# Incident Analysis

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 4.0

Joel Esler

September 2004

## Table of Contents

# Abstract

This paper contains one detect taken from logs retrieved from the GIAC Sans website. All raw data contained in this report has been pre-cleansed by SANS before release. This report is presented as if it were being given to a "University", no references to the author are noted except for the title page.

The report contains such details as "Top Five" offending IP's, "Top Five" most sought after ports, and correlations with other GCIA Practicals found to be containing data from the same log files.

Finally, recommendations to improve security in the "University's" network are also made. These recommendations are made without respect to current security status, as that status was not provided.

## Executive Summary

Analysis of Intrusion Detection System (IDS) logs can prove vital to evidence recovery and the tracking of an unauthorized user inside of the network, as well as attempts to gain access to the network. The University employs the Snort IDS recording malicious traffic inbound and outbound from its network. Without the use of an IDS, the proper configuration, and an in-depth look at its logs, an intruder can gain full access to the University's computer systems, while remaining undetected. This would allow them knowledge of internal network layout and security of the network. The worst-case scenario for the University is the loss and manipulation of data or misuse of the systems for the storage of illegal files and attacks against others.

Using the data provided, this report is presented with the following limitations:

1. The data is being analyzed "after-the-fact". In an instance where the data has already been stolen, it is likely that little can be done to recover the information. However, the possibility to involve law enforcement may be of assistance during the recovery process.
2. All data being analyzed was provided under the security in place at the time of the attack. If changes have been made to the network to strengthen or weaken the security, then these changes are not reflected in the report.
3. The analysis is provided to promote awareness and to help the prevention of this type of activity in the future. Recommendations are made once the results are presented.

After reading the report, the University will have a better understanding of its network health and security. Several different attacks are covered in the following pages, giving an overview of the vast amount of attacks targeting the network. The recommendations made will not cover funding availability or feasibility, as they are simply suggestions about how to manage the network, in addition to protecting it against future attacks.

**Bottom Line up Front** – Review of the University's IDS logs can and will prove vital to prevent the unauthorized use of the network and to ensure the security therein. Some incidents were identified from the logs provided and a few will be discussed in detail, overall the health of the network from the logs provided is moderate to good. Many false positives were noted in the logs, this is a common result found with many IDS's, and in some cases cannot be fully prevented. The highest attack noted throughout these logs is for "Squid Proxy Server". It is unknown if this is present inside the University's network. The placement of the IDS may have hindered the ability to observe internal attacks; it may be beneficial to place a second IDS inside the network. No evidence of compromised machines was found inside the log files provided.

## Detailed Analysis

In order to tune the IDS, some preliminary analysis must take place on the logs provided to establish the University's network setup and IP. Since this was not supplied using the log file, the following command was ran:

```
Tcpdump –nn –r /2002.9.29 –e | less
```

### Tcpdump and Tcpdump Tag Explanation

Tcpdump is a common tool used for recording, analyzing, and playing back network traffic. [1]

-nn tells Tcpdump *not* to print out the "Domain Name Qualification" of host names and not to attempt to "name" the ports. (Example: port 80 would be translated as "http" without this option). [1]

-r tells Tcpdump to "read packets from *file*" which is specified as /2002.9.29. [1]

-e instructs Tcpdump to "print the link-level header on each dump line". [1]

-w lets Tcpdump know to "write recorded packets to *file.*" (While "-w" is not being used in this particular example, it will be referenced later). [1]

The command "tcpdump –nn –r /2002.9.29 –e | less" produced the following output:

```
20:00:15.916507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4
(0x0800),length 321: IP 148.63.214.239.2314 > 32.245.214.229.9511: P
2944716298:2944716565(267) win 8192

20:01:19.916507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4
(0x0800),length 321: IP 148.63.214.239.2314 > 32.245.214.229.9511: P
2944716298:2944716565(267) win 8192

20:01:33.956507 00:00:0c:04:b2:33 > 00:03:e3:d9:26:c0, ethertype IPv4
(0x0800),length 382: IP 32.245.166.236.63421 > 64.154.80.48.80: P
2582873670:2582873998(328) ack 945786467 win 17520
```

### Establishment of Network Setup

The MAC (or Media Access Control) addresses 00:03:e3:d9:26:c0 and 00:00:0c:04:b2:33 need to be translated to their vendors in order to establish the network setup. By using http://coffer.com/mac_find/, a website that is maintained and operated by Jason@coffer.com, will allow the input of MAC addresses to reveal which vendor they are assigned to. In this case, both MAC addresses belong to *Cisco Systems, Inc*. [2] [3]

Many network setups are logically similar. In this case, the network being monitored may be arranged with a router or firewall as the outermost device, the IDS in the middle, with the next security device being a Cisco Router or Firewall. Since the internal MAC appears to be drastically different than the external, this is likely a separate device such as a *PIX* firewall.

The graphic below represents this setup:

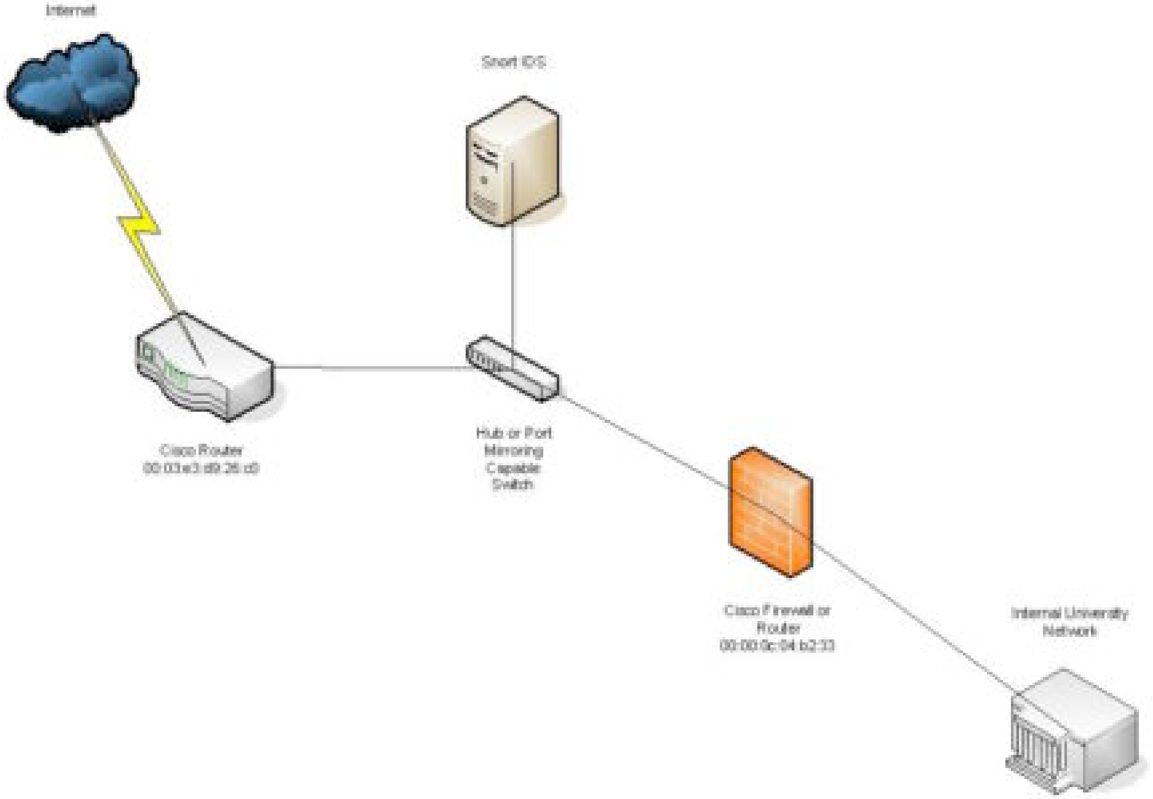


**Figure 1**

Now that the basic layout of the network is established, IP ranges used internally or alongside the IDS must be defined. Due to the fact that most IDS security devices sit in the "DMZ" (or ***De****militarized* ***Z****one* [4]), it is possible that other "publicly needed" boxes, such as a webserver or "DNS" server, may reside on this "unprotected" network.

Since the entire collection of packets from these two files share a common network "32.245.0.0/16" it can be assumed that this network is used by the University.

The University used an unknown version of Snort as the IDS. In order to have an accurate playback of the log files, the most current version of Snort will be utilized. The Snort version referred to in this case was “2.1.3”, which was compiled on a Darwin kernel (Apple). The command line options were setup as seen below:

```
Snort –c /snort-2.1.3/etc/snort.conf –r /2002.9.29 –l /snort-2.1.3/Sep/ -h
32.245.0.0/26 –k none
```

## Snort and Snort Tag Explanation

Snort is an open source IDS which can be found at http://www.snort.org [10]. “Snort is the world's most widely deployed open source intrusion-detection system…a package that can perform protocol analysis, handle content searching and matching, and detect a variety of attacks and probes” -- [5]

-c informs Snort which configuration file to use. (Snort man page)

-r tells Snort which *file* to read. Snort has the capability of reading back tcpdump files created with the tcpdump “-w” option. In this case, the file name is “2002.9.29”. (Snort man page)

-l instructs Snort to log to *directory*, in this case “/snort-2.1.3/Sep” is being used. (Snort man page)

-h sets up Snort so the Home network variable is set to “32.245.0.0/16”. (Snort man page) As Snort understands CIDR (or ***C**lassless* ***I**nter*-***D**omain* ***R**outing*) notation, Snort is instructed to understand that the University’s network is everything from 32.245.0.0 to 32.245.255.255. An IP number has four 8-bit octets. Since each binary bit has two possible values, either on or off (0 or 1), each octet can represent $2^8$ = 256 (which converts to 0..255 as a decimal).[6] It is possible in this occasion to have 65,536 IP’s or $2^{16}$ = 65536.

-k advises the Checksum mode for Snort to use, “none” in this example. (Snort man page)

## Snort statistics generated using Snort Stat

After running Snort in “readback” mode against the data provided by the University, we have the following statistics:

Snort processed 8827 packets
TCP: 8824 Packets
UDP: 3 Packets

Snort generated:
218 Alerts

Snort produced a large number of alerts. In order to separate the false positives from the actual attempts, a tool called "*Snort Stat*" will be able to pin point the top offenders. *Snort Stat* is a perl program written by Yen-Ming Chen (chenym@Alumni.cmu.edu). The current version of Snort stat distributed with Snort 2.1.3 is version 1.4 written March 20, 2003.

Using the command below produces these statistics:

```
cat /snort-2.1.3/Sep/alert | /snort-2.1.3/contrib/snort_stat.pl > /Sepoutput
```

Cat is a command to "concatenate and print files" included with every Unix, Linux, and BSD distribution.  It will read files that have been selected and output those files to "standard output". Currently, the output is being sent over to snort stat through the "pipe" command "|", to be processed.

Snort stat gives us the following statistics:

## *Overall Statistics*

*The log begins from: 10 28 20:04:03*
*The log ends     at: 10 28 19:59:11*
*Total events: 218*
*Signatures recorded: 12*
*Source IP recorded: 19*
*Destination IP recorded: 170*

## Top Ten Alerts

The Top Ten alerts being:

```
========================================================================
 # of
attacks  from            to              method
========================================================================
  4    207.200.85.50    32.245.166.236   SHELLCODE x86 NOOP
  4    64.12.168.18     32.245.166.236   SHELLCODE x86 NOOP
  3    172.178.254.113  32.245.118.118   SCAN Squid Proxy attempt
  3    172.178.254.113  32.245.127.118   SCAN Squid Proxy attempt
  3    172.178.254.113  32.245.121.118   SCAN Squid Proxy attempt
  3    81.48.106.177    32.245.117.118   SCAN Squid Proxy attempt
  3    172.178.254.113  32.245.117.118   SCAN Squid Proxy attempt
  3    172.178.254.113  32.245.126.118   SCAN Squid Proxy attempt
  3    81.48.106.177    32.245.118.118   SCAN Squid Proxy attempt
  3    172.178.254.113  32.245.122.118   SCAN Squid Proxy attempt
```

## *Filtering Out the Majority of False Positives*

Now that general statistics have been determined, the focus can be geared toward high profile IP's. In order to properly assess the network, an additional review of the 2002.9.29 file was completed with tcpdump again, as described below:

tcpdump –nn –r /2002.9.29 | less

```
20:00:15.916507 IP 148.63.214.239.2314 > 32.245.214.229.9511: P 2944716298:29447
16565(267) win 8192
20:01:19.916507 IP 148.63.214.239.2314 > 32.245.214.229.9511: P 2944716298:29447
16565(267) win 8192
20:01:33.956507 IP 32.245.166.236.63421 > 64.154.80.48.80: P 2582873670:25828739
98(328) ack 945786467 win 17520
<truncated>
```

## Establishment of NAT Address

From the review of the tcpdump output seen above, as well as data that has been truncated for brevity, one can surmise that the type of traffic coming and going from the network being monitored, 32.245.166.236 is most likely a NAT (**N**etwork **A**ddress **T**ranslation) or PAT (**P**ort **A**ddress **T**ranslation) address of the firewall beyond the IDS. Using the tcpdump statement "tcpdump –nn –r /2002.9.29 host 32.245.166.236" as a filter shows the following (excerpts have been taken for brevity):

```
20:01:33.956507 IP 32.245.166.236.63421 > 64.154.80.48.80: P 2582873670:25828739
98(328) ack 945786467 win 17520
20:01:33.966507 IP 32.245.166.236.63421 > 64.154.80.48.80: P 2657880093:26578815
53(1460) ack 1637087204 win 33580
20:01:34.196507 IP 32.245.166.236.63421 > 64.154.80.48.80: P 2657881889:26578822
16(327) ack 1637087533 win 33580
<truncated>
21:01:29.926507 IP 208.184.29.233.80 > 32.245.166.236.63467: P 2077563228:207756
4688(1460) ack 3236528313 win 2920
23:29:11.946507 IP 32.245.166.236.61269 > 66.70.56.245.80: P 4036100144:40361005
43(399) ack 4046914030 win 16835
01:34:16.336507 IP 128.167.120.13.80 > 32.245.166.236.62833: . 895861965:8958634
25(1460) ack 313387 win 24820
03:30:35.456507 IP 32.245.166.236.62354 > 207.68.167.253.6667: P 89699782:896997
93(11) ack 3543345906 win 8641
03:30:51.276507 IP 32.245.166.236.62410 > 216.136.173.152.80: P 1851844318:18518
47238(2920) ack 1419273347 win 64240
<truncated>
```

The traffic indicated above and throughout the filter applied, indicates that all traffic coming from the 32.245.166.236 is outbound traffic. Also the traffic coming to it is return traffic, in other words, there is no specifically targeted attack to this IP. Therefore, this IP will be eliminated but considered for later research. Not that NAT addresses are less important than the rest of the addresses, however usually a device that will produce a “NAT” or “PAT” is a “deny all, permit by exception” firewall.

This traffic can also be examined for signs of “violation of corporate policy” issues, such as the below packet:

```
03:30:35.456507 IP 32.245.166.236.62354 > 207.68.167.253.6667: P 89699782:896997
93(11) ack 3543345906 win 8641
```

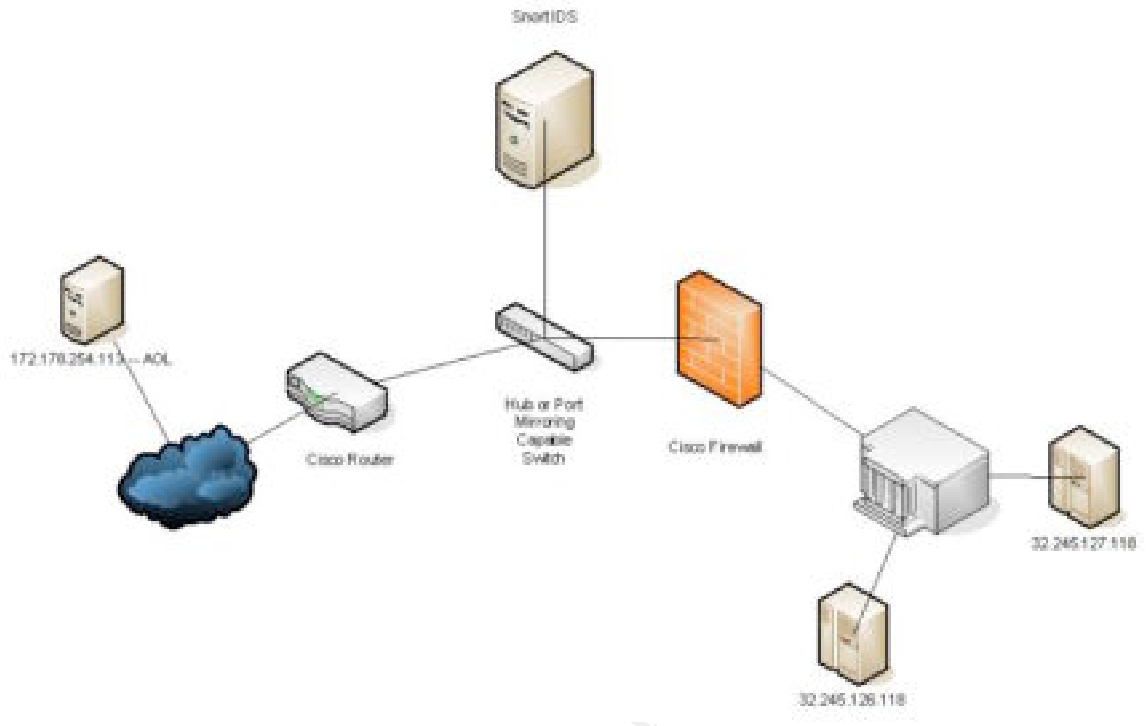
This may be an indication of IRC usage, (against many corporate policies, but, not usually against University policies). Yet, for initial analysis the tcpdump filter: “tcpdump –nn –r /2002.9.29 not host 32.245.166.236 | less” will be used.

This PAT or NAT analysis is in agreement with *Andrew Evans, GCIA*, as he came to the same conclusion in his practical.

# Network Detect 1

## *Description of Attack*

The “attack” or scan under investigation is a simple probe looking for open port 3128 on several hosts. Port 3128 is commonly used for “Squid Proxy Server” [7]. The Squid proxy Server allows other users to use the server to download files, surf the web, or hide attacks. This way, the user being attacked is unaware of the hacker’s true identity. Since the scanner in this attack is jumping, looking for one particular IP .118 on several Class B’s, Squid Proxy Servers on the University’s network already may be evident to the attacker.



**Link Graph 1**

## Reason this Detect was Selected

The reason this particular detect was selected from the logs is because it shows:

a) Prior reconnaissance
b) Knowledge of internal networking

Instead of being sequential, the attacker only targets IP's that end with .118 in the network. This may show that the intruder is either aware of a server already present, or believes that most Squid Proxy servers in the world reside on IP's ending with .118.

## Source of Trace

The source of the trace being used throughout the report, were the log files at http://isc.giac.org/RAW/2002.9.28 and http://isc.giac.org/RAW/2002.9.29. It is evident that the date indicated by the name of the log, in addition to the timestamp of the data inside the log are approximately a month off. This will be explained in Appendix 1.

Going back to the original configuration, with the –h variable defined as 32.245.0.0/16, a few alerts were generated right away.

```
[**] [1:618:8] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/28-20:04:03.216507 172.178.254.113:1956 -> 32.245.126.118:3128
TCP TTL:115 TOS:0x0 ID:32600 IpLen:20 DgmLen:48 DF
******S* Seq: 0xDDCEE1E4  Ack: 0x0  Win: 0x7FFF  TcpLen: 28
TCP Options (4) => MSS: 1360 NOP NOP SackOK

[**] [1:618:8] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/28-20:04:06.076507 172.178.254.113:1956 -> 32.245.126.118:3128
TCP TTL:115 TOS:0x0 ID:32696 IpLen:20 DgmLen:48 DF
******S* Seq: 0xDDCEE1E4  Ack: 0x0  Win: 0x7FFF  TcpLen: 28
TCP Options (4) => MSS: 1360 NOP NOP SackOK

[**] [1:618:8] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/28-20:04:12.136507 172.178.254.113:1956 -> 32.245.126.118:3128
TCP TTL:115 TOS:0x0 ID:32857 IpLen:20 DgmLen:48 DF
******S* Seq: 0xDDCEE1E4  Ack: 0x0  Win: 0x7FFF  TcpLen: 28
TCP Options (4) => MSS: 1360 NOP NOP SackOK

[**] SCAN Squid Proxy attempt [**]
10/28-21:04:41.726507 172.178.254.113:3991 -> 32.245.127.118:3128
TCP TTL:115 TOS:0x0 ID:10481 IpLen:20 DgmLen:48 DF
******S* Seq: 0x7CE289B0  Ack: 0x0  Win: 0x7FFF  TcpLen: 28
TCP Options (4) => MSS: 1360 NOP NOP SackOK

[**] SCAN Squid Proxy attempt [**]
10/28-21:04:44.686507 172.178.254.113:3991 -> 32.245.127.118:3128
TCP TTL:115 TOS:0x0 ID:10572 IpLen:20 DgmLen:48 DF
******S* Seq: 0x7CE289B0  Ack: 0x0  Win: 0x7FFF  TcpLen: 28
TCP Options (4) => MSS: 1360 NOP NOP SackOK

[**] SCAN Squid Proxy attempt [**]
10/28-21:04:50.686507 172.178.254.113:3991 -> 32.245.127.118:3128
TCP TTL:115 TOS:0x0 ID:10749 IpLen:20 DgmLen:48 DF
******S* Seq: 0x7CE289B0  Ack: 0x0  Win: 0x7FFF  TcpLen: 28
TCP Options (4) => MSS: 1360 NOP NOP SackO
```

## Using p0f to Establish Source IP Operating System

P0f is “a versatile passive OS fingerprinting tool.” [11] A command line interface that has the capability of reading tcpdump written files as well as sniffing traffic off of an interface. P0f analyzes traffic read or sniffed for a variety of features, most importantly being operating system. Using the command line options included with p0f, the program was run with the command line option “-s” instructing p0f to “read packets from tcpdump snapshot “. The following command line and the resulting output are below:

```
#  p0f -s 2002.9.28
p0f - passive os fingerprinting utility, version 2.0.4
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on '2002.9.28', 223 sigs (12 generic), rule: 'all'.
<truncated>
172.178.254.113:3178 - Windows XP SP1, 2000 SP4 (3)
  -> 32.245.117.118:3128 (distance 13, link: (Google/AOL))
172.178.254.113:3178 - Windows XP SP1, 2000 SP4 (3)
  -> 32.245.117.118:3128 (distance 13, link: (Google/AOL))
172.178.254.113:3178 - Windows XP SP1, 2000 SP4 (3)
  -> 32.245.117.118:3128 (distance 13, link: (Google/AOL))
172.178.254.113:1513 - Windows XP SP1, 2000 SP4 (3)
  -> 32.245.118.118:3128 (distance 13, link: (Google/AOL))
172.178.254.113:1513 - Windows XP SP1, 2000 SP4 (3)
  -> 32.245.118.118:3128 (distance 13, link: (Google/AOL))
<truncated>
```

This indicates to the analyst that the *America Online* IP 172.178.254.113 was running either Windows 2000 SP4 or Windows XP SP1. Due to the fact that Windows 2000 SP4 was not released until June 26, 2003. The offending must be Windows XP SP1 (released September 9, 2002).

## The Detect

The rule that caught this traffic was:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"SCAN Squid Proxy attempt"; flags:S,12; flow:stateless; classtype:attempted-recon; sid:618; rev:8;)
```

This rule is looking for only the “SYN” flag set, ignoring reserve bit 1 and 2 (*flags:S,12;*) on a connection from an external host to an internal host (*$EXTERNAL_NET any -> $HOME_NET*) on port 3128. It was observed that the 172.178.254.113 address was attempting to connect to 32.245.126.118, and twenty-nine seconds later attempted to scan 32.245.127.118. This indicates to an analyst that the 172.178.254.113 address is attempting to service-scan one class C subnet at a time looking for an open port of 3128. This is probably an effort to avoid detection from an IDS, like *RealSecure* [9] which looks for Service Scans across whole Class C subnets searching for the same port.

## Probability of Spoof

It was not likely the Source IP was spoofed, because if the attacker *was* able to find an IP with port 3128 open, then the response of "SYN, ACK" would not have returned to the attacker. Although this is a simple "SYN" scan, and there are a number of scans looking for port 1080 or port 3128 throughout this log file. It is possible that the "SYN, ACKs" returning to IP 172.178.254.113 could be to a compromised machine or a machine to look for the "SYN, ACK's". Therefore it's possible that the Source IP could be spoofed, but it is unlikely.

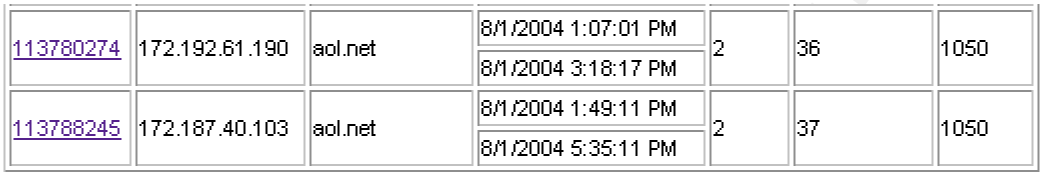
```
nslookup 172.178.254.113

Name:    ACB2FE71.ipt.aol.com
Address:  172.178.254.113

OrgName: America Online
OrgID: AOL
Address: 22000 AOL Way
City: Dulles
StateProv: VA
PostalCode: 20166
Country: US
```

## Attack Mechanism

In this detect, not enough information is available to see whether or not the attacker was successful in their reconnaissance. The Snort IDS is only setup to log traffic that fires off a rule, making it difficult to determine if any of the hosts responded. A way to correct that action would be to add the modifier "tag:session;" or "tag:host;" to the rule. "Tagging" allows the Snort IDS to record responses and further traffic after a rule was set off. However if access was provided to router logs, firewall logs, or the logs on the actual hosts, then it would be possible to see whether or not the attack succeeded. As previously stated, a hacker may find the use of a Proxy server handy to hide their connections as well as redirect all traffic. The attacker may have used a scanning tool such as nmap, setup to use to the –sS (Syn Scan) option. Coupled with a script to change the IP every run (in order to scan only .118 IP's) it could be an extremely useful tool.

## Correlations

Incidents.org and mynetwatchman.com both still show current activity scanning for port 3128. Most of it is attributed to the RingZero Trojan. RingZero scans for three ports: 80, 8080, and 3128. From the logs provided the scans for port 80 or 8080 are not seen from the AOL IP. This could indicate that the attack is *not* the RingZero Trojan and rather, a more specific attack method looking for port 3128 on the University's Subnets. The two IP's pasted below are both from the same ISP as the Intruder, AOL.com and showing activity similar to that of the 172.178.254.113 IP.



| | | | | | | |
|---|---|---|---|---|---|---|
| 113780274 | 172.192.61.190 | aol.net | 8/1/2004 1:07:01 PM<br>8/1/2004 3:18:17 PM | 2 | 36 | 1050 |
| 113788245 | 172.187.40.103 | aol.net | 8/1/2004 1:49:11 PM<br>8/1/2004 5:35:11 PM | 2 | 37 | 1050 |

Looking back at the log files from the previous day, the same exact activity is apparent. The 2002.9.28 log shows the 32.245.117.118 – 32.245.125.118 subnets being scanned from the AOL IP. This is most likely a `slow and low scan' looking for port 3128 on the destination University networks.

### Evidence of Active Targeting

It is possible to assume that some prior reconnaissance has been done on the network, due to the fact that the attacker is looking for port 3128 on IP's ending with .118. Whether the scouting has been from an insider, third party, or from a different compromised box this fact is not revealed in the log files provided. Given that the scan is crossing 10 subnets, it is hard to be convinced that the attacker knows anything about the internal network. At this time it is not believed that the attacker knew what they were looking for and were randomly scanning IP's ending with .118 in order to evade an IDS.

### Severity

Severity is calculated with the following formula (Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity. Each value is rated on a scale of 1 (lowest) to 5 (highest).

### Criticality

Not many production networks run the Squid Proxy for legitimate use. Usually this type of system is not deemed "mission critical", therefore, the Criticality level of this "non-existent" Proxy should be assigned a **2**.

### Lethality

*If* a proxy server would have even existed for the attacker to use, then there may have been a problem. Once an attacker has access to a box on the inside of a network, it is possible to use that internal box as a staging point to attack the rest of the network. In a case where additional boxes, were compromised it would appear as if the proxy server on the inside of the network was the source. The focus would then be on an insider as opposed to strengthening perimeter defenses. A lethality of **4** is being assigned to this category.

### System Countermeasures

There is no evidence to support that the scan detected any servers, let alone being able to use one. The patch level of the system, or whether or not it allowed "external-to-network" connections is unknown. Since there is no evidence that a Squid Proxy server exists, a System Countermeasure rating of **4** is being given.

### Network Countermeasures

Obviously the connection was allowed through the external router or it would not have been recorded on the IDS. Again we have no evidence to support the belief that it was allowed through the firewall because no return connections are recorded.  The Network Countermeasure rates a **4**.

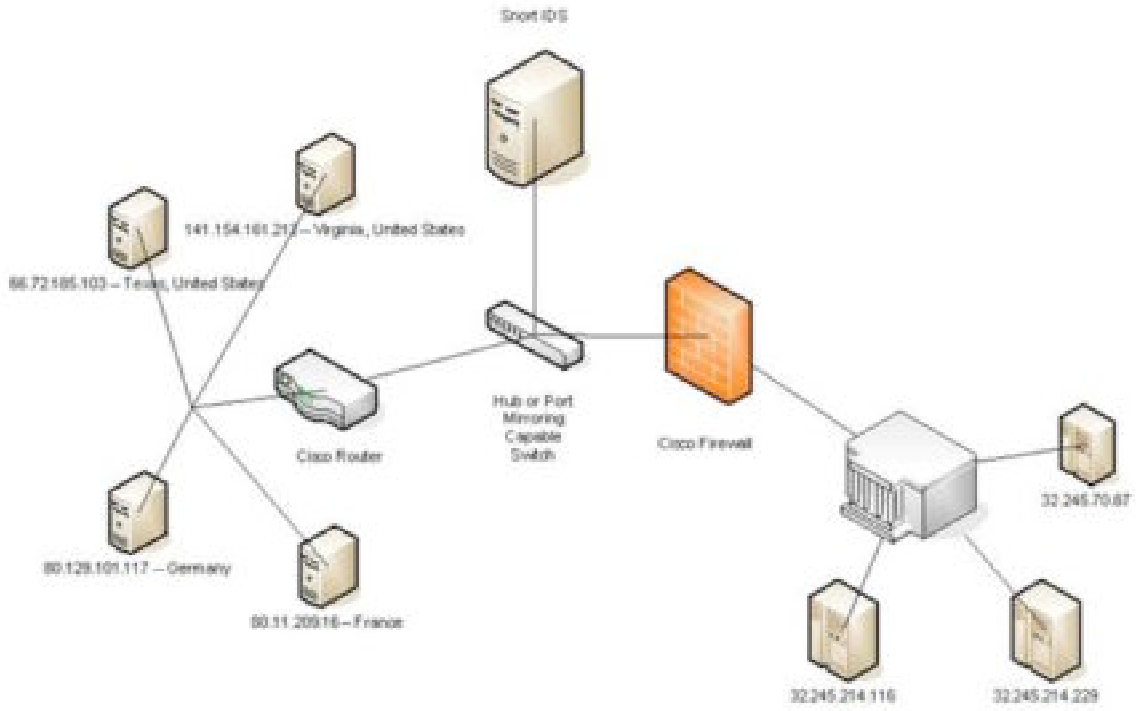
(**2** + **4**) – (**4** + **4**) = **0**.

The formula above indicates that this scan has virtually no impact on the University's network, aside from the possible knowledge of the internal layout.

## Network Detect 2

### *Description of Attack*

The attack under investigation in Detect #2 is an anomalous packet crossing the network setup.  It is believed that the attack could be an attempt to map the network or an attempt to bypass security devices like firewalls or IDS's.  In this detect, the focus is placed on 4 individual packets.  While all 4 packets are from different sources and have similar signatures between them, they are all different.



**Link Graph 2**

## Reason this Detect was Selected

This Detect was selected for its uniqueness in comparison to the other packets in the 2002.9.28 and 2002.9.29 log files. The four packets analyzed here are distinctive, mainly because it appears that the attack targets no particular server or port and does not contain any recognizable payload. Each of the four packets are fragmented to the point where a firewall or IDS may let the traffic slip through, only to be reassembled on the distant machine. Fragmentation is becoming less and less common due to high bandwidth networks, more reliable equipment, and more efficient TCP stack implementations in software. The packets used in this detect are neither the first fragmented packet nor the last packet in the two cases with the “More Fragments” bit set.

## Source of Trace

The Snort alerts for this detect were produced from the log files at http://isc.giac.org/RAW/2002.9.28 and http://isc.giac.org/RAW/2002.9.29 which were used in Network Detect 1. Snort was ran without the “-h” option to ensure that all rules will fire, regardless of the direction of traffic. The following alerts were generated:

```
[**] [1:523:5] BAD-TRAFFIC ip reserved bit set [**]
[Classification: Misc activity] [Priority: 3]
10/28-18:57:44.976507 66.72.185.103 -> 32.245.214.229
TCP TTL:237 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x1D8E   Frag Size: 0x0014
```

```
[**] [1:522:2] MISC Tiny Fragments [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
10/28-09:28:36.456507 80.11.209.16 -> 32.245.214.116
TCP TTL:231 TOS:0x0 ID:0 IpLen:20 DgmLen:40 MF
Frag Offset: 0x02CB   Frag Size: 0x0014

[**] [1:523:5] BAD-TRAFFIC ip reserved bit set [**]
[Classification: Misc activity] [Priority: 3]
10/28-21:02:05.186507 141.154.161.212 -> 32.245.214.229
TCP TTL:232 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0177   Frag Size: 0x0014

[**] [1:522:2] MISC Tiny Fragments [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
10/29-11:33:52.726507 80.129.101.117 -> 32.245.70.87
TCP TTL:244 TOS:0x0 ID:0 IpLen:20 DgmLen:40 MF
Frag Offset: 0x1E28   Frag Size: 0x0014
```

## The Detect

Analysis of the four packets seen above shows that two different rules were used to detect the traffic.

```
Alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC ip
reserved bit set"; fragbits:R; classtype:misc-activity; sid:523; rev:5;)
```

This particular rule is looking for an "External-net" host that is inbound to "home-net" with the reserve bit set. It is important to remember that Snort is configured not to use home-net and external-net distinction.

```
Alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Tiny
Fragments"; dsize:<25; fragbits:M; classtype:bad-unknown; sid:522; rev:2;)
```

This rule is also looking for an "External-net" host that is inbound to "home-net". The "dsize:<25" keyword is used to test the packet payload size.[10] In this case, the rule is searching for a packet less than 25 bytes long. The "fragbits:" keyword is also used here to check if the "More Fragments" bit is set in the IP header. [10]

## Probability of Spoof

The probability of spoof is somewhat unlikely. All four of the packets have the "ACK" flag set, various fragment offsets, and different source ports. What the attacker was hoping to accomplish is unknown at this time. If the packet was a part of a whole fragmented piece of traffic, then Snort would have reassembled this traffic. This is assuming that the frag2 preprocessor was configured on the

University's IDS that is recording this inbound traffic. Using only an ACK flag in a packet, it is possible to slip past routers that are configured with a statement similar to "permit tcp any any established". The "established" keyword [12] is used to "permit return connections on ports that are based on an established connection". A way to bypass this access control on filtering routers or firewalls is to not use a lone "SYN" packet. By attempting to act like the connection is already established, an attacker can effectively bypass the filtering capability of the router. Given the similarities of the packets, the attackers may be using a common tool. If it is an attempt to gather information regarding the destination hosts (open ports, tcp stack replies), then the attack would need responses from the destination hosts to make these packets useful.

## Attack Mechanism

***Packet 1***

The way the packets are assembled is quite odd, therefore, all packets will be examined individually:

```
18:57:44.976507 IP 66.72.185.103 > 32.245.214.229: tcp
      0x0000:  4500 0028 0000 9d8e ed06 279f 4248 b967  E..(......'.BH.g
      0x0010:  20f5 d6e5 8004 2527 9cb6 6592 9cb6 6592  ......%'..e...e.
      0x0020:  5004 0000 fd80 0000 0000 0000 0000       P............
```

| | |
|---|---|
| TOS: 0 | Total Length: 40 |
| Reserved Bit is set | Frag Offset: 60528 |
| SrcIP: 66.72.185.103 | DestIP: 32.245.214.229 |
| SourcePort: 32772 | DestPort: 9511 |
| Seq#: 0x9cb66592 | Ack#: 0x9cb66592 |
| ACK Flag | Window Size: 0 |
| Checksum: 0xfd80 (incorrect) | |

### IP Header

Src IP: 66.72.185.103 -- Tyson Motors, Plano, Texas, United States
In this packet, the "reserve bit" is present which may be an indication of "Explicit Congestion Notification" (ECN), a new TCP/IP protocol. [14] Most hardware and software do not support ECN at this time. ECN was even more rare at the writing of this dump file in 2002. Since this is a fragmented transmission, one would expect that a "More Fragments" bit would exist. Fragment Offset is 60528, but there is nothing to indicate that there is any other traffic before or after this packet. Therefore, this offset should not be present.

### TCP Header

The Source Port could be an RPC port, but review of the Destination Port at www.incidents.org reveals nothing of significance. Sequence Number and Acknowledgement numbers are the same. This is a similarity that is found throughout these packets. Window size in this packet is set to 0, and the packet's checksum is 0xfd80 (incorrect). The checksum does not add up in these log files because SANS has modified the destination IP in order to protect the identity of the network.

***Packet 2***

```
09:28:36.456507 IP 80.11.209.16 > 32.245.214.116: tcp
        0x0000:  4500 0028 0000 22cb e706 8269 500b d110  E..(.."....iP...
        0x0010:  20f5 d674 8271 008b 7c1c 1520 7c1c 1520  ...t.q..|...|...
        0x0020:  5004 0000 dbcf 0000 0000 0000 0000       P............
```

```
TOS: 0                  Total Length: 40
More Fragments is set                     Frag Offset:  5720
SrcIP:  80.11.209.16             DestIP:  32.245.214.116
SourcePort: 33373                DestPort:  139
Seq#:  0x7c1c1520                Ack#: 0x7c1c1520
                        ACK Flag    Window Size: 0
Checksum: 0xdbcf (incorrect)
```

**IP Header**
Source IP: 80.11.209.16 -- IP2000-ADSL-BAS -- France
Fragment offset is 5720. Yet, without the surrounding data to indicate if this is a lone packet or a part of a larger whole, the idea that a fragment offset exists is irrelevant. The "More Fragments" bit is set in the IP header in this packet, however, there are no further packets to analyze (Snort would have reassembled them using the Frag2 preprocessor).

**TCP Header**
While the Source Port 33393 does not indicate much information, the destination port 139 has a number of interesting vulnerabilities for it. 139 is the port that samba uses for it's netbios communications as well as being the official Microsoft "NETBIOS Session Service" port. Multiple CVE's can be found in relation to this port, however, the traffic shown does not lead to any in particular. Seq and Ack numbers are again the same.

***Packet 3***

```
21:02:05.186507 IP 141.154.161.212 > 32.245.214.229: tcp
        0x0000:  4500 0028 0000 8177 e806 14f7 8d9a a1d4  E..(...w........
        0x0010:  20f5 d6e5 8046 2527 0028 ed80 0028 ed80  .....F%'.(...(..
        0x0020:  5004 0000 f2bf 0000 0000 0000 0000       P............
```

```
TOS: 0                  Total Length: 40
Reserve Bit is set               Frag Offset:  3000
SrcIP:  141.154.161.212          DestIP:  32.245.214.229
```

```
SourcePort: 10240                     DestPort:  9511
Seq#:  0x0028ed80                     Ack#: 0x0028ed80
                        ACK Flag      Window Size: 0
Checksum:  0xf2bf (incorrect)
```

**IP Header**
Src IP:  141.154.161.212 -- Verizon Internet Services, Reston, Virginia, United States
Fragment Offset is 3000, and again no Fragment bit is set.  This packet has the Reserve Bit set like Packet 1.  This is not the only similarity between Packet 3 and Packet 1.  Both are attempting to communicate to IP:  32.245.214.229.

**TCP Header**
The similarity between Packet 1 and 3 does not end with the IP. It also continues with destination port: 9511.   At this point, a closer review of the destination IP is warranted.  While running tcpdump with the following command line "tcpdump –nnr /2002.0.29 host 32.245.214.219", it is observed that a total of 67 packets in this file alone are generated, and each of them have the destination port of 9511. Examination of the traffic from this IP indicates that this is Gnutella traffic as observed in Appendix 2.  Source port 32838 does not yield any significant results at www.incidents.org.

***Packet 4***

```
11:33:52.726507 IP 80.129.101.117 > 32.245.70.87: tcp
      0x0000:  4500 0028 0000 3e28 f406 574e 5081 6575  E..(..>(..WNP.eu
      0x0010:  20f5 4657 8143 0f1b 6132 f846 6132 f846  ..FW.C..a2.Fa2.F
      0x0020:  5004 0000 3b36 0000 0000 0000 0000       P...;6........
```

```
TOS: 0                  Total Length: 40
More Fragments bit is set                 Frag Offset:  61760
SrcIP:  80.129.101.117                DestIP:  32.245.70.87
SourcePort: 33091                     DestPort:  3867
Seq#:  0x6132f846                     Ack#: 0x6132f846
                        ACK Flag      Window Size: 0
Checksum:  0x3b36 (incorrect)
```

**IP Header**
Src IP:  80.129.101.117 -- Deutsche Telekom AG, Germany
Reverting back to the "More Fragments" bit again, the fragment offset is currently being set at 61760.

**TCP Header**
Neither Source port or Destination Port in this packet gives us any clear indications as to their use, however, reported attempts to port 3867 did jump to 2482 on August 27,2004. http://isc.sans.org//port_details.php?port=3867&repax=1&tarax=2&srcax=2&percent=N&days=40.

## Correlations

There were many other GCIA analysts that observed the same activity. Several were found with interesting comments. For instance, *Mario Ricci, GCIA,* states that "An E-mail from Jeff Oxenreider describes his finding GNUTELLA as generating these types of packets" referring to the "MISC Tiny Fragments" packets. The same result was found in the analysis of packets 1 and 3.

## Evidence of Active Targeting

## Severity

Severity is calculated with the following formula (Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity. Each value is rated on a scale of 1 (lowest) to 5 (highest).

## Criticality

Since two of the packets (2 and 4) can not be linked to any indications of further traffic, and packets 1 and 3 can be linked to Gnutella activity, it is possible that these are stray P2P packets. Since P2P clients like Gnutella are mostly likely installed on workstations or desktops, the criticality of these packets is **3**.

## Lethality

The damage suffered to the machines by these individual packets is negligible since this traffic is most likely related to P2P type activity. Even if a link to P2P could not be proven, (assuming that these packets passed through the firewall and made it to the destination) the most likely response that could be extracted would be an ICMP Port Unreachable or similar message. Lethality is being placed at a **2**.

## System Countermeasures

There is no evidence to support that the packet was received and processed by the destination IP. Since one of the Destination IP's actually can be confirmed as having more traffic going to port 9511, the odds of the packet being received by the Destination IP are very possible. Most likely this would not cause much malicious traffic. However, P2P applications should not be run on production networks. System Countermeasures is being placed at **3**.

## Network Countermeasures

It is obvious that the connection was allowed through the external router, otherwise, it would not have been recorded on the IDS. There are other indicators that the packets going to the destination IP address were being received since GNUTELLA activity can be shown on the same port. Since this traffic is occurring on a non-Gnutella standard port, it may already be blocked. However, this traffic should not be permitted into the network. Given that the packet is not well formed, a state-ful firewall should block this activity. The Network Countermeasure rates a **4**.

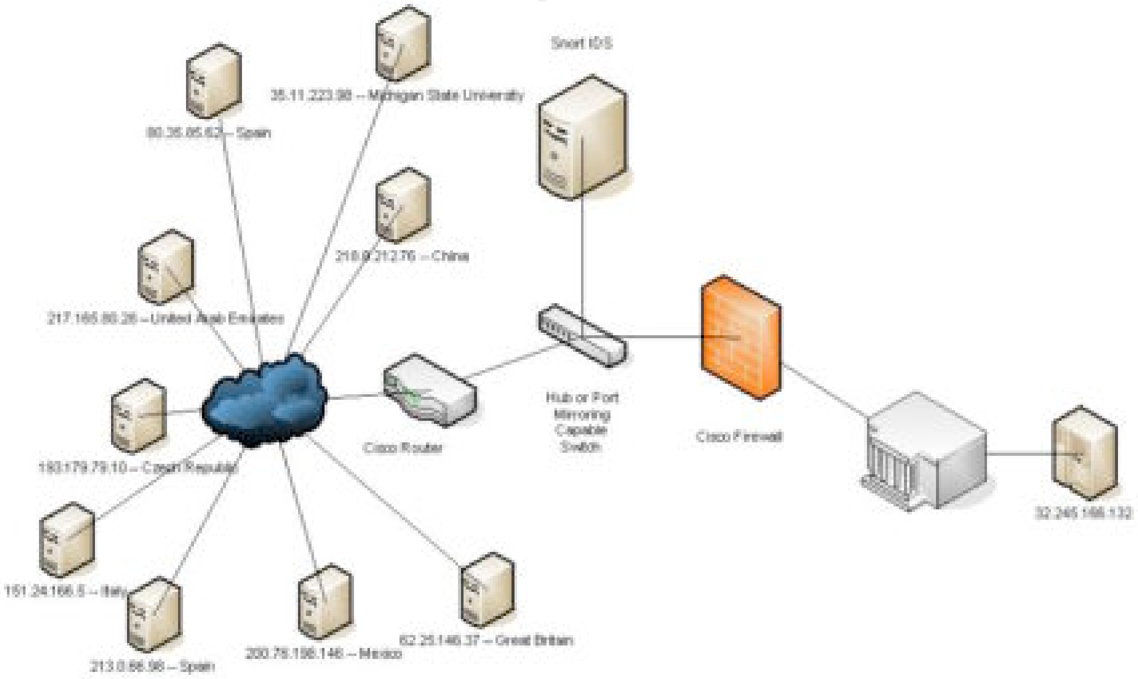
(**3**+ **2**) – (**3** + **4**) = **-2**.

The formula above indicates that this scan has virtually no impact on the University’s network, aside from the existence of P2P traffic on the network.

# Network Detect 3

## *Description of Attack*

The Attack being reviewed in Detect #3 is an attempt for a total of nine different machines to map to drive “C” on Destination IP: 32.245.166.132, netbios name B2B. All attacks are specifically targeted to this IP, and no attempts to contact any other machine are observed.


Snort IDS
35.11.223.98 – Michigan State University
80.35.85.62 – Spain
218.9.212.76 – China
217.165.85.26 – United Arab Emirates
193.179.79.10 – Czech Republic
Cisco Router
Hub or Port Mirroring Capable Switch
Cisco Firewall
32.245.166.132
151.24.166.5 – Italy
213.0.86.98 – Spain
200.78.198.146 – Mexico
62.25.146.37 – Great Britain

**Link Graph 3**

## Reason this Detect was Selected

The fact that there are so many (9) connections to this one machine either leads one to believe that this machine is being specifically targeted by a geographically wide-spread attack or perhaps a configuration on the firewall that allows connections to this destination IP over specific ports.  This is most likely a purposeful configuration, simply because it is the only machine being contacted. If this is truly a purposeful configuration, then the amount of boxes connecting to it from all over the world may be related to worm activity.

## Source of Trace

The source of the trace being used in this detect, were the log files at http://isc.giac.org/RAW/2002.9.28 and http://isc.giac.org/RAW/2002.9.29.  This detect was found through analysis of tcpdump and ethereal data.  Snort rules did not trigger on this activity and is further explained below in “The Detect” section of Detect 3.

The tcpdump logs are as follows for these events:

```
20:11:35.246507 IP (tos 0x0, ttl 111, id 65325, offset 0, flags [DF], length: 99, bad cksum 2d68 (->4280)!)
35.11.223.98.2644 > 32.245.166.132.139: P
[bad tcp cksum c4de (->d9f6)!] 2551575:2551634(59) ack 2522361066 win 8756
        0x0000:  4500 0063 ff2d 4000 6f06 2d68 230b df62  E..c.-@.o.-h#..b
        0x0010:  20f5 a684 0a54 008b 0026 ef17 9658 2cea  .....T...&...X,.
        0x0020:  5018 2234 c4de 0000 0000 0037 ff53 4d42  P."4.......7.SMB
        0x0030:  7500 0000 0000 0000 0000 0000 0000 0000  u...............
        0x0040:  0000 0000 0000 0000 0000 0000 04ff 0000  ................
        0x0050:  0000 0001 000c 0021 5c5c 4232 425c 4300  .......!\\B2B\C.
        0x0060:  413a 00                                  A:.
23:16:16.776507 IP (tos 0x0, ttl 108, id 679, offset 0, flags [DF], length: 99, bad cksum 89fb (->9f13)!) 80.35.85.62.1685 >
32.245.166.132.139: P [ba
d tcp cksum b4f2 (->ca0a)!] 2501320:2501379(59) ack 981122511 win 8756
        0x0000:  4500 0063 02a7 4000 6c06 89fb 5023 553e  E..c..@.l...P#U>
        0x0010:  20f5 a684 0695 008b 0026 2ac8 3a7a bdcf  .........&*.:z..
        0x0020:  5018 2234 b4f2 0000 0000 0037 ff53 4d42  P."4.......7.SMB
        0x0030:  7500 0000 0000 0000 0000 0000 0000 0000  u...............
        0x0040:  0000 0000 0000 0000 0000 0000 04ff 0000  ................
        0x0050:  0000 0001 000c 0021 5c5c 4232 425c 4300  .......!\\B2B\C.
        0x0060:  413a 00                                  A:.
04:45:10.176507 IP (tos 0x0, ttl 113, id 23090, offset 0, flags [DF], length: 99, bad cksum a90f (->be27)!)
217.165.80.28.10032 > 32.245.166.132.139:
P [bad tcp cksum dfe8 (->f500)!] 992832809:992832868(59) ack 1591430164 win 8756
        0x0000:  4500 0063 5a32 4000 7106 a90f d9a5 501c  E..cZ2@.q.....P.
        0x0010:  20f5 a684 2730 008b 3b2d 6d29 5edb 4c14  ....'0..;-m)^.L.
        0x0020:  5018 2234 dfe8 0000 0000 0037 ff53 4d42  P."4.......7.SMB
        0x0030:  7500 0000 0000 0000 0000 0000 0000 0000  u...............
        0x0040:  0000 0000 0000 0000 0000 0000 04ff 0000  ................
        0x0050:  0000 0001 000c 0021 5c5c 4232 425c 4300  .......!\\B2B\C.
        0x0060:  413a 00                                  A:.
04:52:50.776507 IP (tos 0x0, ttl 106, id 25382, offset 0, flags [DF], length: 99, bad cksum 2288 (->37a0)!)
218.8.212.76.1890 > 32.245.166.132.139: P
[bad tcp cksum 82e6 (->97fe)!] 254453:254512(59) ack 1705640928 win 8756
        0x0000:  4500 0063 6326 4000 6a06 2288 da08 d44c  E..cc&@.j."....L
        0x0010:  20f5 a684 0762 008b 0003 e1f5 65aa 03e0  .....b......e...
        0x0020:  5018 2234 82e6 0000 0000 0037 ff53 4d42  P."4.......7.SMB
        0x0030:  7500 0000 0000 0000 0000 0000 0000 0000  u...............
        0x0040:  0000 0000 0000 0000 0000 0000 04ff 0000  ................
        0x0050:  0000 0001 000c 0021 5c5c 4232 425c 4300  .......!\\B2B\C.
        0x0060:  413a 00                                  A:.
07:32:47.436507 IP (tos 0x0, ttl  42, id 21060, offset 0, flags [DF], length: 99, bad cksum 3102 (->461a)!)
193.179.47.10.2070 > 32.245.166.132.139: P
```

```
 [bad tcp cksum f9b4 (->ecd)!] 26265677:26265736(59) ack 4091339581 win 4820
        0x0000:  4500 0063 5244 4000 2a06 3102 c1b3 2f0a  E..cRD@.*.1.../.
        0x0010:  20f5 a684 0816 008b 0190 c84d f3dc e33d  ...........M...=
        0x0020:  5018 12d4 f9b4 0000 0000 0037 ff53 4d42  P..........7.SMB
        0x0030:  7500 0000 0000 0000 0000 0000 0000 0000  u...............
        0x0040:  0000 0000 0000 0000 0000 0000 04ff 0000  ................
        0x0050:  0000 0001 000c 0021 5c5c 4232 425c 4300  .......!\\B2B\C.
        0x0060:  413a 00                                  A:.
11:10:50.016507 IP (tos 0x0, ttl 107, id 23675, offset 0, flags [DF], length: 99, bad cksum 47a9 (->5cc1)!)
200.76.198.146.1045 > 32.245.166.132.139:
P [bad tcp cksum 425b (->5773)!] 3477079967:3477080026(59) ack 3048014740 win 16612
        0x0000:  4500 0063 5c7b 4000 6b06 47a9 c84c c692  E..c\{@.k.G..L..
        0x0010:  20f5 a684 0415 008b cf40 079f b5ad 0394  .........@......
        0x0020:  5018 40e4 425b 0000 0000 0037 ff53 4d42  P.@.B[.....7.SMB
        0x0030:  7500 0000 0000 0000 0000 0000 0000 0000  u...............
        0x0040:  0000 0000 0000 0000 0000 0000 04ff 0000  ................
        0x0050:  0000 0001 000c 0021 5c5c 4232 425c 4300  .......!\\B2B\C.
        0x0060:  413a 00                                  A:.

09:34:56.096507 IP (tos 0x0, ttl 108, id 53509, offset 0, flags [DF], length: 99, bad cksum 499b (->5eb3)!)
213.0.66.98.27223 > 32.245.166.132.139: P
[bad tcp cksum a782 (->bc9a)!] 18117604:18117663(59) ack 1618249079 win 8756
        0x0000:  4500 0063 d105 4000 6c06 499b d500 4262  E..c..@.l.I...Bb
        0x0010:  20f5 a684 6a57 008b 0114 73e4 6074 8577  ....jW....s.`t.w
        0x0020:  5018 2234 a782 0000 0000 0037 ff53 4d42  P."4.......7.SMB
        0x0030:  7500 0000 0000 0000 0000 0000 0000 0000  u...............
        0x0040:  0000 0000 0000 0000 0000 0000 04ff 0000  ................
        0x0050:  0000 0001 000c 0021 5c5c 4232 425c 4300  .......!\\B2B\C.
        0x0060:  413a 00                                  A:.
11:12:55.266507 IP (tos 0x0, ttl 110, id 59389, offset 0, flags [DF], length: 99, bad cksum ae8 (->2000)!)
151.24.166.5.1025 > 32.245.166.132.139: P [
bad tcp cksum 7154 (->866c)!] 92406781:92406840(59) ack 3079606431 win 8756
        0x0000:  4500 0063 e7fd 4000 6e06 0ae8 9718 a605  E..c..@.n.......
        0x0010:  20f5 a684 0401 008b 0582 03fd b78f 109f  ................
        0x0020:  5018 2234 7154 0000 0000 0037 ff53 4d42  P."4qT.....7.SMB
        0x0030:  7500 0000 0000 0000 0000 0000 0000 0000  u...............
        0x0040:  0000 0000 0000 0000 0000 0000 04ff 0000  ................
        0x0050:  0000 0001 000c 0021 5c5c 4232 425c 4300  .......!\\B2B\C.
        0x0060:  413a 00                                  A:.
13:13:51.936507 IP (tos 0x0, ttl 112, id 39105, offset 0, flags [DF], length: 99, bad cksum c503 (->da1b)!)
62.25.146.37.1813 > 32.245.166.132.139: P
[bad tcp cksum b935 (->ce4d)!] 680021682:680021741(59) ack 588658447 win 17516
        0x0000:  4500 0063 98c1 4000 7006 c503 3e19 9225  E..c..@.p...>..%
        0x0010:  20f5 a684 0715 008b 2888 4eb2 2316 370f  ........(.N.#.7.
        0x0020:  5018 446c b935 0000 0000 0037 ff53 4d42  P.Dl.5.....7.SMB
        0x0030:  7500 0000 0000 0000 0000 0000 0000 0000  u...............
        0x0040:  0000 0000 0000 0000 0000 0000 04ff 0000  ................
        0x0050:  0000 0001 000c 0021 5c5c 4232 425c 4300  .......!\\B2B\C.
        0x0060:  413a 00                                  A:.
```

## The Detect

The Snort rule that is included with the standard Snort ruleset that normally would detect this activity is:

Alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:”NETBIOS SMB C$ share access”; flow:to_server,established; content:”|00|”; depth:1; content:”|FF|SMBu”; depth:5; offset:4; byte_test:1,<,128,6,relative; content:C|24 00|”; distance:32; nocase; classtype:protocol-command-decode; sid:533; rev:8;)

In order to detect the traffic seen above, the rule must be modified. This built-in rule is looking for “C$”, and the traffic shown in tcpdump is \\B2B\C.  The rule is re-written:

Alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB C share access"; content:"|00|"; depth:1; content:"|FF|SMBu"; depth:5; offset:4; byte_test:1,<,128,6,relative; content:C"; distance:32; nocase; classtype:protocol-command-decode; rev:9)

Take notice that the hex string "|24 00|", ASCII = "$.", and the "flow" modifier have been removed from the original rule. Since we are doing analysis on playback of logs, Snort cannot establish a "flow". See Appendix 2 for further discussion. When Snort is re-ran we get the following result:

```
[**] [1:0:9] NETBIOS SMB C share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
10/29-09:34:56.096507 213.0.66.98:27223 -> 32.245.166.132:139
TCP TTL:108 TOS:0x0 ID:53509 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x11473E4  Ack: 0x60748577  Win: 0x2234  TcpLen: 20

[**] [1:0:9] NETBIOS SMB C share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
10/29-11:12:55.266507 151.24.166.5:1025 -> 32.245.166.132:139
TCP TTL:110 TOS:0x0 ID:59389 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x58203FD  Ack: 0xB78F109F  Win: 0x2234  TcpLen: 20

[**] [1:0:9] NETBIOS SMB C share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
10/29-13:13:51.936507 62.25.146.37:1813 -> 32.245.166.132:139
TCP TTL:112 TOS:0x0 ID:39105 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x28884EB2  Ack: 0x2316370F  Win: 0x446C  TcpLen: 20

[**] [1:0:9] NETBIOS SMB C share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
10/27-20:11:35.246507 35.11.223.98:2644 -> 32.245.166.132:139
TCP TTL:111 TOS:0x0 ID:65325 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x26EF17  Ack: 0x96582CEA  Win: 0x2234  TcpLen: 20

[**] [1:0:9] NETBIOS SMB C share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
10/27-23:16:16.776507 80.35.85.62:1685 -> 32.245.166.132:139
TCP TTL:108 TOS:0x0 ID:679 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x262AC8  Ack: 0x3A7ABDCF  Win: 0x2234  TcpLen: 20

[**] [1:0:9] NETBIOS SMB C share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
10/28-04:45:10.176507 217.165.80.28:10032 -> 32.245.166.132:139
TCP TTL:113 TOS:0x0 ID:23090 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x3B2D6D29  Ack: 0x5EDB4C14  Win: 0x2234  TcpLen: 20

[**] [1:0:9] NETBIOS SMB C share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
10/28-04:52:50.776507 218.8.212.76:1890 -> 32.245.166.132:139
```

```
TCP TTL:106 TOS:0x0 ID:25382 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x3E1F5  Ack: 0x65AA03E0  Win: 0x2234  TcpLen: 20

[**] [1:0:9] NETBIOS SMB C share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
10/28-07:32:47.436507 193.179.47.10:2070 -> 32.245.166.132:139
TCP TTL:42 TOS:0x0 ID:21060 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x190C84D  Ack: 0xF3DCE33D  Win: 0x12D4  TcpLen: 20

[**] [1:0:9] NETBIOS SMB C share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
10/28-11:10:50.016507 200.76.198.146:1045 -> 32.245.166.132:139
TCP TTL:107 TOS:0x0 ID:23675 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0xCF40079F  Ack: 0xB5AD0394  Win: 0x40E4  TcpLen: 20
```

## Probability of Spoof

The likelihood of this attack being spoofed is relatively low. Seeing as these are established connections (Flags in TCP packets are ACK+PSH), they are communications already in progress. Sequence and Acknowledgement numbers are all varying as are Source Ports. While it is possible to spoof this traffic, an expectation like all traffic coming from the same Source Port would be observed. If the traffic were spoofed, the attacker would not be able to see the consequences of their actions. Since an SMB session is being established it is necessary to have a three-way handshake.

## Attack Mechanism

All of the packets listed above have the same attack mechanism. They all share common destination ports, and the exact same payload traffic "!\\B2B\C". This indicates to an analyst that either this is a common mis-configuration, or this is the only box that is allowed to receive SMB stimulus. Perhaps the best explanation of this traffic is found at https://www.europe.f-secure.com/v-descs/opasoft.shtml .

←truncated→

1. sets a connection with the \\hostname\C resource, where "hostname" = the name of the victim computer which is defined when the victim computer answers Opasoft (by sending its "reply data") during the scan

2. if the resource is password-protected the worm runs through all possible "one symbol" passwords - conducting a "brute-force" attack

←truncated→

Since the password used in this transaction was "!", (the first printable character in the ASCII "alphabet") it is possible this traffic could be related to the Opasserv worm. "!" is being used as the brute-force password in the initial negotiation stages of the worm's infection method. However, this worm only affects Microsoft Windows 95 through ME, so the target host may not be vulnerable.

## Correlations

http://securityresponse.symantec.com/avcenter/venc/data/w32.opaserv.worm.html -- The Symantec website describing the worm.
https://www.europe.f-secure.com/v-descs/opasoft.shtml -- The best explanation found of the worm.
*Daniel Wesseman, GCIA* also correlated this activity to the Opaserv worm in much the same method. His rewrite of the Snort rule is also similar to the one found here. His practical can be found at: http://www.giac.org/practical/GCIA/Daniel_Wesemann_GCIA.pdf.

## Evidence of Active Targeting

As previously stated, the only targeted machine was the 32.245.166.132 IP. This is most likely a result of the configuration of the *PIX* firewall to allow port 139 traffic through to the destination. It is unknown why the firewall is configured in such a way. Perhaps this is a public file-sharing server statically translated through the *PIX* firewall. The ACL on the firewall may need to be reviewed to see if this traffic is necessary.

## Severity

Severity is calculated with the following formula (Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity. Each value is rated on a scale of 1 (lowest) to 5 (highest).

## Criticality

It is unknown what the actual function of the destination machine is. However, it does have NETBIOS sharing turned on. As previously stated, the IP is statically translated. So it is possible that the machine could be a publicly needed machine such as a fileserver. Criticality is rated at a **4**.

## Lethality

It is believed that this traffic is the opening negotiation of the Opaserv worm from September 2002. If the connection were to be successful, (which it is not believed to be) the worm propagation would take place. Such an act would infect the machine, and possibly infect others on the same network. It may also cause a bandwidth issue depending on the amount of scanning the virus conducts on the networks. Lethality is rated at a **4**.

## System Countermeasures

The virus appears to *not* have infected the machine. Had the virus been able to propagate to the machine, one would observe the machine scanning outbound in a similar pattern. It is unclear what the patch level or Operating System of the machine is, and the presence of antivirus is not known. However, the virus did not infect the destination machine. System Countermeasures are being placed at **4**.

### Network Countermeasures

It is obvious that the traffic was permitted through all routers and firewalls since this rule actually triggers during on-going sessions. The packet capture observed here proves that fact with flags set at "ACK+PSH". Network Countermeasures is rated at a **1**.

(**4** + **4**) – (**4** + **1**) = **3**

The formula above indicates that this possible virus may have an impact on the University's network. The evidence does support the theory that the box most likely is not infected.

## *Network Statistics*

### Top Talkers

The chart below is directly from Snort Stat's output.

```
==================================================================
      #  of
  %   attacks   from            method
==================================================================
36.70    80      255.255.255.255   BACKDOOR Q access
32.11    70      216.77.219.159    SCAN SOCKS Proxy attempt
15.14    33      172.178.254.113   SCAN Squid Proxy attempt
 2.75    6      81.48.106.177     SCAN Squid Proxy attempt
 1.83    4      64.12.168.18      SHELLCODE x86 NOOP
```

### Top Five Targeted Services or Ports

The following chart is directly from the analysis of the Snort Logs and Snort Stat output

```
==================================================================
      #  of
  %   attacks  from            method
==================================================================
36.70          515          255.255.255.255    Backdoor Q access
32.11          1080         216.77.219.159     SCAN SOCKS Proxy attempt
15.14          3128         172.178.254.113    SCAN Squid Proxy attempt
2.75           3128         81.48.106.177      SCAN Squid Proxy Attempt
1.83           63651        64.12.168.18       SHELLCODE x86 NOOP
```

## Profile of the Three Most Suspicious External Source Addresses

1. **172.178.254.113** – This AOL IP is most likely a home user scanning the University's network specifically for boxes ending with .118. This person may have knowledge about the network, either from being an insider, and knowing the internal network layout or knows what some IDS's trigger on, and how to bypass them.

2. **68.36.170.9** -- This IP belonging to Comcast Cable Communications out of New Jersey attempts to use a FrontPage extension vulnerability against a web server at IP 32.245.166.119. There are many efforts to use this vulnerability against this same IP, (32.245.166.119 appears to be a mail server with a webpage front end) however, the Comcast IP seems to be the most persistent, causing a total of 56 packets of data to be sent from 68.36.170.9 to the web server.

3. **35.11.223.98** – This IP belonging to Michigan State University attempted to connect to IP: 32.245.166.132 using SMB over port 139. The traffic indicates an attempted connection to the share \\B2B\C. While it is one of 9 IP's that attempts to do this during the duration of the packet capture, it is the only American IP. All other IP's come from locations as close as Mexico, to as far away as China. This could be legitimate traffic, such as a mistyped share (ex. \\B2B\C instead of \\B2B\C$) but, with the data provided, it is impossible to know.

## *Correlations from GCIA Practicals*

*Andrew Evans, GCIA* is just one of the many analysts that drew the same conclusion as I did concerning NAT addressing and the PIX Firewall theory. This can be found in his practical at http://www.giac.org/practical/GCIA/Andrew_Evans_GCIA.pdf.

*Blaine Hein, GCIA* and *Erik Montcalm, GCIA* were the only other practicals written containing analysis from the 2002.9.28 and 2002.9.29 log files. Although Blaine and Erik did not use the same detects as I did, Blaine and I both drew the same conclusion regarding the NAT addressing and PIX Firewall theories. While Blaine's practical is not yet published on the GCIA website, his Version 3.4 Practical Detect posts can be found at http://www.dshield.org/pipermail/intrusions/2004-May/007964.php and http://www.dshield.org/pipermail/intrusions/2004-April/007910.php. Erik's practical can be found on the GCIA website at http://www.giac.org/practical/GCIA/Erik_Montcalm_GCIA.pdf.

### *Insights into machines*

It's quite obvious from these two files that this is probably a network that may not be managed locally. Many of the things associated with the network come from a lack of systems security personnel and/or lack of specific security policy on the network. Spyware (an IP behind the 32.245.166.236 firewall reported to Gator several times), P2P (32.245.214.229 is receiving inbound connections from the Gnutella network), IRC and Instant Messaging (both can be found in the '2002.9.29' log file) can also be seen on this network. No specific evidence can be found of compromised or infected machines, while there is a great deal of poor security practices and possible misconfigurations, there is no concrete evidence to support an intrusion in these two log files.

### *Defensive Recommendations*

1. Correct the time on the remote sensors and master collector, if this is how the University's IDS's logs are configured. Suggestion: Set all clocks to GMT (Greenwich Mean Time, also known as "Zulu" time).

2. Place all machines possible behind the NAT firewall, to ensure proper security.

3. Remove all IRC (found in '2002.9.29'), P2P (found in both '2002.9.28' and '2002.9.29'), instant messaging (specifically MSN Messenger, found in file '2002.9.29'), and spyware (Found in '2002.9.29' and '2002.9.28') inside of the network.

4. Block well-known ports that are not part of the services needed on the network (3128, 8080, and 1080 for example if you do not run proxy servers) if possible.

5. Establish a policy on the outer-most router of deny all, permit by exception.

6. Establish a policy of "least privilege" for all personnel and networks. This will keep outsiders and insiders from attaining unauthorized access to the computers and files, allowing them to change and/or erase any data they choose. Removing "Administrator" privileges from common users will keep them from installing unauthorized applications such as P2P.

## Analysis Process

All data provided was analyzed using:

1. Snort version 2.1.3 from the Snort.org website with the default ruleset implemented. The only changes made to the Snort IDS were to configure the snort.conf file to include all rulesets.

2. Tcpdump version 3.8-cvs.

3. Snort Stat version 1.4

4. Ethereal version 0.10.3 [13]

5. All programs, as well as analysis and reporting, were compiled on an Apple iBook G3 900 MHz PowerPC processor with 256k of RAM running OS X 10.3.5 “Panther”. The report was written using Microsoft Office’s Word 2004 for the OS X platform.

6. Graphics were generated using Microsoft Visio on a Microsoft Windows XP Professional computer running Microsoft Office 2003.

7. Commands used for the above programs are found in-line with the detects as well as in the beginning of the practical.

8. One major problem discussed through the paper was the “flow” rule modifier, which allows Snort figure out the direction in which traffic is flowing. This will be discussed in depth in Appendix 2.

# Appendix 1

The files that were provided were dated September 28, 2002 and September 29, 2002, (file names were 2002.9.28, 2002.9.29) however, the data contained inside the files covered a time range from 10/27/2002 19:11 to 10/29/2002 14:18. This indicates to the analysts, that the Snort process is killed every day at about 7:00 pm (19:00) and restarted again, logging to a different tcpdump binary file. If this is done via a script, it could possibly set the date on the logs using a command such as:

“set date = `date ‘+%Y.%m.%d’` “

Assuming there is one box collecting the logs and several different sensors actually recording the logs on different networks within the University, the dates and times on the “Master” Collector and the “Remote” Sensors are drastically off by approximately a month. The logs reviewed in this case are from a sensor monitoring the 32.245.0.0/16 network from October 27, 2002 to October 29, 2002.

The process is stopped everyday at 1900, it could be as simple as University policy, or it could be attributed to a series of events. These events may include:

1. The Master collector runs the script to login to the remote sensors.
2. Kill their Snort processes.

3. Starts a new Snort process, changing the name of the output file by using the above date command.
4. Copies the files back to the master collector.

If this is the case and assuming the script is ran at midnight, then the network being reviewed may be located in Hawaii.   HST (Hawaiian Standard Time) is a 6-hour difference from EST (Eastern Standard Time) when EST is *not* on Day Light Savings time.  Since Day Light savings time was in effect at the time of recording these logs (September 2002), the time difference is only 5 hours. HST does not observe Day Light Savings Time (2400 hours – 1900 hours = 5 hour difference).

# Appendix 2

## Application of Tcpdump filter

An immediate application of the tcpdump filter "tcpdump –nn –r /2002.9.29 no host 32.245.166.236 | less" shows these four packets at the beginning:

```
20:00:15.916507 IP 148.63.214.239.2314 > 32.245.214.229.9511: P 2944716298:29447
16565(267) win 8192
20:01:19.916507 IP 148.63.214.239.2314 > 32.245.214.229.9511: P 2944716298:29447
16565(267) win 8192
20:02:23.916507 IP 148.63.214.239.2314 > 32.245.214.229.9511: P 2944716298:29447
16565(267) win 8192
20:03:27.926507 IP 148.63.214.239.2314 > 32.245.214.229.9511: P 2944716298:29447
16565(267) win 8192
```

While the source port and destination port do not immediately pop out as anything malicious, a review of the ports at http://www.incidents.org/port_details.php?port=2314 and http://www.incidents.org/port_details.php?port=9511 indicate high amounts of traffic. On the right hand side of the www.incidents.org page, some ports are mapped to specific programs. In this case studying these fields yield no conclusive results. However, in correlation with Detect 2, port 9511 in our log files has been attributed to GNUTELLA activity, as seen below.

## How to view ASCII Text using Tcpdump

Tcpdump is re-ran with the following statement: "tcpdump –nn –r /2002.9.29 –X not host 32.245.166.236 | less"

-X in tcpdump tells the program to "print each packet (minus its link level header) in hex and ASCII."  -- [1]

It is immediately clear when looking at the following:

```
20:00:15.916507 IP 148.63.214.239.2314 > 32.245.214.229.9511: P 2944716298:29447
16565(267) win 8192
```

```
0x0000:  4500 0133 f9af 4000 7106 96f3 943f d6ef  E..3..@.q....?..
0x0010:  20f5 d6e5 090a 2527 af84 ce0a 0000 0000  ......%'........
0x0020:  5e08 2000 d38a 0000 474e 5554 454c 4c41  ^.......GNUTELLA
0x0030:  2043 4f4e 4e45 4354 2f30 2e36 0d0a 582d  .CONNECT/0.6..X-
0x0040:  556c 7472 6170 6565 723a 2046 616c 7365  Ultrapeer:.False
0x0050:  0d0a 5573 6572 2d41 6765 6e74 3a20 4265  ..User-Agent:.Be
0x0060:  6172 5368 6172 6520 342e 302e 310d 0a4d  arShare.4.0.1..M
0x0070:  6163 6869 6e65 3a20 312c 382c 3531 302c  achine:.1,8,510,
0x0080:  312c 3136 3938 0d0a 506f 6e67 2d43 6163  1,1698..Pong-Cac
0x0090:  6869 6e67 3a20 302e 310d 0a58 2d51 7565  hing:.0.1..X-Que
0x00a0:  7279 2d52 6f75 7469 6e67 3a20 302e 310d  ry-Routing:.0.1.
0x00b0:  0a48 6f70 732d 466c 6f77 3a20 312e 300d  .Hops-Flow:.1.0.
0x00c0:  0a4c 6973 7465 6e2d 4950 3a20 3134 382e  .Listen-IP:.148.
0x00d0:  3633 2e32 3134 2e32 3339 3a36 3334 360d  63.214.239:6346.
0x00e0:  0a52 656d 6f74 652d 4950 3a20 5858 5858  .Remote-IP:.XXXX
0x00f0:  5858 582e 3938 2e31 3133 0d0a 4650 2d41  XXX.98.113..FP-A
0x0100:  7574 682d 4368 616c 6c65 6e67 653a 204d  uth-Challenge:.M
0x0110:  4a53 424d 3455 5853 355a 544c 524f 4d42  JSBM4UXS5ZTLROMB
0x0120:  4736 5035 3742 4e52 4b4a 514f 4846 550d  G6P57BNRKJQOHFU.
0x0130:  0a0d 0a                                   ...
```

Analysis of the data in this packet displays IP: 148.63.214.239, external to the network and attempting to connect to internal IP: 32.245.214.229. But wait! Isn't NAT being used on this network? Tcpdump was re-ran again with the –e option and received the same MAC addresses as before: 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33. This reveals that the 32.245.214.229 is not on the DMZ with the Snort IDS. It is in fact placed behind the 00:00:0c:04:b2:33 Cisco PIX firewall, indicating that the .229 IP is "statically translated".

Looking at the right hand side of this "dump" the ASCII translation of the hex data can be seen, showing "GNUTELLA CONNECT/0.6" an indication of the Peer – to – Peer network Gnutella. The 148.63.214.239 IP making this connection is using the commonly used program "Bearshare".

Going back to the Snort stat output, there is no evidence of a Gnutella connection, because if it was in the packet, Snort should have detected it.

## Analysis of Snort rules

Let's take a look at the p2p.rules file where the Gnutella rules are written:

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P Outbound GNUTella client request"; flow:to_server,established; content:"GNUTELLA CONNECT"; depth:40; classtype:policy-violation; sid:556; rev:5;)

This rule should have detected the Gnutella traffic, but failed to do so. The reasons being are:

1. The $HOME_NET variable was specified with the –h option, the data being analyzed is a "playback" written previously. Only the inbound connection is visible.

2. The rule modifier "flow:to_server,established" allows rules to only apply in a specified direction. This rule is written to observe traffic outbound from the $HOME_NET to $EXTERNAL_NET.

By removing –h from the command line options *and* removing the "flow" statement, the rule is triggered and displays the following output.

```
[**] P2P outbound GNUTella client request [**]
10/28-20:00:15.916507 148.63.214.239:2314 -> 32.245.214.229:9511
TCP TTL:113 TOS 0x0 ID:63919 IpLen:20 DgmLen:307 DF
****P*** Seq: 0xAF84CE0A Ack: 0x0 Win: 0x2000  TcpLen: 20
```

### Conclusion on incorrectly written Snort Rule

*It is suggested that a separate set of rules be developed for the playback of binary data. Binary data from Snort only records one-way traffic, unless the modifier "tag" is used within the rule. Options like flow cannot be used when reading binary data. Snort is not aware of the direction the traffic is flowing. Care must also be taken when specifying $HOME_NET and $EXTERNAL_NET variables as playback may not function properly with these set.*

## References

*1* -- http://www.tcpdump.org/tcpdump_man.html
*2* -- http://coffer.com/mac_find/
*3* -- http://www.cisco.com
*4* -- http://www.webopedia.com/TERM/D/DMZ.html
*5* -- http://linuxcentral.com/catalog/index.php3?prod_code=T000-272
*6* -- http://www.dynamicelements.com/Helpfulpages/subnet.htm
*7* -- http://www.**squid**-cache.org/
*8* -- http://www.whitehats.com/ids/
*9* --
http://www.iss.net/products_services/enterprise_protection/rsnetwork/sensor.php
*10* – http://www.snort.org
*11* -- http://lcamtuf.coredump.cx/p0f.shtml
*12* –
http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/mod_icn/fwsm/fwsm_2_2/fwsm_red/df.htm#wp1028903
*13* – http://www.ethereal.com