideration the fact that we could not prove ingress filtering. We have to assume though that adequate protection has been made however, as there is a http web proxy server, and we have NIDS logs to view, is an indication that the system administrator knows at least a little about network security. For this reason I gave the figure of three for network countermeasures.

Overall Severity rating = (4 + 1) - (1 + 3) = 1 (Scale is from –10 to 10)

2.3.9. Defensive recommendations

As this detect is a false positive there are not many defensive measures to be taken. I would make sure that the http proxy server was patched with the latest patches, and I would investigate the proxy server's logs to make sure they did not show any abnormal behaviour.

I would also put in a firewall if there wasn't one already as there needs to be control over what ports are exposed to the Internet. The less doors to get in, the less work there is protecting those doors.

2.3.10. Multichoice question

```
alert ip $EXTERNAL_NET any  -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86
NOOP"; content:"|61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61|"; classtype:shellcode -detect; sid:1394; rev :3;)
```

The above snort rule is designed to:

A) Create no false positives
B) Detect as many possible NOOP Sleds as possible
C) Detect the ASCII characters "61" in a document
D) Detect the words "SHELLCODE x86 NOOP" in a document

Answer: B

Detect as many NOOP sleds as possible. This process can cause many false positives but can detect possible exploits before specialised rules are generated for them.

Assignment 3. Analyze This

3.1. Executive Summary

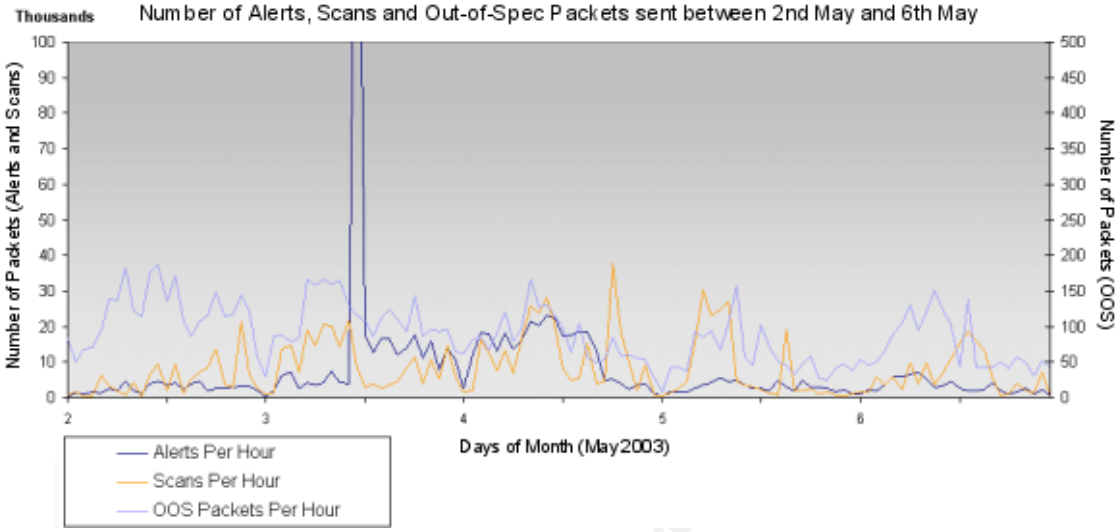


Figure 1 - Number of Alerts, Scans and Out -of-Spec packets sent between 2 nd May and 6 th May

The computer networks of today are important communication enablers. In some instances the need of the computer network is as great as the need for electricity itself. The SANS University network is no different. Being a large and diverse University, the SANS University network has a large amount of important traffic flowing through it. Managing something of that size requires good management practices, and a well-drilled team. Often in today's world of reduced profits and smaller budgets mean that an analysts' time cannot be spared for in-depth analysis of network traffic. It is for this reason that I was asked to produce this report.

The above graph was provided to give an overview of the trends during the period of time this author studied the SANS University network. It shows the hourly rates of three measured Snort metrics - the number of alerts, scans and Out-of-spec files per hour. The alerts per hour are important as they demonstrate that number of times per hour that someone has sent a packet that went against the University 'rules'. The scans per hour show how many times somebody tested to see if the university was protected, or what computers were in use. And the Out-Of-Specs per hour show how many times broken packets were sent – sometimes by accident, sometimes on purpose.

As you can see, late morning on the 3rd of May 2003 had the highest number of triggered alerts. This was the time that one machine (MY.NET.210.114) sent over 354775 invalid packets to an external address. This is why it is one of the top machines to be investigated. Investigation of MY.NET.235.110 and MY.NET.203.98 are also recommended as there are some strange things happening involving those hosts (further information in section 3.5). Some other machines that should be scanned are listed in Appendix D, E and F. Those machines are possibly compromised by various exploits and need to be checked to see if they are infected. If they are compromised, then appropriate action needs to be taken.

Overall though, the SANS University network is in good shape. I will demonstrate that although there are some compromised machines, in general the

situation looks good. There is some work to be done with the detection rulebase itself, as the rules seem to trigger too easily.

3.2. Analyzed Files

The University IT Department provided me with some log files to analyze from Friday 2nd May 2003 to Tuesday 6th May 2003 consecutively. I was told that the logs were generated from an instance of Snort (available from http://www.snort.org) using a fairly standard ruleset. The logs were divided into three types of files – Alerts (showing events of interest), Scans (detected scans) and Out Of Spec (invalid flag and TCP option combinations).

Files Analyzed			
Alerts.030502	Alerts.030503	Alerts.030504	Alerts.030505
Alerts.030506	Scans.030502	Scans.030503	Scans.030504
Scans.030505	Scans.030506	OOS_Report_2003_05_02_28431.txt	
OOS_Report_2003_05_03_7239.txt		OOS_Report_2003_05_04_21395.txt	
OOS_Report_2003_05_05_25821.txt		OOS_Report_2003_05_06_7938.txt	

3.3. Analysis Method

The first step in analysis of the data supplied by the University was to decide on the categorisation. I made the decision that I would group the alerts by number for analysis, as criticality for some of the rules in the snort rule base were difficult to ascertain due to the many customised rules used. I also wanted to acknowledge that not all packets contained dangerous exploits but still needed to be investigated as high rates of noise can 'drown' out the real alerts. Often the firewall policies, host configurations, and the snort rulebase itself can be adjusted to lower the number of false alerts.

I noted that there was many incorrectly formatted log entries in the alert log files. I decided to remove all incorrectly formatted entries as I felt it would be unwise to base any decisions on data that I could not be assured was valid. This meant unless the data had a date, time, alert name, source and destination IP address and port number then it was ignored. This process was automated by the perl file I wrote called prepalerts.pl (based on csv.pl by Tod Beardsley). The program was written to take multiple alert files and to output them as one CSV file for importing into an SQL database. Two other programs (prespscans.pl and prepoos.pl) were created to do a similar job for the scan logs and the OOS logs as well (scripts are available in Appendix G). The information was then imported into Microsoft Access for analysis. Various SQL queries (over 30) were written to generate the information I needed. Some data was exported to Microsoft Excel for use in creating graphs. All generated information was combined and edited to produce what you are reading now.

3.4. Snort IDS Configuration

3.4.1. Sensor location

This snort sensor is located somewhere between the student access network and the main connection to the Internet, on the edge of the Universities network. This is was confirmed by correlating the alert logs with the scan logs and discovering the **real IP** addresses of the University network listed within. This discovery lent weight to the fact that only one alert packet was sent from an internal source to an internal

destination (0.0001% of the total alerts). The discovered student residents' network was a class B, and its assigned range of addresses was MY.NET.0.0 - MY.NET.255.255. It would be fair to say I was surprised to discover the 'real' IP address of this network, as published log files should never contain any real internal addresses – it is too much of a security risk. I would recommend the SANS University takes steps to rectify this.

The snort sensor also alerted on a large number of external to external traffic (22% of total). While some of these packets could really be from an internal address, but have a spoofed source IP address that happens to be external, it is very unlikely that this level would reach 22%. A more likely scenario is that the snort sensor is on a network segment that has 'External traffic' (non-MY.NET.x.x traffic) passing through it, and so is on the very edge of the monitored network.

3.4.2. Network Speed

After investigating the log files it seems that the snort sensor is in a high bandwidth location. The alert log files display some corruption. I believe this was caused by the flow of network traffic being higher than the IDS system could consistently record.

After some research using the real IP address of the MY.NET subnet, I found that the U.S. University was connected to a high-speed Internet backbone of some kind (possibly the Internet2 network). This was lent weight by the fact that many of the 'External' IP addresses were from U.S Universities (e.g. Kent State University, MIT, University of Vermont, UMBC etc.), and other large U.S. organisations (AT&T, US Postal Service, Packard Bell etc.).

3.4.3. Snort Rulebase

The Snort sensor seems to be running a version of Snort approximately v1.8. The configuration file used is based on the old standard rulebase from about the time that Snort v1.8 was released, but also has a few custom alerts added. This means that many rules are not listed in the Snort Rules Database [85], as they have been deprecated, or their alert name has changed.

3.5. Detected Alerts

The following table is based on the earlier work by Les Gordon in his GCIA practical [86]. It is an easy way to see the number of unique hosts (both source and destination) and direction of travel of the alerts found. The slight twist is that the unique source and destination columns contain two figures; the first number is the number of unique hosts, and the second item is how many packets those hosts sent. This allows a comparison to be made as to the ferocity of the attacks.

		# Unique Src > # Pkts		# Unique Dest > # Pkts		Traffic Flow			
Alert Name	# Alert	Int	Ext	Int	Ext	I->I	I->E	E->I	E->E
Incomplete Packet Fragments Discarded	355241	2>354841	107>400	71>401	31> 354840	1	354840	400	
TCP SRC and DST outside network	207965		198959> 207965		2228> 207965				207965
SMB Name Wildcard	204324		26337> 204324	41809> 204305	2>19			204305	19
High port 65535 tcp - possible Red Worm - traffic	35622	71>17203	93>18419	78>18419	103>17203		17203	18419	
High port 65535 udp - possible Red Worm - traffic	27195	81>15319	176>11876	117>11876	235>15319		15319	11876	

		# Unique Src > # Pkts		# Unique Dest > # Pkts		Traffic Flow			
Alert Name	# Alert	Int	Ext	Int	Ext	I->I	I->E	E->I	E->E
CS WEBSERVER – external web traffic	26830		6274> 26830	2>26826	1>4			26826	4
spp_http_decode: IIS Unicode attack detected	24362	558>23457	343>895	201>895	731>23467		23467	895	
Tiny Fragments - Possible Hostile Activity	17637	1>13259	25>4378	26>4376	1040> 13261		13259	4376	2
TFTP - Internal TCP connection to external tftp server	9545	13>5163	32>4382	11>4382	34>5163		5163	4382	
EXPLOIT x86 NOOP	6044		180>6044	158>6044				6044	
spp_http_decode: CGI Null Byte attack detected	5814	139>5675	55>139	5>139	147>5675		5675		
connect to 515 from outside	5039		4>5039	4878>5039				5039	
SUNRPC highport access!	2802		35>2802	32>2801	1>1			2801	1
Null scan!	2573		124>2573	119>2571	1>2			2571	2
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.	1823		82>1823	67>1823				1823	
Queso fingerprint	1717		345>1717	125>1716	1>1			1716	1
MY.NET.30.3 activity	1413		50>1413	1>1413				1413	
MY.NET.30.4 activity	1411		318>1411	1>1411				1411	
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	1253	12>1253			21>1253		1253		
IDS552/web-iis_IIS ISAPI Overflow ida nosize	887		537>887	687>887				887	
CS WEBSERVER – external ftp traffic	871		163>871	1>871				871	
TFTP - Internal UDP connection to external tftp server	549	31>487	15>62	21>62	37>487		487	62	
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	539	2>539			12>539		539		
Possible trojan server activity	398	26>127	53>271	185>271	33>127		127	271	
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	312		10>312	6>312				312	
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	203		11>203	12>203				203	
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	188	2>188			167>188		188		
NMAP TCP ping!	175		52>175	74>175				175	
IRC evil - running XDCC	165	14>165			14>165		165		
External RPC call	151		4>151	151>151				151	
EXPLOIT x86 setuid 0	150		135>150	123>150				150	
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	132		7>132	6>132				132	
SNMP public access	122		6>122	8>122				122	
EXPLOIT x86 setgid 0	61		58>61	56>61				61	
NIMDA - Attempt to execute cmd from campus host	60	2>60			58>60		60		
EXPLOIT x86 stealth noop	48		12>48	6>48				48	
TCP SMTP Source Port traffic	38		4>38	14>38				38	
Notify Brian B. 3.56 tcp	29		27> 29	1>29				29	
Notify Brian B. 3.54 tcp	27		23>27	1>27				27	
Back Orifice	26		2>26	26>26				26	
SMB C access	17		14>17	11>17				17	
Probable NMAP fingerprint attempt	12		8>12	9>12				12	
Attempted Sun RPC high port access	11		4>11	4>11				11	
RFB - Possible WinVNC - 010708-1	9	5>5	4>4	4>4	5>5		5	4	
TFTP - External UDP connection to internal tftp server	7	1>1	5>6	5>6	1>1		1	6	
FTP passwd attempt	7		3>7	2>7				7	
[UMBC NIDS IRC Alert] K\:line'd user detected possible trojan.	6		6>6	5>6				6	
DDOS shaft client to handler	4		2>4	1>4				4	

		# Unique Src > # Pkts		# Unique Dest > # Pkts		Traffic Flow			
Alert Name	# Alert	Int	Ext	Int	Ext	I->I	I->E	E->I	E->E
TFTP - External TCP connection to internal tftp server	3	1>1	2>2	2>2	1>1		1	2	
NIMDA - Attempt to execute root from campus host	3	1>3			3>3		3		
EXPLOIT x86 NOPS	2		1>2	1>2				2	
SYN-FIN scan!	2		2>2	2>2				2	
DDOS TFN Probe	1		1>1	1>1				1	
Site exec - Possible wu-ftpd exploit - GIAC000623	1		1>1	1>1				1	
[UMBC NIDS IRC Alert] Possible trojaned machine detected	1		1>1	1>1				1	
Bugbear@MM virus in SMTP	1		1>1	1>1				1	
TOTALS	943828	962> 437756	234713> 506072	49129> 298079	4907> 645749	1	437755	298078	207994

3.6. Alerts In Detail (triggered more than 5000 times)

Alert 1: Incomplete Packet Fragments Discarded | **Priority:** Medium | **Triggered:** 355241

Traffic Flow | **Int->Int**: 1 | **Int->Ext:** 354840 | **Ext->Int:** 400 | **Ext->Ext:** 0

Snort ID: Old | **Uniq Int Src:** 2 | **Uniq Ext Src:** 107 | **Uniq Int Dst:** 71 | **Uniq Ext Dst:** 31

Background: This is a built in error message generated by snorts deprecated 'D efrag' preprocessor. This preprocessor was dropped in favour of the 'Frag2' preprocessor in Snort v1.9 and upwards [87]. This alert is raised when an incomplete set of fragments are detected. Fragments can be lost in transit, corrupted or crafted so that they have overlapping boundaries, or non-consecutive boundaries to attempt to bypass the protection on the target computer or avoid the IDS system.

```
05/03-13:02:49.646122  [**] Incomplete Packet Fragments Discarded [**]
                       MY.NET.210.114:0 -> 213.97.198.23:0
05/03-12:26:14.532096  [**] Incomplete Packet Fragments Discarded [**]
                       MY.NET.210.114:0 -> 213.97.198.23:0
```

In the University log files we found there were only 2 internal sources of these incomplete packet fragments. The first IP address, MY.NET.210.114 sent only to 213.97.198.23 (213-97-198-23.uc.nombres.ttd.es). There were 354775 alerted packets sent between 11:45am 3rd May 2003 and 4:53pm 4th May 2003. After 4:53pm there were no further alerts produced by this host. On average MY.NET.210.114 sent almost 203 fragmented packets per minute. Checking the scan logs revealed that a large amount of UDP scans were sent from MY.NET.210.114 to 213.97.198.23 (64602 packets).

```
May  3 11:53:33 MY.NET.210.114:0  -> 213.97.198.23:0 UDP
May  3 11:53:30 MY.NET.210.114:2769  -> 213.97.198.23:64542 UDP
May  3 11:53:30 MY.NET.210.114:3092  -> 213.97.198.23:11661 UDP
May  3 11:53:30 MY.NET.210.114:3109  -> 213.97.198.23:48581 UDP
May  3 11:53:30 MY.NET.210.114:3124  -> 213.97.198.23:27101 UDP
```

The ports used by the UDP scans are a combination of null scans and various other source and destination UDP ports. This traffic seems to be produced by some sort of automated program, as the ports scanned seem random, but over time most ports seem to be scanned 5-13 times in a pseudo random fashion. I attempted to see if there was a control packet sent to the internal host but there was no such thing in the logs. The control packet may not have been recorded, or may not have been sent at all. I believe that the host MY.NET.210.114 has either been compromised and is being used in a DDoS attack on 213.97.198.23, or has a fault with it's network card or software (and was switched off at 4:53pm 4th May 2003). Either way this host deserves a visit to see if there is some attacking software present or a faulty network card.

The second host, MY.NET.203.98, sent 5 incomplete fragmented packets to the Null port on different addresses in the 192.168.x.x address ranges as shown below.

```
05/06-06:51:45.980065  [**] Incomplete Packet Fragments Discarded [**]
                       MY.NET.203.98:0 -> 192.168.1.102:0
05/06-07:00:33.581418  [**] Incomplete Packet Fragments Di scarded [**]
                       MY.NET.203.98:0 -> 192.168.1.2:0
05/06-12:28:31.664508  [**] Incomplete Packet Fragments Discarded [**]
```

```
                          MY.NET.203.98:0 -> 192.168.1.12:0
05/06-20:05:27.704195  [**] Incomplete Packet Fragments Discarded [**]
                          MY.NET.203.98:0 -> 192.168.1.102:0
05/06-22:41:37.744149  [**] Incomplete Packet Fragments Discarded [**]
                          MY.NET.203.98:0 -> 192.168.1.12:0
```

This was very suspicious. The 192.168.x.x block of addresses are reserved by the IANA [63] for use as private IP addresses and are supposed to be blocked from entering the internet by any border routers. As the snort instance does not seem to be behind a NAT device (as the NAT device would have converted the private IP address to a real IP address) I can only speculate that this address must be being generated by the MY.NET.203.98 host. A check for other communication with this host found that there were 5 external computers that had attempted to connect to port 137 (cannot tell if that was TCP or UDP). I could not tell if any of these connections had been successful with the level of data that I had available. Whitehats.ca [94] listed only one signature corresponding to port 137 - "NETBIOS-NAME-QUERY". It stated that "*NetBIOS name traffic is considered background noise on the network and should only be considered when combined with other forensic evidence that points to a problem/suspicion*". The strange fragmented packets to 192.168.x.x were unusual enough to qualify. Each of the port 137 connections were at least 2 hours earlier than the fragmented packets so I ruled out them being control connections for a 'bot' of some kind. The MY.NET.203.98 host was also SYN scanned by 4 different external computers for the ports 235, 445 and 80.

The rest of the traffic for other 71 external sources of 'Incomplete Packet Fragments' alerts come from varying locations, but none seem to specifically target a particular host. One external host, 64.12.56.35 (demand4-nm03.stream.aol.com), sent 42 incomplete fragment packets to MY.NET.224.138 in a four minute period. But this activity seems to be randomly spread across many different internal destinations from many different sources.

Correlations: Confirmation that the "Incomplete Packet Fragments Discarded" error message was part of the Defrag snort preprocessor was found during a discussion involving Martin Roesch on the snort users list [90]. The Whitehats.ca database [94] listed only one port 137 event which it said was used for NetBIOS name reconnaissance, but triggered often due to many processes using NetBIOS in their day-to-day operations.

Recommendations:

- Snort should be upgraded to v2.0.0 as the earlier versions contain a snort vulnerability [53]. The use of the 'Defrag' preprocessor has been deprecated. This would allow the newer 'Frag2' preprocessor to be used which is more memory efficient, and has added functionality to detect other evasion techniques like fragroute [87].
- Investigate MY.NET.210.114 and MY.NET.203.98 for signs of mis-use, or compromise. Also check for possible faulty network card or TCP/IP stack on MY.NET.210.114 if a compromise cannot be found as the number of incomplete packets generated was extremely high.
- Make sure that the border firewall can detect and drop invalid fragmented packets correctly.
- I would suggest installing another NIDS on the internal side of the firewall as well to make sure that there are no invalid fragments sneaking past unnoticed.

Alert 2: TCP SRC and DST outside network		**Priority:** Noise		**Triggered:** 207965
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 0	**Ext->Int:** 0	**Ext->Ext:** 207965
Snort ID: Cust.	**Uniq Int Src:** 0	**Uniq Ext Src:** 198959	**Uniq Int Dst:** 0	**Uniq Ext Dst:** 2228

Background: This custom rule seems to only generate an alert if the source and destination are both external IP addresses. It is difficult to understand why this rule is being used, as all it seems to do is generate a lot of alerts. A possible scenario explaining the use of this rule could be that this snort instance had in the past been protected from seeing traffic external to MY.NET, and this rule had been inserted to alert on possible internal IP address spoofing. In it's current location all it seems to do is to create noise.

Correlations: I was unable to find any correlations for this message. Most other GCIA practicals listed this alert in their tables but didn't discuss it any further.

Recommendations:

- Remove this rule from the rulebase.
- If this is not acceptable then tweak the rule slightly to specifically exclude the external networks close to MY.NET.

Alert 3: SMB Name Wildcard		**Priority:** Low		**Triggered:** 204324
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 0	**Ext->Int:** 204305	**Ext->Ext:** 19
Snort ID: Old	**Uniq Int Src:** 0	**Uniq Ext Src:** 26337	**Uniq Int Dst:** 91809	**Uniq Ext Dst:** 2

Background: The SMB Name Wildcard is used by Windows hosts when they are attempting to find the name of a server that they cannot find using DNS. It is very common in a Windows network. The Whitehats.ca port database [94] mentions that external probes on this port can often be reconnaissance probes used to discover hosts in the internal network. They recommend blocking UDP port 137 at the boundary router.

Correlations: Judy Novak sent a good analysis of this rule to Stephen Northcutt [91]. It explains the detect in a good level of detail. J Sage [92] provides an overview a few other explanations of NetBIOS name table probes. ArachNIDS [89] also provide signatures to detect this kind of probe using snort.

Recommendations:

- Unless specifically needed external NetBIOS name service traffic should be blocked at the boundary router.

Alert 4: High port 65535 tcp - possible Red Worm - traffic		**Priority:** Low		**Triggered:** 35622
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 17203	**Ext->Int:** 18419	**Ext->Ext:** 19
Snort ID: Old	**Uniq Int Src:** 71	**Uniq Ext Src:** 93	**Uniq Int Dst:** 78	**Uniq Ext Dst:** 103

Background: This custom alert is designed to trigger when a TCP packet is sent to port 65535. This is suggested by the alert rule title to be a possible Adore/Red Worm which is designed to install a backdoor on a Linux machine [93]. A query on the Whitehats.ca Port Database [94] reveals that the Adore/Red Worm can use TCP for transmission as well as UDP. Investigation of the transmissions between MY.NET.207.150 (a large alerter) and external sources I found this sequence:

```
05/03-05:03:49.150400  [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**]
                       218.78.248.145:1357 -> MY.NET.207.150:80
05/03-20:14:28.629061  [**] SMB Name Wildcard [**] 4.46.132.229:1027  ->
                       MY.NET.207.150:137
05/04-11:42:49.750816  [**] spp_http_decode: IIS Unicode attack detected [**]
                       69.0.94.145:1644 -> MY.NET.207.150:80
05/04-14:32:24.841689  [**] SMB Name Wildcard [**] 81.82.104.163:1029  ->
                       MY.NET.207.150:137
05/05-16:32:38.030434  [**] High port 65535 tcp - possible Red Worm - traffic
                       [**] MY.NET.207.150:3879 -> 65.70.111.122:65535
05/05-16:40:36.136217  [**] High port 65535 tcp - possible Red Worm - traffic
[**] 65.70.111.122:65535 -> MY.NET.207.150:4014
05/05-16:40:40.869697  [**] High port 65535 tcp - possible Red Worm - traffic
[**] 65.70.111.122:65535 -> MY.NET.207.150:4014
```

As you can see, the initial packet is a '.ida' buffer overflow attempt [95] similar to the type used by CodeRed and Nimda, and is then followed by an IIS Unicode attack. These are then followed by the high port TCP access. At first glance it appears one of the first four packets could be a control packet which caused the high port TCP access but I don't believe this to be the case. Firstly there is too much time that has elapsed between the receipt of the first packets and the high port TCP access. Secondly the source IP addresses of the first few packets are completely unrelated to the destination IP addresses in the high port TCP access packets. This makes the spoofing of the three way handshake and TCP data stream very difficult indeed. I believe that the above packet trace shows the two attempted exploits sent to MY.NET.207.150 are unrelated to the High Port false alerts that follow them.

Another possibility is that the use of the high ports are a natural occurrence caused by the wrapping around of ephemeral ports due to a high number of connections. I doubt this is the case as in the trace above there is only about 4.5-14 seconds between the uses of port 65535. This is too short a duration for the wrap around to occur.

It could be that the repeated use of port 65535 is actually one retried connection. This I also doubt as the duration between the retries is not standard or exponential, and there are a lot more than 3 attempts at it (over in the full logs).

A more likely scenario is that this access is just a normal connection that happens to use TCP port 65535. Many of the ports used from 65535 accessed connections in the range of 4200-4299. The IANA lists this range as being used for VRML Multi User Systems. These are 3D virtual worlds across the network. A check of Google didn't find port 65535 mentioned in relation to VRML. I did find some reported activity

```
05/06-03:46:49.705422  [**] High port 65535 tcp  - possible Red Worm  - traffic
                       [**] MY.NET.202.206:2327  -> 12.231.84.153:65535
05/06-03:46:50.218380  [**] High port 65535 tcp  - possible Red Worm  - traffic
                       [**] 12.231.84.153:65535  -> MY.NET.202.206:2327
05/06-03:46:50.221084  [**] High port 65535 tcp  - possible Red Worm  - traffic
                       [**] MY.NET.202.206:2327  -> 12.231.84.153:65535
05/06-03:46:53.254034  [**] High port 65535 tcp  - possible Red Worm  - traffic
                       [**] 12.231.84.153:65535  -> MY.NET.202.206:2327
```

There were some packets found that had an initial source port of 2327 (listed above). A lookup on IANA had this port indicated as being used by XingCSM. Google found a page originall y written by Xing Technologies [96] discussing their StreamWorks Streaming Media Server. This has since been discontinued by Xing Technologies (they have been bought by Real Player) but it could still be in use on the network.

It is, of course, also possib le (as GlockSoft [97] and others show) that if the target machine is a Windows machine it could be the RC1 Trojan. Without further information we just don't know.

Correlations: Les Gordon [86] discussed how Web browsing and KaZaA seem to make up most of his false alerts for this rule.

Recommendations:

- If the purpose of this rule is to detect the Adore/Red Worm then it should be more specific and possibly do some extra content matching.
- If the purpose is to catch all port 65535 traffic the rule should b e removed as the high rate of false positives is restricting it's usefulness.

Alert 5: High port 65535 udp - possible Red Worm - traffic			**Priority:** Low	**Triggered:** 27195
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 15319	**Ext->Int:** 11876	**Ext->Ext:** 0
Snort ID: None	**Uniq Int Src:** 81	**Uniq Ext Src:** 176	**Uniq Int Dst:** 117	**Uniq Ext Dst:** 235

Background: This alert is very similar to the "High port 65535 udp - possible Red Worm – traffic" rule above. The Adobe/Red Worm is more commonly associated with the port 69/udp than p ort 69/tcp. Most of this traffic seems to be completely innocuous. But without more information, or a more specific rule, it is impossible to separate false alert from real positive.

Correlations: Les Gordon [86] discussed how this rule's fal se alerts seemed to be mainly aimed at his detected AFS servers. After some investigatio n [98] I found that AFS servers li sten on ports 7000-7032 and 2106. The only connections listed in the alert logs that delivered to the AFS ports were by IRC bots. I believe this d iscrepancy can be explained by assuming that the snort instance has been moved further to the edge of the MY.NET network, if not outside of it. In all previous practicals I have read there has been far less external to external and more internal to interna l traffic. The change in both indicates the snort instance is likely to have been moved, and explains why Les Gordon's detected AFS Servers were not found by me.

Recommendations:

- If possible the rule should be turned off, but if it is needed to detect th e Adore Worm, it should be tuned to reduce the noise it generates.

Alert 6: CS WEBSERVER - external web traffic			**Priority:** Low	**Triggered:** 26830
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 0	**Ext->Int:** 25826	**Ext->Ext:** 4
Snort ID: Cust.	**Uniq Int Src:** 0	**Uniq Ext Src:** 6274	**Uniq Int Dst:** 1	**Uniq Ext Dst:** 1

Background: This is an alert generated by a custom alert. This rule is designed to record each access to the Universities Department of Computer Science and Electrical Engineering Webserver. An interesting feature o f the logs is the trace shown below:

```
05/04-11:00:43.069382  [**] CS WEBSERVER  - external web traffic [**]
                       216.39.48.127:47244  -> MY.NET.100.165:80
05/04-11:00:43.448596  [**] CS WEBSERVER  - external web traffic [**]
                       216.39.48.127:47248  -> MY.NET.100.165:8 0
05/04-11:00:43.739398  [**] CS WEBSERVER  - external web traffic [**]
                       216.39.48.127:47250  -> MY.NET.100.165:80
```

As you can see there are a lot of connections from 216.39.48.127 to the webserver. MY.NET.100.165. The host 216.39.48.127 has a DNS name of bui ldrack52.sv.av.com (part of Altavista. com). I believe that this sequence of packets is the result of the Altavista Search Engines Web Spider trawling the Internet for web pages.

As this is a public facing webserver we would expect the number of times this rule triggered to be very high. This rule seems to record every connection to the website. This information would only be useful if it was used as a metric for the number of web visitors. It is possible that this rule was put in as a temporary measure to monitor suspicious activity to the CS Web Server and in this context it is fine. But for normal everyday use it would be better to put in narrower more focused rules targeting specific exploits.

Correlations: I could not find any correlations for this rule , as all of the GCIA practicals I looked at did not view the CS Webserver alert rule in any detail.

Recommendations:

- This rule should be removed as it appears to do nothing more than take web server statistics.
- If statistics are required then a web server log parser (such as Websense Log Analyser) should be purchased to continue this monitoring.

Alert 7: spp_http_decode: IIS Unicode attack detected		**Priority:** High		**Triggered:** 24362
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 23467	**Ext->Int:** 895	**Ext->Ext:** 0
Snort ID: None	**Uniq Int Src:** 558	**Uniq Ext Src:** 343	**Uniq Int Dst:** 201	**Uniq Ext Dst:** 731

Background: This spp_http_decode snort preprocessor generated alert tells of the use of Unicode encoded ".", "/" and "\" characters in packets sent to various common HTTP ports. Unpatched versions of Microsoft IIS are vulnerable to Unicode -encoded URL and directory traversal attacks [99][100][101] from various worms, such as CodeRed II [102], Nimda [103] and Sadmind/IIS [104]. This allows the attacker to gain access to various files via directory traversal, and to perform various commands on the web server through IIS.

This alert can generate some false positives as it alerts on all Unicode encoded traffic. This means it could alert on foreign language sites using Korean, Chinese and other asian language character sets, or on SSL traffic. I believe that the use of this rule for monitoring internal sources of this IIS Unicode alert is entirely justified as it only seems to alert on port 80 traffic (SSL traffic normally uses port 443). The fact that this is a U.S. University also lends this weight, as there should be a small percentage of websites that use a foreign character set. This means that an internal source that triggers this alert is highly likely to have been compromised.

```
05/04-16:44:02.681153  [**] spp_http_decode: IIS Unicode attack detected [**]
                       61.179.12.120:22231 -> MY.NET.201.218:80
05/04-16:44:10.068342  [**] spp_http_decode: IIS Unicode attack detected [**]
                       218.91.41.22:1233 -> MY.NET.253.5:80
05/04-17:12:06.766615  [**] spp_http_decode: IIS Unicode attack detected [**]
                       MY.NET.144.51:3311 -> 202.106.182.195:80
```

The Universities MY.NET has 558 unique sources of this IIS Unicode attack. An excerpt of the traffic logged is shown above. They are listed below in Appendix D. Each of the machines listed in Appendix D should be scanned for signs of an exploit, as they are very likely infected computers actively seeking new hosts to infect.

Correlations: A good overview of the IIS Unicode attack was written by Les Gordon [86]. He discusses why this alert is prone to false positives. A differing view is supported by Tod Beardsley [105]. Cert.Org provided a description of the IIS Vulnerability in [100]. An excellent explanation of the Sadmin/IIS vulnerability is the GCIA practical written by Ben Wilson. [106]. James Crossman [107] discusses how this alert corresponds with the Sadmind/IIS Worm.

Recommendations:

- Utilise ingress and egress control by configuring the firewall and boundary routers to mitigate the threat. Cisco have a good whitepaper on defense against Nimda [108]
- Investigate the machine listed in Appendix D for signs of a compromise.
- Make sure all IIS web servers are patched using the latest cumulative patch from Microsoft [99] so that they are not exploited.
- If there is no boundary virus checking of email attachments then implement an email content virus checker as a matter of urgency.
- Ensure the machines all have a virus checker installed with up-to-date virus definition files. This can be accomplished using some centrally managed enterprise level anti-virus software.
- Review the Universities Network Acceptable Use Policy and make sure it states that computers will only be allowed to be connected to the universities network only if they have a virus checker installed with up-to-date virus definition files.

Alert 8: Tiny Fragments - Possible Hostile Activity			**Priority:** Medium	**Triggered:** 17637
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 13259	**Ext->Int:** 4376	**Ext->Ext:** 2
Snort ID: None	**Uniq Int Src:** 1	**Uniq Ext Src:** 25	**Uniq Int Dst:** 26	**Uniq Ext Dst:** 1040

Background: Tiny Fragments are often used in attempts to evade detection by a NIDS. By making the fragments smaller than the encapsulated packet's header length, the attacker hopes the NIDS will not be able to interpret the packets and will drop them. The hope is also that the target host will not drop the packets, but will reassemble the packets, thereby effecting a successful evasion.

This alert is generated when the Minifrag snort pre-processor detects a fragment which is smaller that the threshold configured in the snort configuration file. The Snort users manual for v1.9 states *"Generally speaking, there is no piece of commercial networking equipment that fragments in sizes smaller than 512 bytes…".* So by specifying the upper limit near to 512 bytes, we would be able to catch most of the traffic designed to evade detection by using tiny fragments. Unfortunately we are not made aware of the threshold limit in this instance.

The generation of tiny fragments can be attributed to a faulty network card or network driver, but more often than not it is due to an attempted evasion attack. These attacks are mainly designed to slow down the destination network – essentially a Denial of Service (DoS).

There was only one internal host that sent tiny fragments. MY.NET.235.110 sent over 13266 tiny fragments during the monitored period. The sending rate of MY.NET.235.110 never went above about 3 packets per second, so this was definitely not a DoS attack in itself. There is however a real possibility that this computer was taking part in a Distributed Denial of Service attack.

```
05/02-00:30:02.184659  [**] Tiny Fragments  - Possible Hostile Activity [**]
                       MY.NET.235.110 -> 200.77.81.95
05/02-00:30:02.271481  [**] Tiny Fragments  - Possible Hostile Activity [**]
                       MY.NET.235.110 -> 200.77.81.95
05/02-00:30:03.605148  [**] Tiny Fragments  - Possible Hostile Activity [**]
                       MY.NET.235.110 -> 200.77.81.95
```

I investigated this by listing all traffic to and from MY.NET.235.110. Some interesting scans came to my attention:

```
May  2 00:11:15 130.85.235.110:0  -> 200.77.81.95:0 UNKNOWN 1*UA***F RESERVEDBITS
May  2 00:11:15 130.85.235.110:18503 -> 200.77.81.95:1208 VECNA *2**P**F
                                                    RESERVEDBITS
May  2 00:11:17 130.85.235.110:0  -> 200.77.81.95:0 NOACK **U**RSF
May  2 00:11:20 130.85.235.110:0  -> 63.227.65.64:0 INVALIDACK 1*UAP*SF
                                                    RESERVEDBITS
May  2 00:11:24 130.85.235.110:0  -> 200.77.81.95:0 INVALIDACK ***APRS*
```

This set of scans is unusual. Either there is some scripted scans and attacks being used from MY.NET.235.110, or the network card is sending out corrupted packets. Although there seems to be a pseudo randomness of the TCP ports and flags in the scans, the overall repeated numbers listed leads me to believe this host is likely infected with a bot or backdoor, and needs to be investigated as a matter of urgency.

Correlations: An overview of what this alert does was provided in Martin Roesch's explanation on the Snort users mailing list [109]. Mark Enbrich also researched this alert rule and came to a similar conclusion, but his Internal Source IP was different [110].

Recommendations:

- Visit MY.NET.235.110 and scan it for signs of compromise.

Alert 9: TFTP - Internal TCP connection to external tftp server			**Priority:** Noise	**Triggered:** 9545
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 5163	**Ext->Int:** 4382	**Ext->Ext:** 0
Snort ID: Cust.	**Uniq Int Src:** 13	**Uniq Ext Src:** 32	**Uniq Int Dst:** 11	**Uniq Ext Dst:** 34

Background: This alert appears to be generated when an internal machine sends TCP packets to an external destination using a destination port of 69, or when an external machine replies back using the same connection.

Port 69 is used by Trivial FTP – a type of file transfer protocol which doesn't worry about authentication and security, but which is just concerned with getting the file from point A to point B. I knew that port 69 was also the port used by the Nimda virus [111], but after some research it seemed that only 69/udp was used by Nimda – not 69/tcp. Just to check if there was a different variant of

Nimda that I was unaware of I checked to see if any of the hosts that used port 69/udp had used any of the other common Nimda ports (25, 69, 80, 137 -139,445). I found something. MY.NET.189.41 was sending the following traffic:

```
05/04-14:39:53.921262  [**] TFTP - Internal TCP connection to external tftp
                       server [**] 160.75.92.13:69 -> MY.NET.189.41:3585
05/04-14:39:53.921315  [**] TFTP - Internal TCP connection to external tftp
                       server [**] MY.NET.189.41:3585 -> 160.75.92.13:69
05/04-14:48:08.992616  [**] TFTP - Internal UDP connection to external tftp
                       server [**] MY.NET.189.41:1331 -> 217.234.140.155:69
05/04-14:52:19.746897  [**] spp_http_decode: IIS Unicode attack detected [**]
                       MY.NET.189.41:2793 -> 217.187.78.118:80
05/04-14:54:54.657150  [**] TFTP - Internal TCP connection to external tftp
                       server [**] MY.NET.189.41:4228 -> 160.75.92.13:69
```

MY.NET.189.41 has alerts for both TCP and UDP connections on port 69, one of which is an IIS Unicode alert. The IIS alert could just be a coincidence, as 217.187.78.118 belongs to a German company and it is attempting connection to a web port that could be running Unicode characters. But I believe it is more likely to indicate that the client is infected with Nimda, as there was no webserver available on 217.187.78.118:80. In any case MY.NET.189.41 needs further investigation. The other port 69/tcp connections seem to be valid TFTP sessions.

Correlations: I could not find any correlations for this rule, as all of the GCIA practicals I looked at did not view the "TFTP - Internal TCP connection to external tftp server" alert rule in any detail.

Recommendations:

- Block port 69 for TCP and UDP access for both ingress and egress.
- Investigate MY.NET.189.41 for signs of a compromise.
- Make sure all IIS web servers are patched using the latest cumulative patch from Microsoft [99] so that they are not exploited.
- If there is no boundary virus checking of email attachments then implement an email content virus checker as a matter of urgency.
- Ensure the machines all have a virus checker installed with up-to-date virus definition files. This can be accomplished using some centrally managed enterprise level anti-virus software.
- Review the Universities Network Acceptable Use Policy and make sure it states that computers will only be allowed to be connected to the universities network only if they have a virus checker installed with up-to-date virus definition files.

Alert 10: EXPLOIT x86 NOOP			**Priority:** Noise	**Triggered:** 6044
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 0	**Ext->Int:** 6044	**Ext->Ext:** 0
Snort ID: Old	**Uniq Int Src:** 0	**Uniq Ext Src:** 180	**Uniq Int Dst:** 158	**Uniq Ext Dst:** 0

Background: This rule is triggered when snort detects a sequence of hex-codes which match assembler no-operation commands [112]. These noop's are often used in buffer overrun exploits. They allow the exploit coder to be less specific as to where their exploit code should jump to, and start processing. Robert Graham explains this quite well in his post on the securityfocus-ids mailing list [77]. An important point to note is that this rule can have an extremely high rate of false positives. Many image files just happen to contain the right sort of hexcodes in them to trigger this rule. And that means often they will trigger when users are accessing websites, or just transferring files between computers.

A possible better noop detector would be the spp_fnord snort preprocessor by Dragos Ruiu [113]. This pre-processor is supposed to have fewer false positives and a better detection rate of new noop sleds than the equivalent snort rules [114]. Although it was due for inclusion into Snort v1.9, I could find no mention if this was indeed included on Snort's website.

There are no internal sources of this alert. The latest version of Snort by default does not look for shellcode on port 80, but it seems the version used by the University looks for x86 noop exploits on all ports as the following non-related packet traces show:

```
05/04-02:48:52.077465  [**] EXPLOIT x86 NOOP [**] 142.169.60.119:27005  ->
                       MY.NET.197.6:27015
05/04-10:46:26.034652  [**] EXPLOIT x86 NOOP [**] 12.16.131.99:4944  ->
                       MY.NET.5.45:80
05/04-10:46:26.058680  [**] EXPLOIT x86 NOOP [**] 12.16.131.99:4944  ->
                       MY.NET.5.45:80
05/04-19:04:54.890415  [**] EXPLOIT x86 NOOP [**] 207.44.194.27:80  ->
```

```
                          MY.NET.225.206:3149
05/05-16:46:14.608603  [**] EXPLOIT x86 NOOP [**] 213.140.8.171:36001  ->
                          MY.NET.190.93:139
```

I noticed that there were some strange ports being used in some of the packets that this alert detected. Port 27015 is a common port used for Sierra Online/Valve Games like Halflife, or CounterStrike [115]. MY.NET.197.6 looks as though it's being used to host games. Something that needs addressing rather quickly.

```
05/04-14:37:40.457765  [**] EXPLOIT x86 NOOP [**] 200.218.177.59:54775  ->
                          MY.NET.197.6:27015
```

Unfortunately any other information is hidden deep within the false alerts. There really needs to be some snort rule changes to cut down on false triggering for there to be any use for this rule.

Correlations: HD Moore [116] does give a similar explanation to Robert Graham [76] in his post to the Snort users list. He mentions that an FTP transfer is false alerting due to the traffic content sent.

Recommendations:

- Replace this rule with the spp_fnord snort preprocessor to reduce the number of false alerts
- Upgrade Snort to a later version, or install a newer snort rulebase
- Do not monitor for shellcode on port 80 traffic to reduce the performance drain on the snort instance.
- Investigate the use of MY.NET.197.6 as a online games hosting server.

Alert 11: spp_http_decode: CGI Null Byte attack detected			**Priority:** Low	**Triggered:** 5814
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 5675	**Ext->Int:** 0	**Ext->Ext:** 0
Snort ID: None	**Uniq Int Src:** 139	**Uniq Ext Src:** 55	**Uniq Int Dst:** 5	**Uniq Ext Dst:** 147

Background: This alert is generated by the spp_http_decode Snort preprocessor. It is triggered when the http preprocessor encounters a %00 in the packet payload [117]. The null byte is often used to shorten a URL, to bypass the CGI security mechanisms and gain access to files in the operating systems file system [118]. It does have a tendency to false positive on SSL encrypted traffic and content that includes cookies.

```
05/05-22:50:37.484156  [**] spp_http_decode: CGI Null Byte attack detected [**]
                          MY.NET.98.15:3111  -> 80.247.32.141:80
05/05-22:13:20.643638  [**] spp_http _decode: CGI Null Byte attack detected [**]
                          MY.NET.98.96:49251  -> 207.171.183.16:80
05/05-22:50:37.484156  [**] spp_http_decode: CGI Null Byte attack detected [**]
                          MY.NET.98.15:3111  -> 80.247.32.141:80
```

As expected, most of the alerted traffic is on egress traffic with a destination port 80. Some may be true CGI null byte exploit traffic, others could be cookie traffic but it is difficult to know. There are some other interesting ports

```
05/04-17:05:10.370320  [**] spp_http_decode: CGI Null Byte attack detec ted [**]
                          MY.NET.234.154:1848  -> 207.44.232.38:8080
```

One unusual style of CGI null byte connection stood out. The packet shown above is one of about from 144 logged. A lookup of Dshield.Org found that this resolved to moya.scarywater.net. A check of Google gave me the answer I needed. Moya.scarywater.net is a BitTorrent download site [119]. BitTorrent is a peer to peer downloading client [120] that uses a tit-for-tat ration style downloading algorithm. As it uses 'unused' bandwidth for it's downloading and uploading I expect it is something that the University would like to eradicate from its network.

Overall this rule is generating too many alerts for us to be able to investigate them properly. It should be tuned to reduce this number to a manageable level.

Correlations: I could not find any extra correlations for this rule other that those discussed in the background information above.

Recommendations:

- Change the CGI Null Byte rule to only alert on incoming traffic to the webservers on the common web ports.
- If internal sources of CGI null byte traffic needs to be monitored then put in another rule which check only egress traffic on common web ports.

- Visit the computer MY.NET.234.154 and investigate for any Peer-to-Peer software. Remove the software if found and instruct the user as to the dangers of its use (possible virus infection source).

Alert 12: connect to 515 from outside		**Priority:** Medium		**Triggered:** 5039
Traffic Flow	**Int->Int**: 0	**Int->Ext:** 0	**Ext->Int:** 5039	**Ext->Ext:** 0
Snort ID: Cust.	**Uniq Int Src:** 0	**Uniq Ext Src:** 4	**Uniq Int Dst:** 4878	**Uniq Ext Dst:** 0

Background: This alert triggers when an external source attempts to connect to an internal machine using port 515 – a port commonly used by the UNIX line printer daemon (LPR). Both HP-UX [121] and Solaris [122] have buffer overflow vulnerabilities and these connections could be exploits of those. It is unlikely that there is a need for external access to LPR printing, so port 515 should be blocked by the boundary routers.

The alert traffic that was logged showed that only 4 external sources attempted to connect to an internal 515 Line Printer Daemon (LPD). 128.46.117.76 (civl1240pc2.ecn.purdue.edu), 141.157.67.253 (pool-141-157-67-253.balt.east.verizon.net), 68.49.94.97 (pcp02267324pcs.longhl01.md.comcast.net) and 152.1.193.6 (chjlpc4.chem.ncsu.edu) were the external sources that caused these alerts to be triggered. civl1240pc2.ecn.purdue.edu was especially vociferous, sending 4873 packets in 3 minutes an 13 seconds.

There are no indications that any of the internal hosts were compromised in any of these attacks

Correlations: Jasmir Beciragic [123] discusses a vulnerability CVE-2000-0917 in LPRng 3.6.24 which allows remote attackers to execute arbitrary commands on the exploited computer.

Recommendations:

- Block port 515 at the boundary routers
- Keep all servers patched with the latest patches. Remove all unnecessary services from the servers as well. This minimizes the ways attackers have of compromising systems.

3.7. Other Significant Finds

Any unusual or important information found during alert research is listed below:

- MY.NET.97.181 and MY.NET.97.48 are both infected with a Nimda like virus that is looking for external sites to infect. Both alerted on “IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize” [124], “NIMDA - Attempt to execute cmd from campus host” [125] [126], and “NIMDA - Attempt to execute root from campus host” [127][128]. MY.NET.97.181 SYN-scanned over 10636 different addresses looking for an open port 80 to infect. Both machines need to be checked urgently for viruses and disinfected.

MY.NET.97.181 Virus Traffic

```
05/03-22:59:13.326341  [**] IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL
                       nosize [**] MY.NET.97.181:1618  -> 130.149.203.234:80
05/03-22:59:24.260939  [**] NIMDA - Attempt to execute cmd from campus host
                       [**] MY.NET.97.181:1618  -> 130.149.203.234:80
```

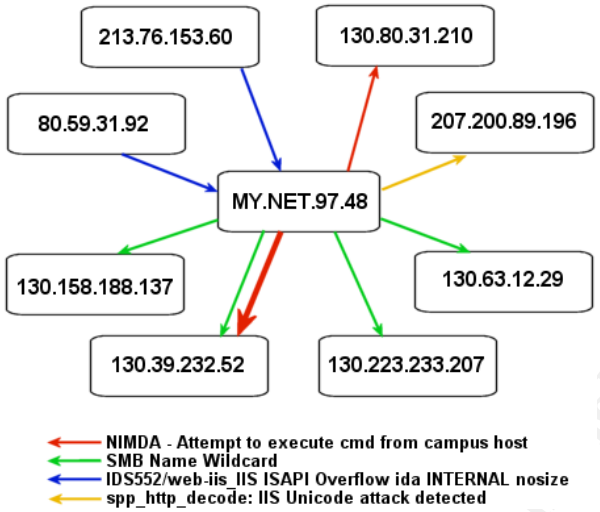


Figure 2 - Link Diagram showing Traffic Flow from MY.NET.97.48

- There seem to be many peer-to-peer file-sharing programs in use on the Universities network. Many seem to have been picked up by alerts with high false alert ratings – such as "High port 65535 udp - possible Red Worm - traffic " and "Queso fingerprint ". I detected about 490 possible KaZaA , 161 possible Gnutella, 1554 possible WinMX and 8 possible Peer-to-peer file-sharing connections from the log files (67 distinct internal sources listed in Appendix E). Peer-to-peer (P2P) file-sharing programs are a well-known file infection 'backdoor' as they often bypass the main http proxy and anti-virus protection systems.

Possible KaZaA Traffic

```
05/02-07:49:19.670538  [**] High port 65535 tcp  - possible Red Worm -
                       traffic [**] 4.46.158.139:65535  -> MY.NET.207.34:1214
05/02-07:49:19.672953  [**] High port 65535 tcp  - possible Red Worm -
                       traffic [**] MY.NET.2 07.34:1214  -> 4.46.158.139:65535
05/02-07:49:19.675526  [**] High port 65535 tcp  - possible Red Worm -
                       traffic [**] 4.46.158.139:65535  -> MY.NET.207.34:1214
```

While it is possible that this could be a Adore/Red virus, the sheer number of consistent connections in both directions makes it more likely that these are actually packets from communication between KaZaA P2P clients.

Possible Gnutella Traffic

```
05/06-10:56:00.814179  [**] Queso fingerprint [**] 130.136.4.226:3369  ->
                       MY.NET.217.54:6346
05/06-10:56:46.877562  [**] Queso fingerprint [**] 130.136.4.226:3637  ->
                       MY.NET.217.54:6346
```

At first glance this looks like the it could be a Queso fingerprint attempt, but I believe it's alerted because the traffic has come from behind an ECN aware router. This Queso fingerprint rule (which seems to have been removed from the current default snort signatures file) alerts on both the Reserved flags and the SYN flag set [129]. Toby Miller showed the difficulty that ECN would cause for Queso and Nmap traffic detection in his paper "ECN and it's impact on Intrusion Detection" back in January 2001 [130]. I believe these packets are actually part of a Gnutella P2P connection. There may be some actual Queso and Nmap scanning attempts in amongst these false alerts but they are difficult to find with the present rulebase.

I would recommend adding specific rules to the snort signatures to detect the peer-to-peer clients more accurately. Snort v2.0.0 has some P2P detection routines available for use in its default configuration. If the University does not want to allow P2P at all, then the commonly used P2P ports (1214, 6346, 6257, 6699, 5555, 6666 and 7777) need to be blocked (Note port 6666 is sometimes used for IRC traffic). Another option is to force all users to go through a proxy server that has P2P proxy controlling capabilities. It would be prudent to remind all users at this point that if they are found to have illegal software and files on their machines then disciplinary measures could be taken against them.

- There are a few IRC users on the campus who seem to be infected with the XDCC bot or the sdbot. The XDCC bot allows file-sharing through IRC chat channels. TonkinGin has an excellent paper on the perils of XDCC available [131]. The sdbot is a DDoS flooding tool controlled from IRC. Cert have released an advisory about the GT-bot and sdbot [132]. They have received reports of sdbot networks as large as 7000 systems. It is important to visit the machines listed in Appendix F and to investigate them for compromise.
- MY.NET.150.86 may have been remote controlled by 68.55.34.178 (pcp255450pcs.howard01.md.comcast.net) using WinVNC. There was an alert that showed that MY.NET.150.86 responded to 68.55.34.178 on a port commonly used for WinVNC remote control. The machine needs to be checked that it does have WinVNC, as it could be a security risk if the user is not authorised, or if the data is being transported across the Internet unencrypted. There needs to be a policy on remote desktop access. If external access needs to be had, then the traffic should be encrypted to secure it.

3.8. Detected Scans

As mentioned earlier the scan logs contained the 'real' IP addresses of the University. This had to be rectified before the log files could be used for input. The prepscans.pl perl script (see Appendix G) was created to obfuscate the 'real' IP address and allow the analysis to continue. The results are listed below.

		Uniq Src > # Pkts		Uniq Dest > # Pkts		Traffic Flow			
Scan Type	# Scans	Int	Ext	Int	Ext	I->I	I->E	E->I	E->E
UDP	616764	407509 > 615538	1060 > 1226	1060 > 1226	407509 > 615538	0	615538	1226	0
SYN	352647	66727 > 70517	267051 > 282130	267051 > 282130	66727 > 70517	0	70517	282130	0
NULL	3059	448> 1162	343 > 1897	343 > 1897	448 > 1162	0	1162	1897	0
FIN	1818	571 > 571	1245 > 1247	1245 > 1247	571 > 571	0	571	1247	0
NOACK	1045	365 > 574	295 > 471	471	365 > 574	0	574	471	0
INVALIDACK	581	160 > 199	341 > 382	341 > 382	160 > 199	0	199	382	0
VECNA	435	97 > 136	174 > 299	174 > 299	97 > 136	0	136	299	0
UNKNOWN	262	71 > 74	183 > 188	183 > 188	71 > 74	0	74	188	0
XMAS	88	47 > 57	26 > 31	26 > 31	47 > 57	0	57	31	0
NMAPID	51	21 > 32	16 > 19	16 > 19	21 > 32	0	32	19	0
FULLXMAS	43	21 > 22	17 > 21	17 > 21	21 > 22	0	22	21	0
SPAU	16	8 > 9	3 > 7	3 > 7	8 > 9	0	9	7	0
SYNFIN	10	6 > 7	4 > 4	4 > 4	6 > 7	0	6	4	0
TOTAL	976819	476051 > 688897	270758 > 287922	270758 > 287922	476051 > 688897	0	688897	287922	0

3.9. Out-of-Spec (OOS) Discussion

These files contained output from the snort instance which was out-of-spec – i.e. it didn't meet the required standard for a valid packet structure. By far the most common packet type was the ECN-setup SYN packet (91.1% of OOS)

```
05/03-00:05:24.337895 81.218.96.254:47004  -> MY.NET.227.150:1214
TCP TTL:49 TOS:0x0 ID:12710 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xAF68B6A9  Ack:  0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 327190287 0 NOP WS: 0
```

This packet is defined in RFC3168 [133] as the initial connection packet of an ECN aware machine. These can safely be ignored in most instances. But could this packet be anything else? Queso and Nmap scans often set the two 'Reserved' bits to see how the OS will react, which allows them to detect what OS is replying to them. ECN traffic uses those same two bits, which limits the effectiveness of using the reserved bits as a Queso or Nmap indicator. Toby Miller explains this well in his paper "ECN and it's impact on Intrusion Detection" [130].
The next most often detected OOS packet (5.4%) had only the Push flags set. Push should never be set by itself, but is always used in conjunction with an Ack flag. The culprit seems to be KaZaA. If you look through the ASCII conversion of the hex dump below you will see the typical KaZaA content header "UserAgent: KazaaClient". It looks like the KaZaA client built on November 3 2002 has a coding error built into it.

```
5/03-07:59:44.086128 148.63.130.64:1183  -> MY.NET.218.62:2373
TCP TTL:109 TOS:0x0 ID:1591 IpLen:20 DgmLen:443 DF
****P*** Seq: 0x8185B00A  Ack: 0x0  Win: 0x2000  TcpLen: 20
47 45 54 20 2F 2E 68 61 73 68 3D 38 64 63  38 62  GET /.hash=8dc8b
66 37 36 35 35 38 64 38 32 62 37 33 64 62 37 35  f76558d82b73db75
35 31 31 35 39 62 61 35 65 30 66 31 33 63 32 39  51159ba5e0f13c29
38 61 31 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F  8a1 HTTP/1.1..Ho
73 74 3A 20 31 33 30 2E 38 35 2E  32 31 38 2E 36  st: MY.NET.218.6
32 3A 32 33 37 33 0D 0A 55 73 65 72 41 67 65 6E  2:2373..UserAgen
74 3A 20 4B 61 7A 61 61 43 6C 69 65 6E 74 20 4E  t: KazaaClient N
6F 76 20 20 33 20 32 30 30 32 20 32 30 3A 32 39  ov  3 2002 20:29
```

The next strange packet is below. This has no TCP flags set at all. Also notice the Windows Size and TCP Length are both set to 0. It is possible that this is an OS fingerprinting scan, or a corrupted TCP/IP stack, which is not sending what it should be sending.

```
05/03-06:09:52.614670 212.202.193.243:63899  -> MY.NET.233.146:1382
TCP TTL:114 TOS:0x0 ID:9258 IpLen:20 DgmLen:40 DF
******** Seq: 0x2FEEF85B  Ack: 0xDD74D5FE  Win:  0x0  TcpLen:  0
```

The rest of the anomalous packets (2.2%) seem to be either OS fingerprint scans or corrupted TCP/IP stacks.

3.10. Top Talkers

These four sections discuss the top talkers in each of the three categories (Alerts, Scans and Out-of-spec packets) and the overall rating. Each of the first three sections have two tables, which each list the internal and external top talkers respectively. The table formats used are based on Les Gordon's design for his GCIA practical.

I wanted to produce top talkers for alerts, scans, and OOS files to try to categorise what sort of traffic was being produce by which hosts. I figured that as the spp_portscan data from within the alert files was not being analysed, I would be able to see who was the top scanner, the top attacker, and the top corrupt packet generator respectively. Only then would I find out who was the most vociferous sender.

3.10.1. Alert Top Talkers

(Top 10 source IP's generating alerts)

Internal Source Top Talkers	
Int Src	# Pkts
MY.NET.210.114	354845
MY.NET.201.58	13311
MY.NET.235.110	13259
MY.NET.202.206	4997
MY.NET.201.38	4033
MY.NET.226.250	3460
MY.NET.97.213	2200
MY.NET.201.42	2007
MY.NET.226.206	1442
MY.NET.153.149	1321

Top 10 Internal Source IP's by Quantity

External Source Top Talkers		
Ext Src	# Pkts	DNS Name Resolved As
216.39.48.127	13959	buildrack52.sv.av.com
133.82.241.150	8412	cuapfs0.imit.chiba-u.ac.jp
12.231.84.153	6008	12-231-84-153.client.attbi.com
12.207.10.226	4966	12-207-10-226.client.attbi.com
128.46.117.76	4873	civl1240pc2.ecn.purdue.edu
67.161.246.193	3294	c-67-161-246-193.client.comcast.net
24.45.157.41	2966	ool-182d9d29.dyn.optonline.net
216.78.180.128	2639	adsl-78-180-128.lft.bellsouth.net
218.141.54.99	2552	YahooBB218141054099.bbtec.net
152.2.210.81	2403	metalab.unc.edu

Top 10 External Source IP's by Quantity

3.10.2. Top 10 Source and Destination Ports

Top 10 Source Ports	
Src Port	# Pkts
0	357579
65535	33583
1026	23304
1025	22408
137	20412
1027	20204
None	17634
1028	17496
1029	13637
5121	10045

Top 10 Source Ports

Top 10 Destination Ports	
Dest Port	# Pkts
0	357536
6667	206759
137	204293
80	64064
65535	29245
None	17635
5121	13031
2327	6008
69	5659
515	5039

Top 10 Destination Ports

3.10.3. Scans Top Talkers

(The top 10 scan generating source IP addresses)

Internal Source Top Talkers	
Int Src	# Pkts
MY.NET.210.114	354845
MY.NET.132.24	50444
MY.NET.240.62	40411
MY.NET.87.50	32605
MY.NET.250.98	29571
MY.NET.1.3	28663
MY.NET.97.190	26833

External Source Top Talkers		
Ext Src	# Pkts	DNS Name Resolved As
152.1.193.6	15962	chjlpc4.chem.ncsu.edu
217.88.231.137	13949	pD958E789.dip.t-dialin.net
217.84.122.16	11688	pD9547A10.dip.t-dialin.net
198.144.65.56	9160	nt-001-00055.greenapple.com
64.212.144.139	8313	No DNS - Frontier Communications
80.161.34.13	8244	0x50a1220d.kd4nxx15.adsl-dhcp.tele.dk
66.130.208.97	7663	modemcable097.208-130-66.que.mc.videotron.ca

MY.NET.234.158	20852
MY.NET.205.150	17296
MY.NET.153.152	15298

Top 10 Internal So urce IP's by Quantity

213.204.66.141	7040	No DNS - Yimpasnet Internet Iletisim Teknoloji As.
208.163.46.185	6939	port0185-cvx-pmbk.cwjamaica.com
12.16.131.99	6932	No DNS – Jackson Marketing

Top 10 External Source IP's by Quantity

3.10.4. Out-Of-Spec Top 10 Talkers

(Top 10 Out-of-Spec packet generating source IP Addresses)

Internal Source Top Talkers	
Int Src	# Pkts
MY.NET.12.4	26
MY.NET.12.2	5
MY.NET.194.179	5
MY.NET.241.82	4
MY.NET.252.14	3
MY.NET.17.30	1
MY.NET.104.113	1
MY.NET.40.11	1

All Internal Source IP's

External Source Top Talkers		
Ext Src	# Pkts	DNS Name Resolved As
209.123.49.137	1638	No DNS – (Net Access)
68.54.93.181	1049	pcp01781292pcs.howard01.md.comcast.net
213.197.10.95	370	010.095.dsl.concepts.nl
81.218.114.59	340	bzq-218-114-59.red.bezeqint.net
81.218.109.79	313	bzq-218-109-79.red.bezeqint.net
210.253.214.11	254	nttfis2-011.246.ne.jp
66.140.25.157	248	proxyscan.freenode.net
151.42.126.19	129	adsl-ull-19-126.42-151.net24.it
148.63.152.228	110	vsat-148-63-152-228.c189.t7.mrt.starband.net
193.233.7.104	105	light.inr.ac.ru

Top 10 External Source IP's by Quantity

3.10.5. Top 5 Overall Talkers

(Top 5 senders overall)

Internal Source Top Talkers	
Int Src	# Pkts
MY.NET.210.114	419777
MY.NET.132.24	50444
MY.NET.240.62	40411
MY.NET.87.50	32605
MY.NET.250.98	29571

Top 5 Internal Source IP's

External Source Top Talkers		
Ext Src	# Pkts	DNS Name Resolved As
152.1.193.6	15966	chjlpc4.chem.ncsu.edu
216.39.48.127	13959	buildrack52.sv.av.com
217.88.231.137	13951	pD958E789.dip.t-dialin.net
217.84.122.16	11689	pD9547A10.dip.t-dialin.net
198.144.65.56	10253	nt-001-00055.greenapple.com

Top 5 External Source IP's

3.10.6. External Registration Information

#1 External Top Talker	**IP Address:** 152.1.193.6
OrgName: North Carolina State University OrgID: NCSU Address: NCSU - Computing Center Box 7109 City: Raleigh StateProv: NC PostalCode: 27695 Country: US	NetRange: 152.1.0.0 - 152.1.255.255 CIDR: 152.1.0.0/16 NetName: NCSU TechName: Host, Master TechPhone: +1-919-515-7571 TechEmail: Hostmaster@ncsu.edu

#2 External Top Talker	**IP Address:** 216.39.48.127
OrgName: AltaVista Company OrgID: ALTAVI-1 Address: 1070 Arastradero Rd City: Palo Alto StateProv: CA PostalCode: 94304 Country: US NetRange: 216.39.48.0 - 216.39.63.255 CIDR: 216.39.48.0/20 NetName: NETBLK-INTERNET-BLK-1-AV	TechName: ALtaVista, Operations TechPhone: +1-650-320-7700 TechEmail: netops@av.com OrgAbuseName: Abuse OrgAbusePhone: +1-650-320-7700 OrgAbuseEmail: abuse@av.com OrgTechName: ALtaVista, Operations OrgTechPhone: +1-650-320-7700 OrgTechEmail: netops@av.com

#3 External Top Talker and **#4 External Top Talker**	**IP Address:** 217.88.231.137 and **IP Address:** 217.84.122.16
descr: Deutsche Telekom AG, Internet service provider address: Deutsche Telekom AG address: D-90449 Nuernberg address: Germany country: DE phone: +49 180 5334332 fax-no: +49 180 5334252 e-mail: ripe.dtip@telekom.de	inetnum: 217.80.0.0 - 217.89.31.255 person: Security Team address: Deutsche Telekom AG address: Germany phone: +49 180 5334332 fax-no: +49 180 5334252 e-mail: abuse@t-ipnet.de

#5 External Top Talker	**IP Address:** 198.144.65.56
OrgName: Green Apple, Inc. OrgID: GRNA Address: 127 W Sixth Ave Address: Suite D City: Lancaster StateProv: OH PostalCode: 43130 Country: US	NetRange: 198.144.64.0 - 198.144.95.255 CIDR: 198.144.64.0/19 NetName: GREENAPPLE OrgTechHandle: SUPPO33-ARIN OrgTechName: Hostmaster OrgTechPhone: +1-740-653-9890 OrgTechEmail: hostmaster@greenapple.com@av.com

Sender of BugBear Virus	**IP Address:** 160.94.128.49
OrgName: University of Minnesota OrgID: UNIVER-234 Address: Networking Services Address: Computer and Information Services Address: University of Minnesota Address: 130 Lind Hall Address: 207 Church St SE Address: Minneapolis MN 55455-0134 Country: US	NetRange: 160.94.0.0 - 160.94.255.255 CIDR: 160.94.0.0/16 AbuseName: UofM OIT Security and Assurance AbusePhone: +1-612-626-8639 AbuseEmail: abuse@umn.edu TechName: NTS - Technical Assistance Center TechPhone: +1-612-625-0006 TechEmail: nts@nts.umn.edu @av.com

3.11. Defensive Recommendations

Overall, the Universities network seems reasonably well protected. There are not any major issues to deal with apart from the large amount of false alerts generated by the snort sensor. There needs to be a balance between too many alerts to overwhelm, and not enough to detect when designing a rulebase. Unfortunately, I believe the University has erred on the side of too broadly specified rules. Some valid alerts are being buried under the weight of the false alerts (IIS Unicode, CGI Null, Exploit x86 NOOP, etc.). At present, their rules are specified too broadly and

need to be narrowed in their scope. This of course should be balanced against the need for low-level information for in-depth analysis.

The version of Snort currently being used by the University needs to be upgraded. From investigations of the error messages given by some of the Snort pre-processors it seems obvious that this version of Snort is v1.8 or earlier. The University should upgrade to v2.0.0, as it includes many bug fixes and performance enhancements. One of the big improvements in the newer versions are the Snort pre-processors, which generally are more efficient in their operation. This may go someway towards alleviating the problem the University is experiencing with log file corruption.

There is a need for a centrally managed enterprise level anti-virus solution. This would have a centrally managed structure, with automated virus updating feature to keep the protection current. This would enable an automated protection against infection, reducing the issues that the incident response team would need to deal with in person. Many anti-virus vendors' products detect known trojans and worms as well as viruses. This would help mitigate the vulnerability the University has with Nimda, and other viruses. This should be used in conjunction with policy and procedure control to force users who connect to the Universities network to have adequate anti-virus protection. While this may be too difficult to fully implement in a university environment with all its diversity, at least some control is needed to mitigate the virus threat.

All machines specifically noted as needing further investigations in the preceeding sections need to be checked for signs of compromise. It is important that these checks are made as the lack of full packet traces mean that we need to err on the side of caution. While it may take Incident response Staff a while to visit each of the machines in question, it would be less time than that needed to clear the organisations network if the security exploits spread.

Appendix A - ICMP Large ICMP Packet

```
12:33:29.516782 IP (tos 0x48, ttl 245, id 2 4826, len 1492) blueice1a.uk.ibm.com
> MY.SUB.NET.3: icmp 1472: echo request seq 0 (DF)
0x0000      4548 05d4 60fa 4000 f501 445d c2c4 643b        EH..`.@...C;..d;
0x0010      XXXX XX03 0800 32cc c533 0000 0000 0000        .........3......
0x0020      0000 0000 0000 0000 0000 0000 0000 0000        ................
--[repetitive 0000's]--
0x05c0      0000 0000 0000 0000 0000 0000 0000 0000        ................
0x05d0      0000 0000                                      ....
```

Appendix B - Logs of packets with a reserved bit set

```
10 May 2002
14:41:19.544488 IP 218.2.129.171 > 46.5.188.185: tcp (frag 0:20@36744)

11 May 2002
None

12 May 2002
None

13 May 2002
17:30:26.004488 IP 192.1.1.188 > 46.5.28.40: tcp (frag 0:20@17184)
21:44:51.094488 IP 192.1.1.188 > 46.5.39.57: tcp (frag 0:20@17184)

14 May 2002
05:52:57.814488 IP 192.1.1.188 > 46.5.170.11: tcp (frag 0:20@17184)
06:11:35.444488 IP 192.1.1.188 > 46.5.114.247: tcp (frag 0:20@17184)
10:49:42.254488 IP 192.1.1.188 > 46.5.15.28: tcp (frag 0:20@17184)
14:16:50.074488 IP 192.1.1.188 > 46.5.192.0: tcp (frag 0:20@17184)
15:17:51.374488 IP 192.1.1.188 > 46.5.73.71: tcp (frag 0:20@17184)
16:52:05.274488 IP 192.1.1.188 > 46.5.248.213: tcp (frag 0:20@17184)
17:10:46.974488 IP 192.1.1.188 > 46.5.157.208: tcp (frag 0:20@17184)
17:10:51.194488 IP 192.1.1.188 > 46.5.108.147: tcp (frag 0:20@17184)
17:54:43.974488 IP 192.1.1.188 > 46.5.69.163: tcp (frag 0:20@17184)
19:52:28.674488 IP 192.1.1.188 > 46.5.69.44: tcp (frag 0:20@17184)
20:45:32.674488 IP 192.1.1.188 > 46.5.174.220: tcp (frag 0:20@17184)
00:02:18.584488 IP 192.1.1.188 > 46.5.76.179: tcp (frag 0:20@17184)
00:59:09.394488 IP 192.1.1.188 > 46.5.136.29: tcp (frag 0:20@17184)

15 May 2002
01:38:45.764488 IP 192.1.1.188 > 46.5.230.126: tcp (frag 0:20@17184)
02:25:08.784488 IP 192.1.1.188 > 46.5.30.15: tcp (frag 0:20@17184)
07:54:52.984488 IP 192.1.1.188 > 46.5.144.147: tcp (frag 0:20@17184)
08:13:50.314488 IP 192.1.1.188 > 46.5.195.197: tcp (frag 0:20@17184)
09:38:31.684488 IP 192.1.1.188 > 46.5.88.21: tcp (frag 0:20@17184)
11:07:00.824488 IP 192.1.1.188 > 46.5.216.1: tcp (frag 0:20@17184)
11:37:15.474488 IP 192.1.1.188 > 46.5.56.12: tcp (frag 0:20@17184)
15:20:11.054488 IP 192.1.1.188 > 46.5.200.108: tcp (frag 0:20@17184)
16:00:01.464488 IP 192.1.1.188 > 46.5.211.185: tcp (frag 0:20@17184)
18:26:44.724488 IP 192.1.1.188 > 46.5.169.175: tcp (frag 0:20@17184)
19:09:38.764488 IP 192.1.1.188 > 46.5.136.117: tcp (frag 0:20@17184)
19:44:41.644488 IP 192.1.1.188 > 46.5.168.246: tcp (frag 0:20@17184)
22:22:24.834488 IP 192.1.1.188 > 46.5.20.2: tcp (frag 0:20@17184)

16 May 2002
04:25:49.004488 IP 192.1.1.188 > 46.5.226.187: tcp (frag 0:20@17184)
04:32:20.144488 IP 192.1.1.188 > 46.5.20.200: tcp (frag 0:20@17184)
09:11:01.594488 IP 192.1.1.188 > 46.5.132.21: tcp (frag 0:20@17184)
12:10:30.204488 IP 192.1.1.188 > 46.5.72.213: tcp (frag 0:20@17184)
14:14:48.704488 IP 192.1.1.188 > 46.5.130.136: tcp (frag 0:20@17184)
21:12:37.084488 IP 192.1.1.188 > 46.5.94.235: tcp (frag 0:20@17184)
21:43:14.304488 IP 192.1.1.188 > 46.5.178.85: tcp (frag 0:20@17184)
22:44:30.484488 IP 192.1.1.188 > 46.5.51.201: tcp (frag 0:20@17184)
00:36:39.194488 IP 192.1.1.188 > 46.5.109.103: tcp (frag 0:20@17184)
```

Appendix C - Both SHELLCODE x86 NOOP packets

```
Packet 1:
[**] SHELLCODE x86 NOOP [**]
11/10-17:12:47.636507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x5EA
130.94.22.249:80 -> 207.166.87.157:64741 TCP TTL:52 TOS:0x0 ID:23339 IpLen:20 DgmLen:1500
DF
***A**** Seq: 0xD454C3BC  Ack: 0xAC7EEF02  Win: 0x1920  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 05 DC 5B 2B 40 00 34 06 6F A0 82 5E 16 F9 CF A6  ..[+@.4.o..^....
0x0020: 57 9D 00 50 FC E5 D4 54 C3 BC AC 7E EF 02 50 10  W..P...T...~..P.
0x0030: 19 20 14 69 00 00 5B AD D5 58 40 00 FF 04 04 00  . .i..[..X@.....
0x0040: 00 00 00 00 00 00 40 00 FF 04 D4 00 00 00 80 04  ......@.........
0x0050: 31 02 FF E3 20 C0 CB 1D EA 9E 97 EC DB 46 B5 C7  1... ........F..
0x0060: 63 28 A9 86 0F 2A 0B 08 90 58 AE 54 75 63 2B 2A  c(...*...X.Tuc+*
0x0070: 22 CD 6F FF A9 9C 54 48 6B 95 0E 38 D3 00 00 01  ".o...THk..8....
0x0080: 6F FF FF 33 E4 C0 3C 58 A5 4E 4D 75 BA DA D0 CA  o..3..<X.NMu....
0x0090: C4 5D 2A C8 FD 3B 55 C8 EB AC 41 52 CD 88 0E 84  .]*..;U...AR....
0x00A0: A2 96 F5 AD 59 59 0F F4 AF 54 F9 3D 64 95 F6 BF  ....YY...T.=d...
0x00B0: F5 EB 7B 1B 54 37 25 B1 C1 B3 FF E3 20 C0 AE 1C  ..{.T7%..... ...
0x00C0: 4B 7A AF F8 C1 4A D9 8A 2A 0C B9 CF 24 38 F8 05  Kz...J..*...$8..
0x00D0: E0 69 A2 2E 3C EC D7 B4 7A 9A B1 56 B6 84 AE 48  .i..<...z..V...H
0x00E0: 00 00 19 6A AA 26 65 B1 9E 01 B7 2B 0D EB 48 61  ...j.&e....+..Ha
0x00F0: F4 0F AC F2 9D 03 F2 42 BA B2 79 79 9A E8 EA F7  .......B..yy....
0x0100: 9D 47 3A 71 83 CF 7E D4 91 5B DF F6 57 B2 AA 16  .G:q..~..[..W...
0x0110: 9E EF 94 19 66 DD 7A 65 3B 5D 75 B1 B9 D7 31 15  ....f.ze;]u...1.
0x0120: EE B0 40 00 FF 04 04 00 00 00 00 00 00 40 00 00  ..@..........@..
0x0130: FF 04 D4 00 00 00 80 04 49 01 FF E3 20 C0 97 15  ........I... ...
0x0140: 32 3E CF F8 69 44 B6 8C 47 7B 58 F0 C8 E1 33 43  2>..iD..G{X...3C
0x0150: F5 47 35 37 99 6E FE 69 25 6B 73 E8 B9 20 00 00  .G57.n.i%ks.. ..
0x0160: 5A BF FF 25 4C 0A E0 0E 3C 6F 3A 32 64 AB 73 B9  Z..%L...<o:2d.s.
0x0170: FA B2 7B F0 7D 76 E4 A3 93 7B A9 1F 69 F6 77 47  ..{.}v...{..i.wG
0x0180: D7 F6 91 6B CD B3 F5 76 75 2B 77 57 46 4D CF 95  ...k...vu+wWFM..
0x0190: 59 D5 E2 2C AD 7F A3 11 5D EA E4 21 DB 46 BC 96  Y..,....]..!.F..
0x01A0: 5C 87 FF E3 20 C0 9D 16 1A 7A BB F8 91 44 B6 49  \... ....z...D.I
0x01B0: 1B 21 11 13 6B 2A EE EC D4 BD B5 FE 89 90 B4 A6  .!..k*..........
0x01C0: 73 9C 70 00 11 5A FF FF 28 96 1E 02 A9 E2 52 B9  s.p..Z..(.....R.
0x01D0: 99 EB 2C F4 D2 8F 11 4D 93 DB 91 D2 B5 DE 7E 77  ..,....M......~w
0x01E0: 57 46 29 3B C4 32 31 68 94 95 76 A7 8E B9 F6 FD  WF);.21h..v.....
0x01F0: AC D3 A1 59 26 EC 85 7A D9 DE 87 65 A6 CE DA 2A  ...Y&..z...e...*
0x0200: EE 76 5A B5 0E 8B DB 57 B6 89 40 00 BF 00 1A 00  .vZ....W..@.....
0x0210: 00 00 11 00 6A F5 97 D4 38 E2 4A 80 00 01 14 00  ....j...8.J.....
0x0220: 00 00 00 01 25 AB E0 1C C7 75 E6 00 86 06 06 5D  ....%....u.....]
0x0230: 00 11 00 00 FF 04 04 00 00 00 00 00 00 00 40 00  ..............@.
0x0240: BF 00 1A 00 00 00 12 00 6A F5 97 D4 38 E3 5B 00  ........j...8.[.
0x0250: 00 01 14 00 00 00 00 01 25 AB E0 1C C7 95 8A 00  ........%.......
0x0260: 85 06 03 5D 00 12 00 FF 04 6C 00 00 00 40 02 61  ...].....l...@.a
0x0270: 00 FF E3 20 C0 9F 15 E3 86 CB F8 50 84 DA 2B C9  ... .......P..+.
0x0280: 44 79 0B 67 DD 9D 57 BB A4 F7 76 5C 88 F5 D7 61  Dy.g..W...v\...a
0x0290: 2A 56 18 00 01 A2 AA AA 28 44 4C 03 E3 7E 56 B5  *V......(DL..~V.
0x02A0: 6B 5F BE 48 BD 72 FF CA 7F F2 52 E7 FC 0B FF 9B  k_.H.r....R.....
0x02B0: FF FF F9 DF 6C 8A FD 9B 10 61 A3 6F 4E D0 39 92  ....l....a.oN.9.
0x02C0: 01 00 9A 6C 4A 2D 36 32 2F 20 C1 D8 7C 10 04 82  ...lJ-62/ ..|...
0x02D0: 91 DF 7F 6C 96 4B F6 7C DF 40 00 BF 00 1A 00 00  ...l.K.|.@......
0x02E0: 00 13 00 6A F5 97 D4 38 E3 F0 80 00 01 14 00 00  ...j...8........
0x02F0: 00 00 01 25 AB E0 1C C7 A5 0F C0 85 06 03 5D 00  ...%..........].
0x0300: 13 00 FF 04 04 00 00 00 00 00 00 40 00 3F 02  ...........@.?.
0x0310: 3E 02 00 00 FF D8 FF DB 00 43 00 10 0B 0C 0E 0C  >........C......
0x0320: 0A 10 0E 0D 0E 12 11 10 13 18 27 19 18 16 16 18  ..........'.....
0x0330: 30 22 24 1C 27 39 32 3C 3B 38 32 37 36 3F 47 5A  0"$.'92<;8276?GZ
0x0340: 4C 3F 43 55 44 36 37 4E 6B 4F 55 5D 60 65 66 65  L?CUD67NkOU]`efe
0x0350: 3D 4B 6F 77 6E 62 76 5A 63 65 61 FF DB 00 43 01  =KownbvZcea...C.
0x0360: 11 12 12 18 15 18 2E 19 19 2E 61 41 37 41 61 61  ..........aA7Aaa
0x0370: 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
0x0380: 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
0x0390: 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
0x03A0: FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00 00  ................
0x03B0: 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0A  ................
0x03C0: 0B FF C4 00 B5 10 00 02 01 03 03 02 04 03 05 05  ................
0x03D0: 04 04 00 00 01 7D 01 02 03 00 04 11 05 12 21 31  .....}........!1
0x03E0: 41 06 13 51 61 07 22 71 14 32 81 91 A1 08 23 42  A..Qa."q.2....#B
0x03F0: B1 C1 15 52 D1 F0 24 33 62 72 82 09 0A 16 17 18  ...R..$3br......
0x0400: 19 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A 43  ..%&'()*456789:C
0x0410: 44 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A 63  DEFGHIJSTUVWXYZc
0x0420: 64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A 83  defghijstuvwxyz.
0x0430: 84 85 86 87 88 89 8A 92 93 94 95 96 97 98 99 9A  ................
```

```
0x0440: A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8  ................
0x0450: B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6  ................
0x0460: D7 D8 D9 DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1 F2  ................
0x0470: F3 F4 F5 F6 F7 F8 F9 FA FF C4 00 1F 01 00 03 01  ................
0x0480: 01 01 01 01 01 01 01 01 00 00 00 00 00 00 01 02  ................
0x0490: 03 04 05 06 07 08 09 0A 0B FF C4 00 B5 11 00 02  ................
0x04A0: 01 02 04 04 03 04 07 05 04 04 00 01 02 77 00 01  .............w..
0x04B0: 02 03 11 04 05 21 31 06 12 41 51 07 61 71 13 22  .....!1..AQ.aq."
0x04C0: 32 81 08 14 42 91 A1 B1 C1 09 23 33 52 F0 15 62  2...B.....#3R..b
0x04D0: 72 D1 0A 16 24 34 E1 25 F1 17 18 19 1A 26 27 28  r...$4.%.....&'(
0x04E0: 29 2A 35 36 37 38 39 3A 43 44 45 46 47 48 49 4A  )*56789:CDEFGHIJ
0x04F0: 53 54 55 56 57 58 59 5A 63 64 65 66 67 68 69 6A  STUVWXYZcdefghij
0x0500: 73 74 75 76 77 78 79 7A 82 83 84 85 86 87 88 89  stuvwxyz........
0x0510: 8A 92 93 94 95 96 97 98 99 9A A2 A3 A4 A5 A6 A7  ................
0x0520: A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5  ................
0x0530: C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E2 E3  ................
0x0540: E4 E5 E6 E7 E8 E9 EA F2 F3 F4 F5 F6 F7 F8 F9 FA  ................
0x0550: FF D9 BF 01 63 16 00 00 14 00 FF D8 FF E0 00 10  ....c...........
0x0560: 4A 46 49 46 00 01 01 00 00 01 00 01 00 00 FF C0  JFIF............
0x0570: 00 11 08 00 D2 01 0E 03 01 22 00 02 11 01 03 11  ........."......
0x0580: 01 FF DA 00 0C 03 01 00 02 11 03 11 00 3F 00 F4  .............?..
0x0590: 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80  .(...(...(...(..
0x05A0: 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80  .(...(...(...(..
0x05B0: 0A 29 AC EA 83 2E C1 47 A9 38 AA 93 6A D6 10 E7  .).....G.8..j...
0x05C0: 7D CA 13 E8 A7 77 F2 A0 0B B4 56 1C DE 29 B1 8F  }....w....V..)..
0x05D0: EE 2C 92 1F A0 15 46 7F 15 4F B3 7C 56 81 53 38  .,....F..O.|V.S8
0x05E0: DE F9 22 95 C0 EA A9 2B 85 9B                    .."....+..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Packet 2:

```
[**] SHELLCODE x86 NOOP [**]
11/10-17:13:13.236507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x5EA
130.94.22.249:80 -> 207.166.87.157:64741 TCP TTL:52 TOS:0x0 ID:23403 IpLen:20 DgmLen:1500
DF
***A**** Seq: 0xD456219D  Ack: 0xAC7EF2EC  Win: 0x2180  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 05 DC 5B 6B 40 00 34 06 6F 60 82 5E 16 F9 CF A6  ..[k@.4.o`.^....
0x0020: 57 9D 00 50 FC E5 D4 56 21 9D AC 7E F2 EC 50 10  W..P...V!..~..P.
0x0030: 21 80 A5 D2 00 00 75 AB 97 7E 7E 6D 8F 47 65 A3  !.....u..~~m.Ge.
0x0040: 88 95 75 57 43 AE 69 AB 6A B3 A5 EA 66 B8 E5 55  ..uWC.i.j...f..U
0x0050: 42 A4 88 72 0B A8 D1 ED 51 53 1E 14 FF 4F C3 71  B..r....QS...O.q
0x0060: CA C5 1E 4F 37 BA 9D C9 4D ED 7B 59 4E 3B DB D0  ...O7...M.{YN;..
0x0070: 06 A9 8C 58 D3 8A BD 87 C3 D5 AD 13 A7 D6 4C A0  ...X..........L.
0x0080: 84 0A 04 40 B8 FF E3 20 C0 D7 16 C2 66 B7 F2 C1  ...@... ....f...
0x0090: 44 B7 4B 55 90 53 D3 0D 40 25 D2 2B 48 40 27 5C  D.KU.S..@%.+H@'\
0x00A0: E1 60 D0 2D 54 EA 4E F2 E3 9D 98 3D A9 CD ED 33  .`.-T.N....=...3
0x00B0: 73 73 79 77 11 A6 C7 AC 2C BE 2D 91 B9 82 6E 67  ssyw....,.-...ng
0x00C0: 44 77 FA 53 39 96 BB F3 23 85 C5 28 BE B2 53 CB  Dw.S9...#..(..S.
0x00D0: 45 96 6E 64 A6 E6 41 DB D4 18 93 A1 4A 58 45 71  E.nd..A.....JXEq
0x00E0: 9F 53 96 CE 42 C8 27 B4 E6 6F EF FD FC 40 00 BF  .S..B.'..o...@..
0x00F0: 00 1A 00 00 00 11 00 6A 85 14 50 38 E3 5B 00 00  .......j..P8.[..
0x0100: 01 14 00 00 00 00 01 25 AA 1E 1C C7 95 8A 00 85  .......%........
0x0110: 06 03 5D 00 11 00 FF 04 04 00 00 00 00 00 00 00  ..].............
0x0120: 40 00 BF 00 1A 00 00 00 12 00 6A 85 14 50 38 E3  @.........j..P8.
0x0130: F0 80 00 01 14 00 00 00 00 01 25 AA 1E 1C C7 A5  ..........%.....
0x0140: 0F C0 85 06 03 5D 00 12 00 FF 04 6C 00 00 00 40  .....].....l...@
0x0150: 02 D9 00 FF E3 20 C0 D7 19 AA 8E C7 F8 79 CA B7  ..... .......y..
0x0160: BE 7D C8 50 39 05 D9 D0 00 00 0F FF EF 3E 82 14  .}.P9........>..
0x0170: 9F A0 04 52 BA BA 8E 62 ED CE C1 BD A1 A2 E5 EA  ...R...b........
0x0180: EE 37 6A 40 EC 95 61 8A 9B F3 46 CC 3E D2 46 9C  .7j@..a...F.>.F.
0x0190: F2 B1 05 AB 74 15 89 DB F6 BA 12 E8 5F DB A5 7E  ....t......._..~
0x01A0: A8 E5 FB 5B 7F FE 8B 49 4C A5 56 34 71 95 4C C6  ...[...IL.V4q.L.
0x01B0: 29 05 83 A1 21 A4 64 5B AD D5 58 40 00 3F 02 3E  )...!.d[..X@.?.>
0x01C0: 02 00 00 FF D8 FF DB 00 43 00 10 0B 0C 0E 0C 0A  ........C.......
0x01D0: 10 0E 0D 0E 12 11 10 13 18 27 19 18 16 16 18 30  .........'.....0
0x01E0: 22 24 1C 27 39 32 3C 3B 38 32 37 36 3F 47 5A 4C  "$.'92<;8276?GZL
0x01F0: 3F 43 55 44 36 37 4E 6B 4F 55 5D 60 65 66 65 3D  ?CUD67NkOU]`efe=
0x0200: 4B 6F 77 6E 62 76 5A 63 65 61 FF DB 00 43 01 11  KownbvZcea...C..
0x0210: 12 12 18 15 18 2E 19 19 2E 61 41 37 41 61 61 61  .........aA7Aaaa
0x0220: 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
0x0230: 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
0x0240: 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 FF  aaaaaaaaaaaaaaa.
0x0250: C4 00 1F 00 00 01 05 01 01 01 01 01 01 00 00 00  ................
0x0260: 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0A 0B  ................
0x0270: FF C4 00 B5 10 00 02 01 03 03 02 04 03 05 05 04  ................
0x0280: 04 00 00 01 7D 01 02 03 00 04 11 05 12 21 31 41  ....}........!1A
0x0290: 06 13 51 61 07 22 71 14 32 81 91 A1 08 23 42 B1  ..Qa."q.2....#B.
0x02A0: C1 15 52 D1 F0 24 33 62 72 82 09 0A 16 17 18 19  ..R..$3br.......
```

```
0x02B0: 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A 43 44  .%&'()*456789:CD
0x02C0: 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A 63 64  EFGHIJSTUVWXYZcd
0x02D0: 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A 83 84  efghijstuvwxyz..
0x02E0: 85 86 87 88 89 8A 92 93 94 95 96 97 98 99 9A A2  ................
0x02F0: A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9  ................
0x0300: BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7  ................
0x0310: D8 D9 DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1 F2 F3  ................
0x0320: F4 F5 F6 F7 F8 F9 FA FF C4 00 1F 01 00 03 01 01  ................
0x0330: 01 01 01 01 01 01 01 00 00 00 00 00 00 01 02 03  ................
0x0340: 04 05 06 07 08 09 0A 0B FF C4 00 B5 11 00 02 01  ................
0x0350: 02 04 04 03 04 07 05 04 04 00 01 02 77 00 01 02  ............w...
0x0360: 03 11 04 05 21 31 06 12 41 51 07 61 71 13 22 32  ....!1..AQ.aq."2
0x0370: 81 08 14 42 91 A1 B1 C1 09 23 33 52 F0 15 62 72  ...B.....#3R..br
0x0380: D1 0A 16 24 34 E1 25 F1 17 18 19 1A 26 27 28 29  ...$4.%.....&'()
0x0390: 2A 35 36 37 38 39 3A 43 44 45 46 47 48 49 4A 53  *56789:CDEFGHIJS
0x03A0: 54 55 56 57 58 59 5A 63 64 65 66 67 68 69 6A 73  TUVWXYZcdefghijs
0x03B0: 74 75 76 77 78 79 7A 82 83 84 85 86 87 88 89 8A  tuvwxyz.........
0x03C0: 92 93 94 95 96 97 98 99 9A A2 A3 A4 A5 A6 A7 A8  ................
0x03D0: A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5 C6  ................
0x03E0: C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E2 E3 E4  ................
0x03F0: E5 E6 E7 E8 E9 EA F2 F3 F4 F5 F6 F7 F8 F9 FA FF  ................
0x0400: D9 BF 01 50 2E 00 00 13 00 FF D8 FF E0 00 10 4A  ...P...........J
0x0410: 46 49 46 00 01 01 00 00 01 00 01 00 00 FF C0 00  FIF.............
0x0420: 11 08 00 E7 01 0E 03 01 22 00 02 11 01 03 11 01  ........".......
0x0430: FF DA 00 0C 03 01 00 02 11 03 11 00 3F 00 F4 0A  ............?...
0x0440: 28 A4 A0 05 A2 92 90 9A 00 75 25 26 69 BB C5 00  (........u%&i...
0x0450: 3E 8A 6E EA 37 71 9A 00 75 14 DD DC 51 BB 34 00  >.n.7q..u...Q.4.
0x0460: EA 29 9B F0 79 A5 DD 40 0E A2 98 5C 51 BE 80 1F  .)..y..@...\Q...
0x0470: 45 37 77 14 6E 18 CD 00 3A 8A 6E EE 29 0C 80 0C  E7w.n...:.n.)...
0x0480: 93 40 0F A2 A0 6B B8 95 B0 5C 66 95 2E 63 71 94  .@...k...\f..cq.
0x0490: 60 47 A8 A0 09 A8 A6 EF 14 9B C5 00 3E 8A 6E FA  `G..........>.n.
0x04A0: 37 50 03 A8 A6 96 A4 DE 28 01 F4 53 77 51 BA 80  7P......(..SwQ..
0x04B0: 1D 45 37 75 1B A8 01 F4 53 73 4B 40 0B 45 25 2D  .E7u....SsK@.E%-
0x04C0: 00 14 94 B4 86 80 1A 4D 44 EF 8E A6 96 46 C6 7E  .......MD....F.~
0x04D0: 95 52 57 3E 60 00 FC BD 4F B5 20 27 92 60 8B 82  .RW>`...O. '.`..
0x04E0: 6A B9 BB 0B D4 D5 1B DB B5 0E 50 BE D6 54 CE 33  j.........P..T.3
0x04F0: 8C D6 5A 6A 00 B6 D0 57 2D 91 96 F5 ED 42 D5 5C  ..Zj...W-....B.\
0x0500: 0E A1 27 0D C0 3C 9E 94 91 CE 1F 70 15 CF E9 F7  ..'..<.....p....
0x0510: E1 EE 04 33 65 0F BF 06 AD 2C AF 0D F9 0E 40 12  ...3e....,....@.
0x0520: A9 65 CF 7C 1E 6B 3E 6B 3B 30 35 3E D0 07 DE 3C  .e.|.k>k;05>...<
0x0530: 1C 7E 14 EF 3B 1E D8 15 85 75 75 E5 99 11 9B E5  .~..;....uu.....
0x0540: 19 19 F4 CF 43 45 BE A4 B2 16 5C 9C A6 08 27 8E  ....CE....\...'.
0x0550: BD A8 E6 D2 E8 46 BB DC 8E 46 46 40 CF D6 A4 4B  .....F...FF@...K
0x0560: 90 D1 82 3B 62 B9 8B AD 40 09 5D 77 1D CB 82 30  ...;b...@.]w...0
0x0570: 32 4A F7 E3 D4 53 ED 2E EE DE 2D 8D 0B AE 25 20  2J...S....-...%
0x0580: B3 70 31 D8 8F 63 54 A4 9A 19 D1 A5 C0 7D C3 F8  .p1..cT......}..
0x0590: 97 A8 A8 DA ED 15 58 B3 8F 97 9A E5 AF B5 26 D3  ......X.......&.
0x05A0: AE 1E 79 A7 8D D9 87 16 F1 36 E7 1E E7 B0 15 99  ..y......6......
0x05B0: 16 BC 93 5C 93 2D AC 85 5B 24 05 97 E6 5F 6F 70  ...\.-..[$..._op
0x05C0: 69 DD DF 40 3B A8 F5 48 37 F9 6D 20 C9 E7 AD 55  i..@;..H7.m ...U
0x05D0: 7D 76 24 69 E3 45 67 68 CE 3E 50 4E 7D 2B 92 7B  }v$i.Egh.>PN}+.{
0x05E0: EB AF 35 7E C3 A5 B4 64 0C B7                    ..5~...d..
```

Appendix D - Possible Worm Infected Machines

Probable Worm Infected Machines						
MY.NET.101.44	MY.NET.153.127	MY.NET.198.217	MY.NET.226.86	MY.NET.53.49	MY.NET.97.100	MY.NET.97.48
MY.NET.104.119	MY.NET.153.135	MY.NET.198.254	MY.NET.226.98	MY.NET.53.53	MY.NET.97.102	MY.NET.97.49
MY.NET.104.155	MY.NET.153.136	MY.NET.201.102	MY.NET.227.106	MY.NET.53.54	MY.NET.97.103	MY.NET.97.50
MY.NET.105.140	MY.NET.153.137	MY.NET.201.250	MY.NET.227.142	MY.NET.53.55	MY.NET.97.104	MY.NET.97.51
MY.NET.105.22	MY.NET.153.143	MY.NET.202.118	MY.NET.227.2	MY.NET.53.56	MY.NET.97.105	MY.NET.97.52
MY.NET.106.102	MY.NET.153.144	MY.NET.202.134	MY.NET.227.94	MY.NET.53.57	MY.NET.97.107	MY.NET.97.53
MY.NET.106.103	MY.NET.153.146	MY.NET.202.158	MY.NET.228.158	MY.NET.53.58	MY.NET.97.141	MY.NET.97.54
MY.NET.106.106	MY.NET.153.147	MY.NET.203.106	MY.NET.229.54	MY.NET.53.60	MY.NET.97.143	MY.NET.97.55
MY.NET.106.176	MY.NET.153.149	MY.NET.203.150	MY.NET.233.18	MY.NET.53.72	MY.NET.97.145	MY.NET.97.56
MY.NET.106.24	MY.NET.153.150	MY.NET.203.174	MY.NET.234.110	MY.NET.53.76	MY.NET.97.147	MY.NET.97.57
MY.NET.106.89	MY.NET.153.152	MY.NET.203.206	MY.NET.234.118	MY.NET.53.92	MY.NET.97.15	MY.NET.97.62
MY.NET.107.74	MY.NET.153.153	MY.NET.203.6	MY.NET.234.154	MY.NET.54.207	MY.NET.97.150	MY.NET.97.63
MY.NET.107.76	MY.NET.153.154	MY.NET.203.82	MY.NET.234.70	MY.NET.54.23	MY.NET.97.154	MY.NET.97.64
MY.NET.109.101	MY.NET.153.157	MY.NET.204.210	MY.NET.235.202	MY.NET.54.28	MY.NET.97.155	MY.NET.97.66
MY.NET.110.211	MY.NET.153.159	MY.NET.204.26	MY.NET.235.206	MY.NET.54.33	MY.NET.97.16	MY.NET.97.67
MY.NET.110.234	MY.NET.153.163	MY.NET.204.58	MY.NET.235.22	MY.NET.60.170	MY.NET.97.160	MY.NET.97.69
MY.NET.111.30	MY.NET.153.164	MY.NET.204.6	MY.NET.235.6	MY.NET.60.89	MY.NET.97.161	MY.NET.97.70
MY.NET.112.165	MY.NET.153.165	MY.NET.204.74	MY.NET.235.74	MY.NET.65.11	MY.NET.97.162	MY.NET.97.71
MY.NET.112.187	MY.NET.153.168	MY.NET.205.10	MY.NET.236.106	MY.NET.70.16	MY.NET.97.163	MY.NET.97.72
MY.NET.112.193	MY.NET.153.170	MY.NET.205.118	MY.NET.236.142	MY.NET.71.164	MY.NET.97.164	MY.NET.97.73
MY.NET.112.204	MY.NET.153.172	MY.NET.205.130	MY.NET.236.170	MY.NET.75.107	MY.NET.97.165	MY.NET.97.74

Probable Worm Infected Machines						
MY.NET.112.220	MY.NET.153.173	MY.NET.205.146	MY.NET.236.202	MY.NET.75.133	MY.NET.97.167	MY.NET.97.75
MY.NET.112.32	MY.NET.153.176	MY.NET.205.186	MY.NET.236.238	MY.NET.75.148	MY.NET.97.168	MY.NET.97.77
MY.NET.112.38	MY.NET.153.182	MY.NET.205.190	MY.NET.236.26	MY.NET.75.6	MY.NET.97.171	MY.NET.97.78
MY.NET.115.167	MY.NET.153.185	MY.NET.206.170	MY.NET.236.54	MY.NET.83.171	MY.NET.97.173	MY.NET.97.79
MY.NET.115.175	MY.NET.153.198	MY.NET.206.186	MY.NET.236.74	MY.NET.83.217	MY.NET.97.174	MY.NET.97.80
MY.NET.116.75	MY.NET.153.201	MY.NET.207.190	MY.NET.236.90	MY.NET.83.248	MY.NET.97.176	MY.NET.97.81
MY.NET.116.84	MY.NET.153.202	MY.NET.207.214	MY.NET.237.170	MY.NET.83.93	MY.NET.97.179	MY.NET.97.83
MY.NET.118.6	MY.NET.153.208	MY.NET.207.46	MY.NET.237.186	MY.NET.84.141	MY.NET.97.18	MY.NET.97.84
MY.NET.130.150	MY.NET.153.209	MY.NET.207.86	MY.NET.237.98	MY.NET.84.146	MY.NET.97.180	MY.NET.97.85
MY.NET.130.60	MY.NET.153.210	MY.NET.208.122	MY.NET.239.222	MY.NET.84.147	MY.NET.97.182	MY.NET.97.87
MY.NET.130.73	MY.NET.153.213	MY.NET.208.38	MY.NET.239.42	MY.NET.84.16	MY.NET.97.188	MY.NET.97.89
MY.NET.138.15	MY.NET.153.46	MY.NET.209.226	MY.NET.240.154	MY.NET.84.216	MY.NET.97.195	MY.NET.97.90
MY.NET.138.16	MY.NET.157.49	MY.NET.209.242	MY.NET.240.86	MY.NET.84.218	MY.NET.97.196	MY.NET.97.91
MY.NET.138.24	MY.NET.162.194	MY.NET.210.158	MY.NET.240.94	MY.NET.85.52	MY.NET.97.197	MY.NET.97.93
MY.NET.138.46	MY.NET.163.233	MY.NET.210.50	MY.NET.241.162	MY.NET.85.87	MY.NET.97.198	MY.NET.97.94
MY.NET.141.102	MY.NET.168.115	MY.NET.210.62	MY.NET.242.10	MY.NET.86.110	MY.NET.97.20	MY.NET.97.95
MY.NET.143.107	MY.NET.168.125	MY.NET.211.158	MY.NET.242.250	MY.NET.86.71	MY.NET.97.200	MY.NET.97.96
MY.NET.144.51	MY.NET.168.139	MY.NET.212.110	MY.NET.242.30	MY.NET.87.100	MY.NET.97.202	MY.NET.97.97
MY.NET.145.197	MY.NET.168.154	MY.NET.212.174	MY.NET.242.58	MY.NET.87.107	MY.NET.97.203	MY.NET.97.99
MY.NET.145.199	MY.NET.168.161	MY.NET.212.30	MY.NET.249.114	MY.NET.87.121	MY.NET.97.207	MY.NET.98.100
MY.NET.145.27	MY.NET.168.165	MY.NET.217.194	MY.NET.249.138	MY.NET.87.126	MY.NET.97.209	MY.NET.98.102
MY.NET.149.16	MY.NET.168.166	MY.NET.217.234	MY.NET.249.222	MY.NET.87.148	MY.NET.97.213	MY.NET.98.104
MY.NET.15.212	MY.NET.168.170	MY.NET.217.42	MY.NET.250.122	MY.NET.87.193	MY.NET.97.215	MY.NET.98.105
MY.NET.15.222	MY.NET.168.179	MY.NET.217.6	MY.NET.250.206	MY.NET.87.70	MY.NET.97.216	MY.NET.98.11
MY.NET.150.121	MY.NET.168.183	MY.NET.218.158	MY.NET.250.230	MY.NET.87.89	MY.NET.97.218	MY.NET.98.128
MY.NET.150.137	MY.NET.168.205	MY.NET.218.182	MY.NET.251.102	MY.NET.88.101	MY.NET.97.219	MY.NET.98.139
MY.NET.150.203	MY.NET.168.214	MY.NET.218.2	MY.NET.251.162	MY.NET.88.130	MY.NET.97.22	MY.NET.98.14
MY.NET.150.210	MY.NET.168.219	MY.NET.218.22	MY.NET.251.206	MY.NET.88.131	MY.NET.97.226	MY.NET.98.15
MY.NET.151.120	MY.NET.168.234	MY.NET.219.214	MY.NET.251.70	MY.NET.88.143	MY.NET.97.227	MY.NET.98.152
MY.NET.151.124	MY.NET.168.56	MY.NET.220.134	MY.NET.252.150	MY.NET.88.148	MY.NET.97.228	MY.NET.98.153
MY.NET.151.85	MY.NET.17.54	MY.NET.220.18	MY.NET.252.182	MY.NET.88.149	MY.NET.97.229	MY.NET.98.156
MY.NET.152.12	MY.NET.178.140	MY.NET.220.26	MY.NET.252.202	MY.NET.88.150	MY.NET.97.23	MY.NET.98.172
MY.NET.152.126	MY.NET.178.23	MY.NET.220.34	MY.NET.253.130	MY.NET.88.167	MY.NET.97.230	MY.NET.98.18
MY.NET.152.15	MY.NET.18.30	MY.NET.221.154	MY.NET.253.170	MY.NET.88.171	MY.NET.97.233	MY.NET.98.28
MY.NET.152.160	MY.NET.183.25	MY.NET.221.162	MY.NET.53.101	MY.NET.88.182	MY.NET.97.234	MY.NET.98.30
MY.NET.152.163	MY.NET.188.19	MY.NET.221.38	MY.NET.53.105	MY.NET.88.199	MY.NET.97.237	MY.NET.98.36
MY.NET.152.165	MY.NET.189.41	MY.NET.221.42	MY.NET.53.120	MY.NET.88.210	MY.NET.97.238	MY.NET.98.38
MY.NET.152.167	MY.NET.189.54	MY.NET.222.146	MY.NET.53.128	MY.NET.88.225	MY.NET.97.239	MY.NET.98.44
MY.NET.152.173	MY.NET.193.161	MY.NET.222.166	MY.NET.53.133	MY.NET.88.254	MY.NET.97.24	MY.NET.98.45
MY.NET.152.183	MY.NET.193.217	MY.NET.222.170	MY.NET.53.143	MY.NET.88.75	MY.NET.97.241	MY.NET.98.47
MY.NET.152.22	MY.NET.194.227	MY.NET.222.214	MY.NET.53.158	MY.NET.91.100	MY.NET.97.243	MY.NET.98.48
MY.NET.152.248	MY.NET.194.31	MY.NET.222.246	MY.NET.53.160	MY.NET.91.101	MY.NET.97.245	MY.NET.98.49
MY.NET.153.105	MY.NET.194.5	MY.NET.222.74	MY.NET.53.180	MY.NET.91.104	MY.NET.97.247	MY.NET.98.52
MY.NET.153.107	MY.NET.194.91	MY.NET.223.146	MY.NET.53.184	MY.NET.91.109	MY.NET.97.249	MY.NET.98.55
MY.NET.153.110	MY.NET.195.143	MY.NET.223.170	MY.NET.53.185	MY.NET.91.119	MY.NET.97.25	MY.NET.98.62
MY.NET.153.111	MY.NET.195.31	MY.NET.224.106	MY.NET.53.214	MY.NET.91.120	MY.NET.97.26	MY.NET.98.66
MY.NET.153.112	MY.NET.195.83	MY.NET.224.126	MY.NET.53.222	MY.NET.91.139	MY.NET.97.27	MY.NET.98.69
MY.NET.153.113	MY.NET.195.89	MY.NET.224.54	MY.NET.53.30	MY.NET.91.147	MY.NET.97.29	MY.NET.98.75
MY.NET.153.114	MY.NET.196.123	MY.NET.225.170	MY.NET.53.32	MY.NET.91.2	MY.NET.97.35	MY.NET.98.77
MY.NET.153.115	MY.NET.196.181	MY.NET.225.34	MY.NET.53.35	MY.NET.91.8	MY.NET.97.38	MY.NET.98.82
MY.NET.153.117	MY.NET.196.7	MY.NET.226.106	MY.NET.53.37	MY.NET.91.85	MY.NET.97.40	MY.NET.98.85
MY.NET.153.118	MY.NET.197.2	MY.NET.226.110	MY.NET.53.38	MY.NET.91.92	MY.NET.97.41	MY.NET.98.86
MY.NET.153.120	MY.NET.197.42	MY.NET.226.174	MY.NET.53.39	MY.NET.91.93	MY.NET.97.42	MY.NET.98.98
MY.NET.153.123	MY.NET.197.70	MY.NET.226.198	MY.NET.53.41	MY.NET.91.95	MY.NET.97.43	MY.NET.99.165
MY.NET.153.124	MY.NET.198.101	MY.NET.226.206	MY.NET.53.44	MY.NET.91.96	MY.NET.97.44	
MY.NET.153.125	MY.NET.198.175	MY.NET.226.78	MY.NET.53.47	MY.NET.97.10	MY.NET.97.46	

Appendix E – Possible Peer-to-Peer Clients

Possible Peer-to-Peer Clients						
MY.NET.240.62	MY.NET.218.222	MY.NET.233.250	MY.NET.205.178	MY.NET.249.13	MY.NET.219.18	MY.NET.238.23
MY.NET.207.230	MY.NET.217.170	MY.NET.223.250	MY.NET.225.70	MY.NET.210.94	MY.NET.249.12	MY.NET.227.70
MY.NET.233.10	MY.NET.242.42	MY.NET.242.94	MY.NET.240.130	MY.NET.226.25	MY.NET.239.17	MY.NET.207.10
MY.NET.206.70	MY.NET.250.78	MY.NET.202.234	MY.NET.209.206	MY.NET.205.21	MY.NET.223.94	MY.NET.98.36
MY.NET.70.176	MY.NET.226.178	MY.NET.207.34	MY.NET.210.122	MY.NET.208.22	MY.NET.97.44	MY.NET.113.4
MY.NET.206.130	MY.NET.211.154	MY.NET.223.106	MY.NET.53.43	MY.NET.193.13	MY.NET.252.19	MY.NET.221.2
MY.NET.201.38	MY.NET.201.234	MY.NET.203.42	MY.NET.253.106	MY.NET.224.11	MY.NET.237.22	MY.NET.240.11
MY.NET.241.118	MY.NET.244.182	MY.NET.194.13	MY.NET.201.186	MY.NET.168.17	MY.NET.242.15	MY.NET.238.23
MY.NET.233.134	MY.NET.240.2	MY.NET.195.209	MY.NET.220.54	MY.NET.84.216	MY.NET.242.13	MY.NET.227.70
MY.NET.253.102	MY.NET.211.198	MY.NET.217.54	MY.NET.228.62	MY.NET.253.15	MY.NET.225.62	MY.NET.207.10

Appendix F – Probable XDCC or SdBot infected Clients

Probable XDCC or SdBot infected Clients						
MY.NET.83.100	MY.NET.132.27	MY.NET.84.250	MY.NET.194.125	MY.NET.112.19	MY.NET.80.149	MY.NET.223.78
MY.NET.97.128	MY.NET.101.42	MY.NET.195.99	MY.NET.105.48	MY.NET.105.20	MY.NET.226.25	MY.NET.198.22

Appendix G – Listing of prepalerts.pl, prespscans.pl and prepoos.pl

Prepalerts.pl

```
#!/cygdrive/c/Perl/bin/perl.exe
# Name: prepalerts.pl
# By Terry MacDonald
# Based on csv.pl by Tod Beardsley
# Usage: prepalerts.pl infile [infile2] [infile3]
unless ($ARGV[0]) {
  print "You didn't specify the snort file to be used asinput...\n\n";
}
$outfile = "$ARGV[0]_alerts.csv";
open(OUTFILE,">$outfile") || die "Can't open $outfile for writing!\n";
$maxlength = $#ARGV;
for (0 .. $maxlength) {
        if (-e "$ARGV[0]") {
                open(INFILE,"$ARGV[0]") || die "Can't open ${ARGV[0]} for reading! Skipping...\n";
                print "\nUsing $ARGV[0] as Input\nSeparating into $outfile.\n";
                print "This may take a while...\n";
                @calendar=qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);
                while (<INFILE>) {
                        if (/^:[0-9]+/) {
                           #it is neither a portscan or an alert - so ignore it
                        } elsif (/spp_portscan\:/) {            # Ignore spp_portscan notifications.
                        } elsif (/ \[\*\*\] /) {                                        # Alert report.
                           #get data
                           ($date_and_time,$alert,$src_and_dst) = split /\s+\[\*\*\]\s/;
                           ($date,$time) = split /-/,$date_and_time ;
                           ($month_number,$day) = split(/\//,$date);
                           $month = $calendar[$month_number-1];
                           ($src,$dst) = split(/\s-\>\s/,$src_and_dst);
                           ($src_host,$src_port) = split(/:/,$src);
                           ($dst_host,$dst_port) = split(/:/,$dst);
                           $alert =~ s/,//g;
                           $snort_entry="ALERT" ;
                           #check everything is there
                            if ((!defined ($month)) or (!defined ($day)) or (!defined ($time)) or (!defined ($alert)) or (!defined
                                        ($src_host)) or (!defined ($dst_host))) {
                                next;
                           }
                           print OUTFILE "$snort_entry,";
                           chomp $month; chomp $day; chomp $time; chomp $alert; chomp$src_host;chomp $dst_host;
                           print OUTFILE "$month,$day,$time,$alert,$src_host,";
                           if (defined($src_port)) {
                             chomp $src_port;  print OUTFILE "$src_port,";
                           } else {
                             print OUTFILE "None,";
                           }
                           print OUTFILE "$dst_host,";
                           if (defined($dst_port)) {
                             chomp $dst_port; print OUTFILE "$dst_port";
                           } else {
                             print OUTFILE "None";
                           }
                           print OUTFILE "\n";
                          } else {
                            #it is neither a portscan or an alert - so ignore it
                          }
                          $numberdone++;
                          print "*" if $numberdone % 1000 == 0;
                }
                print "\n\n";
                close(INFILE) || die "\nCan't close ${ARGV[0]} after finishing!\n";
        } else {
                        print "\nCan't find ${ARGV[0]}! Skipping...\n\n";
        }
        shift (@ARGV);
}

close(OUTFILE) || die "\nCan't close $outfile after finishing!\n";
```

Prepscans.pl

```
#!/cygdrive/c/Perl/bin/perl.exe
# Name: prepscans.pl
# By Terry MacDonald
# Based on csv.pl by Tod Beardsley
# Reads in a Snort scan log
# Usage: prepscans.pl infile [infile2] [infile3] ...
unless ($ARGV[0]) {
  print "You didn't specify the snort scan file to beused as input...\n\n";
}
$outfile = "scan_ss.csv";
open(OUTFILE,">$outfile") || die "Can't open $outfile for writing!\n";
$maxlength = $#ARGV;
for (0 .. $maxlength) {
        if (-e "$ARGV[0]") {
                open(INFILE,"$ARGV[0]") || die "Can't open ${ARGV[0]} for reading! Skipping...\n";
                print "\nUsing $ARGV[0] as Input\nSeparating into $outfile.\n";
                print "This may take a while...\n";
                while (<INFILE>) {
                $_ =~ s/130.85./MY.NET./g ;
                if (/\s\-\>\s/) { # Scans
                        if (/\sUDP\s/) { #UDP scan
                                ($month,$day,$time,$src,$arrow,$dst,$scantype) = split /\s+/;
                                chomp $month; chomp $day; chomp $time; chomp $src_host;
                                chomp $src_port; chomp $dst_host; chomp $dst_port; chomp $scan_type;
                                ($src_host,$src_port)=split (/:/,$src);
                                ($dst_host,$dst_port)=split (/:/,$dst);
                                print OUTFILE "$src_host\n";
                        } else { #TCP scan
                                ($month,$day,$time,$src,$arrow,$dst,$scantype,$flags,$reserved) = split /\s+/;
                                chomp $month; chomp $day; chomp $time; chomp $src_host;
                                chomp $src_port; chomp $dst_host; chomp $dst_port; chomp $scan_type;
                                chomp $flags; chomp $reserved;
                                ($src_host,$src_port)=split (/:/,$src);
                                ($dst_host,$dst_port)=split (/:/,$dst);
                                ($f1,$f2,$f3,$f4,$f5,$f6,$f7,$f8)=split (//,$flags);
                                if (defined ($reserved)){
                                        print OUTFILE "$src_host\n";
                                } else {
                                        print OUTFILE "$src_host\n";
                                }
                        }
                 } else {
                          #it is not a portscan so ignore it
                 }
                 $numberdone++;
                 print "*" if $numberdone % 1000 == 0;
                }
                print "\n\n";
                close(INFILE) || die "\nCan't close ${ARGV[0]} after finishing!\n";
        } else {
                        print "\nCan't find ${ARGV[0]}! Skipping...\n\n";
        }
        shift (@ARGV);
}
close(OUTFILE) || die "\nCan't close $outfile after finishing!\n";
```

Prepoos.pl

```
#!/cygdrive/c/Perl/bin/perl.exe
# Name: prepoos.pl
# By Terry MacDonald
# Based on csv.pl by Tod Beardsley
# Usage: prepoos.pl infile [infile2] [infile3]

unless ($ARGV[0]) {
  print "You didn't specify the snort file to be used as input...\n\n";
}

$outfile = "oos_report.csv";

open(OUTFILE,">$outfile") || die "Can't open $outfile for writing!\n";

$maxlength = $#ARGV;
for (0 .. $maxlength) {
        if (-e "$ARGV[0]") {
                open(INFILE,"$ARGV[0]") || die "Can't open ${ARGV[0]} for reading! Skipping...\n";
                print "\nUsing $ARGV[0] as Input\nProducing into $outfile.\n";
                print "This may take a while...\n";
                while (<INFILE>) {
                if ((/ \-\> /) and (/^[0-9][0-9]\//)) { # on the first line of an OOS record
                    ($date_and_time,$src,$arrow,$dst) = split /\s+/;
                    ($date,$time) = split /-/,$date_and_time ;
                    ($month_number,$day) = split(/\//,$date);
                    $month = $calendar[$month_number-1];
                    ($src_host,$src_port) = split(/:/,$src);
                    ($dst_host,$dst_port) = split(/:/,$dst);
                    $snort_entry="OOS" ;
                    #check everything is there
                    if ((!defined ($month)) or (!defined ($day)) or (!defined ($time)) or (!defined ($src_host)) or (!defined ($dst_host))) {
                            next;
                    }
                    print OUTFILE "$snort_entry,";
                    chomp $month; chomp $day; chomp $time; chomp $src_host; chomp $dst_host;
                    print OUTFILE "$month,$day,$time,$src_host,";
                    if (defined($src_port)) {
                      chomp $src_port;
                      print OUTFILE "$src_port,";
                    } else {
                      print OUTFILE "None,";
                    }
                    print OUTFILE "$dst_host,";
                    if (defined($dst_port)) {
                      chomp $dst_port;
                      print OUTFILE "$dst_port";
                    } else {
                      print OUTFILE "None";
                    }
                    print OUTFILE "\n";
                  } else {
                    #it is not a OOS line - so ignore it
                  }
                  $numberdone++;
                  print "*" if $numberdone % 1000 == 0;
                }
                print "\n\n";
                close(INFILE) || die "\nCan't close ${ARGV[0]} after finishing!\n";
        } else {
                        print "\nCan't find ${ARGV[0]}! Skipping...\n\n";
        }
        shift (@ARGV);
}
close(OUTFILE) || die "\nCan't close $outfile after finishing!\n";
```

References for Assignment 2 and 3

[31] Roesch, Martin et al. "Snort Program". Snort.Org Website. 25 April 2003. URL: http://www.snort.org (25 April 2003).
[32] Anonymous. "Snort Rules Download Page". Snort.Org Website. 25 April 2003. URL: http://www.snort.org/dl/rules (25 April 2003).
[33] Anonymous. "GIAC Certification Practical Logs", Incidents.Org Website. URL: http://www.incidents.org/logs/Raw (14 April 2003).
[34] Anonymous. "IP Info". DShield.Org Website. URL: http://www.dshield.org/ipinfo.php (5 June 2003)
[35] Anonymous. "Google UK". Google Website. URL: http://www.google.co.uk (10 May 2003)
[36] Stinson, Nancy. "AT&T Now - April 2000". AT&T Website. April 2000. URL: http://www.att.com/retirees/attnow/0400/a0004-11.html (6 June 2003)
[37] Anonymous. "Snort Signature Database". Snort.Org Website. 03 May 2003.URL: http://www.snort.org/snort-db/sid.html?sid=499 (03 May 2003)
[38] Mixter. "IDS246 "DOS-LARGE-ICMP"". Whitehats.ca Website. URL: http://www.whitehats.com/info/IDS246 (6 June 2003)
[39] Kettler, Matt. "Re: [Snort-users] Common false positives". Email posting on Snort-users mailing list. 25 February 2003. URL: http://www.pantek.com/library/general/lists/snort.org/snort-users/msg00422.html (6 June 2003)
[40] Anonymous. "BlueIce: Intelligent Computing & Communications ". IBM's Research Website. URL: http://www.research.ibm.com/compsci/communications/projects/infrastructure/blueice.html (6 June 2003)
[41] Giles, Jim. "Jim Giles - Research". IBM Research Website. URL: http://www.research.ibm.com/people/g/gilesjam/research.htm (6 June 2003)
[42] Slemko, Marc. "Path MTU Discovery and Filtering ICMP". 18 January 1998. URL: http://boyan.ludost.net/clueless-faq/pmtu.html (6 June 2003)
[43] Bueno, Pedro. "RES: Large ICMP ping packets". Email posting on incidents@intrusions.org mailing list. 4 April 2002. URL: http://www.incidents.org/archives/intrusions/msg04528.html (6 June 2003)
[44] Daviel, Andrew. Re: Large ICMP ping packets" Email posting on incidents@intrusions.org mailing list. 4 April 2002. URL: http://www.incidents.org/archives/intrusions/msg04521.html (6 June 2003)
[45] MacDonald, Terry. "LOGS: GIAC GCIA Version 3.3 Practical Detect(s)". Email post on intrusions@incidents.org list. 8 May 2003. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/05/msg00063.html (9 May 2003)
[46] Anonymous. "GIAC Certification Practical Logs Readme ". Incidents.Org Website. URL: http://www.incidents.org/logs/Raw/README (14 April 2003).
[47] Cormier, André. "LOGS: GIAC GCIA Version 3.3 Practical Detect(s)". Email post on intrusions@incidents.org list. 16 January 2003. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00121.html (22 April 2003).
[48] Anonymous. "WinDump". WinDump: tcpdump for Windows Website . 8 August 2002. URL: http://windump.polito.it/ (03 May 2003).
[49] Anonymous. "WinDump Manual". WinDump: tcpdump for Windows Website. 8 August 2002. URL: http://windump.polito.it/docs/manual.htm (03 May 2003).
[50] Anonymous. "Public OUI Listing". IEEE.Org Website. 03 May 2003. URL: http://standards.ieee.org/regauth/oui/oui.txt (03 May 2003).
[51] Sanfilippo, Salvatore (Antirez). "Hping Homepage". Hping Website. URL: http://www.hping.org (03 May 2003).
[52] Ricketts, Mike. "SendIp". Project Purple Website. 21 April 2003. URL: http://www.earth.li/projectpurple/progs/sendip.html (03 May 2003).
[53] Anonymous. "CERT® Advisory CA-2003-13 Multiple Vulnerabilities in Snort Preprocessors". CERT® Coordination Center Website. 23 April 2003. URL: http://www.cert.org/advisories/CA-2003-13.html (24 April 2003).
[54] Houghton, Nigel. "Snort Signature Database". Snort.Org Website. 03 May 2003.URL: http://www.snort.org/snort-db/sid.html?sid=523 (03 May 2003)
[55] Anonymous. "RFC791 Internet Protocol DARPA Internet Program Protocol Specification". Internet RFC/STD/FYI/BCP Archives. September 1981. URL: http://www.faqs.org/rfcs/rfc791.html (03 May 2003)
[56] Anonymous. "InterNic WHOIS Database Search". InterNIC Website. 27 October 2001. URL: http://www.internic.net/whois.html (03 May 2003)

[57] Anonymous. "ARIN WHOIS Database Search". ARIN Website. 02 May 2003. URL: http://ws.arin.net/cgi-bin/whois.pl (03 May 2003)
[58] Anonymous. "Yahoo Search Engine". Yahoo Websi te. URL: http://www.yahoo.com (04 May 2003)
[59] Network Working Group. "RFC1141 Incremental Updating of the Internet Checksum". Internet RFC/STD/FYI/BCP Archives. January 1990. URL:http://www.faqs.org/rfcs/rfc1141.html (04 May 2003)
[60] Anonymous. "Who we are". BBN Technologies Website. URL: http://www.bbn.com/about/index.html (04 May 2003)
[61] Udell, Jon. "RFC 1597 Revisited". Byte.Com Website. November 1995. URL: http://www.byte.com/art/9511/sec8/art2.htm (04 May 2003)
[62] Network Working Group. "RFC1166 Internet Numbers". Internet RFC/STD/FYI/ BCP Archives. July 1990. URL: http://www.faqs.org/rfcs/rfc1166.html (04 May 2003)
[63] Network Working Group. "RFC1918 Address Allocation for Private Internets". Internet RFC/STD/FYI/BCP Archives. Febr uary 1996. URL: http://www.faqs.org/rfcs/rfc1918.html (04 May 2003)
[64] Anonymous. "12000 Packet Filter Verification Tests". McAfee ASap Website. http://hq.mcafeeasap.com/vulnerabilities/vuln_data/12000.asp (05 May 2003)
[65] Network Working Group. "RFC1858 Security Considerations for IP Fragment Filtering". Internet RFC/STD/FYI/BCP Archives. October 1995. URL: http://www.faqs.org/rfcs/rfc1858.html (05 May 2003)
[66] Arkin, Ofir. "[Corrected Post] - Using the Unused (Identifying Sun Solaris & HPUX 11.0 OSs)". Email post on bugtraq@securityfocus.org mailing list. 13 August 2000. URL: http://www.sys-security.com/archive/bugtraq/ofirarki n2000-08.txt (05 May 2003)
[67] Miller, Toby. "Passive OS Fin gerprinting: Details and Techniques". Incident.Org Website. URL: http://www.incidents.org/papers/OSfingerprinting.php (05 May 2003)
[68] SWITCH ISP Staff. "Default TTL Values in TCP/IP". URL: http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html (05 May 2003)
[69] Spitzner, Lance. "Lists of fingerprints for passive fingerprint monitoring". 23 May 2000. http://project.honeynet.org/papers/finger/traces.txt (05 May 2003)
[70] Kernel Bug People. "minor mistake in ip -input.c". NetBSD Website. 20 September 1996. URL: http://mail-index.netbsd.org/netbsd-bugs/1996/09/20/0002.html (04 May 2003)
[71] Arkin, Ofir. "Trojan Horse Port List". Sys -Security Group Website. URL:http://www.sys-security.com/html/papers/trojan_list.html (05 May 2003)
[72] Granier, T. Brian. "LOGS: GIAC GCIA Version 3.3 Practical Detect(s)". Email post on intrusions@incidents.org list. 27 Nov 2002. URL:http://cert.uni-stuttgart.de/archive/intr usions/2002/11/msg00256.html (05 May 2003)
[73] Wrisley, Brent. "LOGS: GIAC GCIA Version 3.2 Practical Detect". Email post on intrusions@inciden ts.org list. 4 October 2002. http://cert.uni-stuttgart.de/archive/intr usions/2002/10/msg00079.html (5 May 2003)
[74] Shuck, Ron. "LOGS: GIAC GCIA Version 3.3 Practical D etect(s)". Email post on intrusions@incidents.org list. 10 February 2003. http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00088.html (05 May 2003)
[75] Hough ton, Nigel. "Snort Signature Database". Snort.Org Website. 03 May 2003.URL: http://www.snort.org/snort-db/sid.html?sid=1394 (03 May 2003)
[76] Burns, Brian. "RE: snort: SHELLCODE x86 NOOP". E mail post on focus-ids@securityfocus.org mailing list. 8 April 2002. http://www.derkeiler.com/Mailing-Lists/securityfocus/focus-ids/2002-04/0038.html (9 May 2003)
[77] Graham, Robert David. "RE: snort: SHELLCODE x86 NOOP". Email post on focus - ids@securityfo cus.org mailing list. 8 April 2002. http://www.derkeiler.com/Mailing-Lists/securityfocus/focus-ids/2002-04/0046.html (9 May 2003)
[78] Anonymous. "NTT/Verio Homepage". NTT/Verio Homepage. URL: http://www.verio.net (9 May 2003)
[79] CenterGate Research Group. "GEEKTOOLS: Whois Proxy". GeekTools Website. http://www.geektools.com/cgi-bin/proxy.cgi (9 May 2003)
[80] Infobear. "Infobear's web interface to nslookup". Infobear's Lair. URL: http://www.infobear.com/cgi-bin/nslookup.cgi (9 May 2003)
[81] Ruiu, Dragos. "Re: [Snort-users] SHELLCODE x86 unicode NOOP". Email post on Snort-users@lists.sourceforge.net mailing list. 22 April 2002. URL: http://www.mcabee.org/lists/snort-users/Apr-02/msg00763.html (9 May 2003)
[82] Hamilton, Eric. "JPEG File Int erchange Format". W3C Website. 1 September 2003. URL:

http://www.w3.org/Graphics/JPEG/jfif.txt (10 May 2003)
[83] Lane, Tom. "How do I recognize which file format I have, and what do I do about it?". JPEG Image Compression FAQ. 7 May 2003. URL: http://www.faqs.org/faqs/jpeg-faq/part1/section-15.html (9 May 2003)
[84] Hudak, Tyler. "LOGS: GIAC GCIA Version 3.3 Practical Detect(s)". Email post on intrusions@incidents.org list. 8 March 2003. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/03/msg00105.html (10 May 2003)
[85] Anonymous. "Snort Rules Database". Snort.Org Website. URL: http://www.snort.org/snort-db/ (31 May 2003)
[86] Gordon, Les. "Intrusion Analysis - The Director's Cut!". GCIA Website. 22 November 2002. URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (30 May 2003)
[87] Roesch, Martin. Green, Chris. "Snort Users Manual Snort Release: 2.0.0". Snort.Org website. URL: http://www.snort.org/docs/writing_rules/chap2.html - tth_sEc2.4.4 (27 May 2003)
[88] Network Working Group. "RFC1918 Address Allocation for Private Internets". Internet RFC/STD/FYI/BCP Archives. February 1996. URL: http://www.faqs.org/rfcs/rfc1918.html (04 May 2003)
[89] Whitehats.ca. "arachNIDS database". Whitehats.ca Website. URL: http://www.whitehats.com/ids/ (15 June 2003)
[90] Roesch, Martin. " Re: [Snort-users] Weird fragmentation plugin error". MCABEE Website. 19 April 2001. URL: http://www.mcabee.org/lists/snort-users/Apr-01/msg00540.html (26 May 2003)
[91] Novak, Judy; Northcutt, Stephen. "Detects Analyzed 6/15/00". 15 June 2000. URL: http://www.sans.org/y2k/061500.htm (29 May 2003)
[92] Sage, J. "Two examples of udp:137 netBIOS name table probes". Finchhaven.Com website. 2 March 2002. URL: http://www.finchhaven.com/pages/incidents/030102_udp_137.html (27 May 2003)
[93] Anonymous. "Linux.Adore.Worm" . 15 April 2002. URL: http://securityresponse.symantec.com/avcenter/venc/data/linux.adore.worm.html (29 May 2003)
[94] Anonymous. "Whitehats.ca – Port Query". Whitehats.ca Security Website. 2003. URL: http://www.whitehats.ca/main/tools/portquery2/portquery2.html (29 May 2003)
[95] Anonymous. "Symantec Enterprise Security Solutions protect against the Microsoft Windows IIS Index Server ISAPI System-level Remote Access Buffer Overflow". Symantec Website. 20 June 2001. URL: http://securityresponse.symantec.com/avcenter/security/Content/2001_06_20a.html (29 May 2003)
[96] Anonymous. "StreamWorks Server & Player FAQs:". Real Networks Website. URL: http://docs.real.com/docs/xingtech/StreamWorks%20_Server_FAQs.pdf (30 May 2003)
[97] Anonymous. "RC1 trojan". http://www.glocksoft.com/trojan_list/RC1_trojan.htm (29 May 2003)
[98] Anonymous. "UDP Ports Used by AFS". University of Hohenhiem Website. URL: http://www.rz.uni-hohenheim.de/netzwerkbetriebssysteme/afs36/debug/admin/UDP.html (3 June 2003)
[99] Anonymous. "Patch Available for 'Web Server Folder Traversal' Vulnerability". 17 October 2000. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-078.asp (29 May 2003)
[100] Anonymous. "CERT® Advisory CA-2001-12 Superfluous Decoding Vulnerability in IIS". Cert/CC Website. 15 May 2001. URL: http://www.cert.org/advisories/CA-2001-12.html (30 May 2003)
[101] Anonymous. "Microsoft IIS 4.0 / 5.0 vulnerable to directory traversal via extended unicode in url (MS00-078)". CERT/CC. 18 September 2001. URL: http://www.kb.cert.org/vuls/id/111677 (31 May 2003).
[102] Anonymous. ""Code Red II:" Another Worm Exploiting Buffer Overflow In IIS Indexing Service DLL". Cert/CC Website. 6 August 2001. URL: http://www.cert.org/incident_notes/IN-2001-09.html (29 May 2003)
[103] Anonymous. "CERT® Advisory CA-2001-26 Nimda Worm". Cert/CC Website. 25 September 2001. URL: http://www.cert.org/advisories/CA-2001-26.html (31 May 2003)
[104] Anonymous. "CERT® Advisory CA-2001-11 sadmind/IIS Worm". Cert/CC Website. 10 May 2001. URL: http://www.cert.org/advisories/CA-2001-11.html (30 May 2003)
[105] Beardsley, Tod. "Intrusion Detection and Analysis: Theory, Techniques, and Tools". GCIA.org Website. 8 May 2002. URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc (30 May 2003)
[106] Wilson, Ben. "CGI Script Vulnerability in Microsoft IIS 4.0 and 5.0". GCIA Website. 14 March 2001. URL: http://www.giac.org/practical/gsec/Ben_Wilson_GSEC.pdf (29 May 2003)

[107] Crossman, James. "Re: Think It's Netscape". Intrusions@incidents.org Mailing List. 31 May 2001. URL: http://www.incidents.org/archives/intrusions/msg03817.html (29 May 2003)
[108] Anonymous. "SAFE Nimda *Attack Mitigation*". Cisco Website. 20 September 2001. URL: http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/snam_wp.htm (31 May 2003)
[109] Roesch, Martin. "Re: [snort] Tiny Fragments". Posted to the Snort Users Mailing list. 14 May 2000. URL: http://archives.neohapsis.com/archives/snort/2000-05/0103.html (31 May 2003)
[110] Embrich, Mark. "Intrusion Detection In Depth". GIAC Website. 14 February 2002. URL: http://www.giac.org/practical/Mark_Embrich_GCIA.htm (3 June 2003)
[111] Anonymous. W32.Nimda.A@mm. Symantec Security Response Website. 7 February 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.nimda.a@mm.html (31 May 2003)
[112] Moore, H.D. "Re: x86 NOPS". Posting on the SecurityFocus-IDS mailing list. 23 May 2001. URL: http://archives.neohapsis.com/archives/sf/ids/2001-q2/0398.html (31 May 2003)
[113] Ruiu, Dragos. "Spp_fnord.c". CanSecWest website. URL: http://www.cansecwest.com/spp_fnord.c (31 May 2003)
[114] Ruiu Dragos, "mutants! - spp_fnord.c (It can see the FNORDs! :-)". Posting on the SecurityFocus-IDS mailing list . 1 March 2002. URL: http://cert.uni-stuttgart.de/archive/bugtraq/2002/03/msg00088.html (31 May 2003)
[115] Godin, Jeff. "Re: Probes on UDP port 27015 ". Email Post on intrusions@incidents.org mailing list. 26 December 2000. URL: http://cert.uni-stuttgart.de/archive/incidents/2000/12/msg00134.html (31 May 2003)
[116] Moore, H.D. "Re: X86 NOPS". Email post on the SecurityFocus IDS mailing list. 23 May 2001. URL: http://archives.neohapsis.com/archives/sf/ids/2001-q2/0398.html (9 June 2003)
[117] Stuart, Joe. "Re: [Snort-users] CGI Null Byte Attack". Email Post on Snort users mailing list. 20 November 2000. URL: http://archives.neohapsis.com/archives/snort/2000-11/0244.html (31 May 2003)
[118] Birznieks, Gunther. "Web Application Security – Tying the past and present together". eXtropia Website. URL: http://www.extropia.com/presentations/birznieks/pdf/cgi_security_history.pdf (1 June 2003)
[119] Quazimodorulez. "BitTorrent Websites". Chello Members Webpages. URL: http://members.chello.nl/p.wiersema/list.html (1 June 2003)
[120] Anonymous. "BitTorrent FAQ". BitTorrent Website. http://bitconjurer.org/BitTorrent/FAQ.html (1 June 2003)
[121] Kohlrausch, Jan. "[Advisory] Buffer Overflow in HP-UX Line Printer Daemon - CA-2001-32". Email posting on win-sec-ssc mailing list. 22 November 2001. URL: http://cert.uni-stuttgart.de/archive/win-sec-ssc/2001/11/msg00037.html (1 June 2003)
[122] Havrilla, Jeffrey S. "CERT® Advisory CA-2001-15 Buffer Overflow In Sun Solaris in.lpd Print Daemon". CERT/CC Website. 31 August 2001. URL: http://www.cert.org/advisories/CA-2001-15.html (1 June 2003)
[123] Beriragic, Jamir. "GCIA Certification Practical". GIAC Website. February 2001. URL: http://www.giac.org/practical/Jasmir_Beciragic_GCIA.doc (9 June 2003)
[124] Anonymous. "IDS552 "IIS ISAPI OVERFLOW IDA"". Whitehats.ca Website. URL: http://www.whitehats.com/info/IDS552 (2 June 2003)
[125] Anonymous. "Patch Available for 'Web Server Folder Traversal' Vulnerability". Microsoft Website. 17 October 2000. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-078.asp (2 June 2003)
[126] Caswell, Brian; Houghton, Nigel. "WEB-IIS cmd.exe access". URL: http://www.snort.org/snort-db/sid.html?sid=1002 (2 June 2003)
[127] Hart, Jon. "WEB-IIS CodeRed v2 root.exe access". URL: http://www.snort.org/snort-db/sid.html?sid=1256 (2 June 2003)
[128] Danyliw, Roman; Householder, Alan. "CERT® Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL". 17 January 2002. URL: http://www.cert.org/advisories/CA-2001-19.html (2 June 2003)
[129] Rayford, Joe. "Intrusion Detects and Analysis GCIA Practical Assignment". Giac Website. 2001. URL: http://www.giac.org/practical/Joe_Rayford_GCIA.doc (3 June 2003)
[130] Miller, Toby. "ECN and it's impact on Intrusion Detection". SANS Website. January 2001. URL: http://www.sans.org/y2k/ecn.htm (3 June 2003)
[131] TonkinGin. "XDCC – An .EDU Admin's Nightmare". Russonline Website. 11 September 2002. URL: http://www.russonline.net/tonikgin/EduHacking.html (3 June 2003)

[132] Danyliw, Roman; Householder, Allen. "CERT® Advisory CA-2003-08 Increased Activity Targeting Windows Shares". Cert Website. 11 March 2003. URL: http://www.cert.org/advisories/CA-2003-08.html (3 June 2003)
[133] Ramakrishnan, K.; Floyd, S. "The Addition of Explicit Congestion Notification (ECN) to IP". RFCIndex Website. September 2001. http://rfc-3168.rfc-index.com/rfc-3168.htm (3 June 2003)
[134] Hava, Kristina et. al. "Initial Tool For Monitoring Performance of Websites". Dublin City University Website. URL: http://www.eeng.dcu.ie/~murphyj/publ/c15.pdf (6 June 2003)
[135] Network Working Group. "RFC793 Transmission Control Protocol". Internet RFC/STD/FYI/BCP Archives. September 1981. URL: http://www.faqs.org/rfcs/rfc793.html (04 May 2003)
[136] Anonymous. "Sample Ping Packet Decode". PCAUSA Website. 5 February 2003. URL: http://www.pcausa.com/resources/ndispacket_decode.htm (9 May 2003)
[137] Anonymous. "Snort(TM) Advisory: Integer Overflow in Stream4". Snort.Org Website. 16 April 2003. URL: http://www.snort.org/advisories/snort-2003-04-16-1.txt (27 May 2003)