Intrusion Detection In-Depth

GCIA Practical Assignment

Version 3.1

Jason Tant

SANS – Washington D.C

May 5 – May 12, 2002

Table of Contents

1 The Current State of Intrusion Detection

"It Began With a Ping"

1.0 The Fallacies

"Why would anyone hack into our network? We do not have any data worth stealing and we have so few machines." An associate of mine recently asked me this question after a single machine isolated inside the DMZ of his firewall, was compromised. The answer was simple: The machine was hacked because it could be hacked. There was some security flaw, hole, or vulnerability in the system that allowed it to be subverted and used for someone else's purposes.

I remember falling into that same trap some time before with a Microsoft Windows network to which I had access. This particular network was small. It was merely ten to fifteen machines sharing a connection to the Internet. I fell into this trap, that is, until my eyes were opened through a study of intrusion detection and analysis. In this paper, I will demonstrate some of the common tools and techniques used today for intrusion analysis in the context of a real-world hack captured on this small network. Through this discussion, I shall illustrate the importance of thoroughness, diligence and knowing your network in the scope of an intrusion analysts' duties as well as driving home the point that complacency is our worst enemy.

1.1 WinDump: Harvesting the Data

At the time of the event in question, I was manually rotating my WinDump logs every morning. According to the Politecnico di Torino website which hosts the software, WinDump is: "the porting to the Windows platform of tcpdump, the most used network sniffer/analyzer for UNIX...[WinDump] is fully compatible with tcpdump and can be used to watch and diagnose network traffic according to various complex rules...WinDump is free and released under a BSD-style license."[1] Tcpdump version 3.7 actually introduced the capability to handle log rotation, but the current release of WinDump at the time of this writing is based upon tcpdump version 3.5.2, which does not. An example script and method to handle WinDump log rotation in a Windows environment can be found in Appendix A.

WinDump's place in this network was to capture and log all incoming and outgoing traffic in binary format for later review. This would create an audit trail for what had happened on our network during the previous day. In this environment, WinDump was run with the "-s 1500" and "-w <file>" options.

According to the WinDump manual, "-s 1500" is used to expand the default packet capture size in bytes, known as the snaplen, to 1500.[2] In her presentation on fragmentation theory, security analyst Judy Novak states that 1500 bytes is the

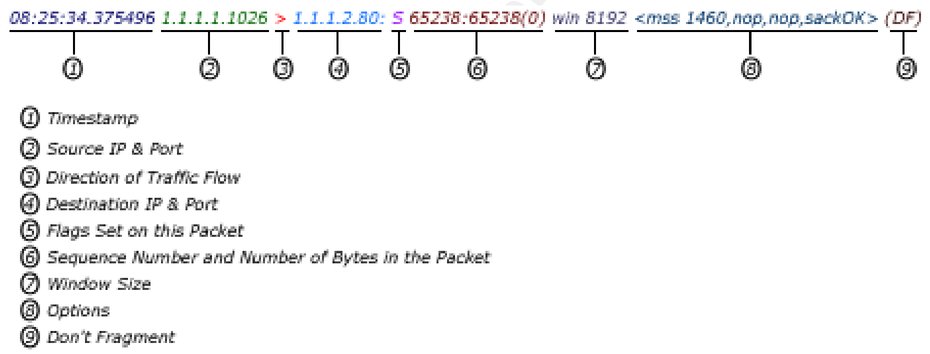
[1] Politecnico di Torino. "WinDump: tcpdump for Windows."
[2] Politecnico di Torino. "Windump Manual."

maximum size for an IP datagram[3]. Thus, this option ensured that full packets were captured as they traversed the network.

The WinDump manual also states that the "-w <file>" option will cause the application to "write the raw packets to *file* rather than parsing and printing them out. They can later be printed with the -r option."[4] This allowed us to do post processing of the data offline. As a word of caution, on this small, low-volume network, WinDump captured an average of three hundred megabytes of data per day. In a larger environment, this setup would most likely not be a scalable solution without addressing space and bandwidth utilization.

1.1.0 The Anatomy of WinDump Text Output

The options presented thus far allow WinDump to log packets in its binary format. Later in this paper, we will see WinDump used with filters to deliver specific subsets of packets in a text-based format. A breakdown of typical WinDump TCP text output (adapted from a description found in Intrusion Signatures and Analysis for tcpdump[5]) can be seen below in Figure 1.



```
08:25:34.375496 1.1.1.1.1026 > 1.1.1.2.80: S 65238:65238(0) win 8192 <mss 1460,nop,nop,sackOK> (DF)
```

①
②
③
④
⑤
⑥
⑦
⑧
⑨
① Timestamp
② Source IP & Port
③ Direction of Traffic Flow
④ Destination IP & Port
⑤ Flags Set on this Packet
⑥ Sequence Number and Number of Bytes in the Packet
⑦ Window Size
⑧ Options
⑨ Don't Fragment

Figure 1 - Typical WinDump TCP Text Output

1.2 Snort: Finding Needles in the Haystack

Once the WinDump log files had been rotated, it was time to begin analysis. For this task, I turned to Snort. The official Snort website describes Snort as: "a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks…Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture."[6]

[3] Novak, p.3-4
[4] Politecnico di Torino. "Windump Manual."
[5] Northcutt et al. p.2
[6] Roesch et al.

Snort is open source, free, and licensed under the GNU Public License. In addition to its real-time analysis features, Snort possesses the capability to read in packet logs written in the libpcap format (which is used by both WinDump and tcpdump). By using Snort in this fashion, we are able to carefully analyze the network traffic from the previous day in search of potential problems.

There are a host of different configurations available for using Snort which can be readily reviewed in the Snort User's Manual provided with the software at: http://www.snort.org/dl. Up to date Windows binary versions of the Snort software are available from: http://www.silicondefense.com/techsupport/downloads.htm. At the time of this event, I was using Snort with an up to date rule set (found at: http://www.snort.org/dl/signatures/), the default preprocessors, and a small library of local rules which I had built to monitor traffic from suspicious IP addresses. I ran Snort twice against each log file. Both configurations logged packets of interest to binary files. The first run output in full alert mode and the second in fast alert mode (after changing the name of the alert.ids and portscan.log files accordingly).

The purpose for running Snort offline in this fashion was to separate the capturing of the data from the processing of the data. Data processing tends to be the most CPU intensive part of intrusion detection. Thus offline processing of data allowed me to perform larger and more complex analysis in Snort without a concern for CPU utilization and the effects this would have on real-time analysis.

1.2.0 The Anatomy of a Snort Alert

Before proceeding further, it may be helpful to review the format in which Snort prints full alerts. A typical Snort full text alert[7] can be seen in Figure 2.

Alert Message Name
[**] [1:530:5] NETBIOS NT NULL session [**]

Alert Message Classification
[Classification: Attempted Information Leak] [Priority: 2]

Timestamp	Source IP Address : Port	Traffic Flow	Dest IP Address : Port
01/17-06:25:55.102886	1.1.1.1:1256	->	1.1.1.2:139

Protocol	Time To Live	Type of Service	IP ID	Header Length	Datagram Length	Frag Bits
TCP	TTL:128	TOS:0x0	ID:18220	IpLen:20	DgmLen:214	DF

TCP Flags	TCP Sequence ID	TCP Ack	TCP Window Size	TCP Header Length
AP	Seq: 0x2827F3BF	Ack: 0xBE4898EA	Win: 0x21D3	TcpLen: 20

References to this Alert
[Xref => http://www.securityfocus.com/bid/1163] [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0347] [Xref => http://www.whitehats.com/info/IDS204]

[7] Based upon Roesch, p.39.

Figure 2 - A Typical Snort Full Text Alert

1.3 Sorting the Needles

With the log files rotated and the alert files generated, it was time to review the Snort output. With all of the Snort rule sets active, it can generate quite a bit of traffic. Even on a small network you can quickly fall prey to information overload unless you have a managed, systematic approach to analysis. There are tools available to help aid in sorting through the information, such as SnortSnarf, which takes the text-based log files and converts them into linked HTML pages. Preferring a bit more control over my data though, I wrote a very simple script, which uses ActiveState's Perl for Windows and the Cygwin Unix toolset for Windows to parse out unique alert messages from the full alerts list and present a quick list in a text file. This script can be found in Appendix A.

At first glance, I thought it would be just another day. I would check all of the alert messages and they would turn out to be the standard network traffic that I would see on a daily basis. Glancing through the relatively short list, nothing really jumped out at me. That was my first mistake.

1.4 A Needle Among Needles

So I went about business as usual. Again using the Cygwin toolset, I created text files from the fast alert list containing each particular error message. Visually skimming through the gaggle of ICMP PING alert messages, I searched for IP addresses not belonging inside the network. Lo and behold, I found one. Looking a bit closer at the message I realized that this was something new. I flipped back to the summary file I had built. Did I miss something?

The answer was an unequivocal yes. A different type of ICMP PING had come across the network overnight. In my haste however, I had missed it. The alert message triggered by Snort was:

```
[**] ICMP PING Delphi-Piette Windows [**]
```

At this point my curiosity was peaked. Opening the full alerts file and searching on "Delphi" I found the following alert:

```
[**] [1:372:4] ICMP PING Delphi-Piette Windows [**]
[Classification: Misc activity] [Priority: 3]
01/17-18:06:47.847936 some.bad.guy.ip -> my.local.ip.net
ICMP TTL:46 TOS:0x0 ID:35287 IpLen:20 DgmLen:84
Type:8  Code:0  ID:512   Seq:51388  ECHO
[Xref => http://www.whitehats.com/info/IDS155]
```

Figure 3 - Snort ICMP Delphi-Piette Alert Message

Noticing the reference link at the end of the alert, I hopped over to the Whitehat arachNIDS database, which stated: "This event indicates that a ping request was sent to your network. Ping requests are usually used to determine whether a host is responsive, but can be misused to map your network. This particular ping was probably generated by software using Delphi code (written by F. Piette)."[8]

This summary of the alert was fairly standard in comparison with other ICMP PING summaries, with the exception of the last line about Delphi and F. Piette. Still, this was not enough to raise serious concerns in my mind. I accepted the possibility that it could be someone attempting to map the network and warranted further investigation.

Immediately following the summary section on the Whitehat web site was a section called "How Specific". In that section it was indicated that the rule associated with this particular event was very specific.[9] Checking the Snort rule set confirmed that the content section was indeed detailed, as shown in the figure below.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING Delphi-Piette Windows";
content:"|50696e67696e672066726f6d2044656c|"; itype:8; depth:32; reference:arachnids,155; sid:372;
classtype:misc-activity; rev:4;)
```

Figure 4 - Snort ICMP Delphi-Piette Rule

1.5 So You Found a Ping. What Now?

Using WinDump's "–X" option for printing both hex and ASCII translations of the packet, "-n" to not convert IP addresses to names, "-vv" for extra verbose output, and a special filter to only display icmp data from this particular IP, I decoded the packet again. The data in the packet revealed a surprising ASCII translation of the PING message which precisely correlated back to the content string in the Snort rule, as shown in the underlined portion below.

```
18:06:47.847936 some.bad.guy.ip > my.local.ip.net: icmp: echo request (ttl 46, id 35287, len 84)
                 0x0000  4500 0054 89d7 0000 2e01 8417 xxxx xxxx      E..T........xxxx
                 0x0010  xxxx xxxx 0800 5096 0200 c8bc 5069 6e67      xxxx..P.....Ping
                 0x0020  696e 6720 6672 6f6d 2044 656c 7068 6920      ing.from.Delphi.
                 0x0030  636f 6465 2077 7269 7474 656e 2062 7920      code.written.by.
                 0x0040  462e 2050 6965 7474 6520 2020 2020 2020      F..Piette.......
                 0x0050  2020 2020                                    ....
```

Figure 5 - WinDump ASCII Dump of ICMP Ping

Following this packet came an echo reply from the machine in question, which happened to be a local Windows 2000 server. Some quick web research on "Delphi" and "Piette" turned up the web site "Overbyte – Freeware by Frank Piette". Available from this site was a product called ICS, or Internet Connection Suite, which is described

[8] Whitehat.
[9] Whitehat.

as “various Internet components and applications…distributed as freeware with *full source code* for all Delphi and C++Builder versions.”[10] According to the site, one of the standard, pre-built components is “TPing” which provides “ICMP Ping support. You can Ping a host and get the resulting info.”

Having a background in software engineering, the wheels in my mind began to draw upon the possibility that someone could have built an application on top of this suite of tools for more malevolent purposes. I started to wonder if this person attempted more upon the network than just ping the fileserver. I searched the alerts files for the foreign IP address. There were no further alerts containing that source, but was there more than what Snort was telling me?

1.6 That Sinking Feeling

Using WinDump once again, I built a filter to show a text dump of all packets coming from or going to the external IP address that had pinged my network. I knew that there was something wrong when the text output, which should not have been more than one or two kilobytes, turned out to be eight megabytes of data. This was much more than a single ping.

1.6.0 Port 3389: The Second Volley

The text file showed that immediately following the initial ping (within 1 second of the response from the server) a TCP SYN packet was sent to port 3389, as the trespasser attempted to establish a connection. The server responded with a SYN-ACK, acknowledging that something was listening on that port and ready to accept connections. The hacker’s machine gracefully responded with a TCP ACK packet one second later, thus completing the TCP three-way handshake and practically eliminating the possibility that this IP address was being spoofed. Roughly one minute later, the local server responded with an RST packet, closing the connection. The transaction is shown in the figure below.

```
18:06:48.432833 some.bad.guy.ip.2699 > my.local.ip.net.3389: S [tcp sum ok]
1809622998:1809622998(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 35347, len 48)

18:06:48.433019 my.local.ip.net.3389 > some.bad.guy.ip.2699: S [tcp sum ok]
1920898890:1920898890(0) ack 1809622999 win 17520 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id
5035, len 48)

18:06:49.019275 some.bad.guy.ip.2699 > my.local.ip.net.3389: . [tcp sum ok] 1:1(0) ack 1 win 17520
(DF) (ttl 110, id 35421, len 40)

18:07:49.411456 my.local.ip.net.3389 > some.bad.guy.ip.2699: R [tcp sum ok]
1920898891:1920898891(0) win 0 (DF) (ttl 128, id 5332, len 40)
```

Figure 6 - TCP Connection to Port 3389

[10] Piette.

Checking my trusty Neohapsis ports list, I found that port 3389 belongs to Microsoft terminal services.[11] Just to verify this information, I checked the Microsoft support network, which confirmed that terminal services does indeed use that port.[12] Being familiar with this particular server, I was well aware that it was running terminal services.

1.6.1 Paging NetBIOS: Please Pick Up the White Courtesy Phone

According to the WinDump log file, two seconds after sending the ACK packet completing the three-way handshake on the terminal services port, the attacker attempted to connect to port 139. The Neohapsis ports list identifies this port as a host for a number of possible Trojans including Chode, God Message Worm, Msinit, Netlog, Network, Qaz, Sadmind, and SMBRelay.[13] Additionally, port 139 over TCP is well known as the host for the Microsoft Windows NetBIOS session service. Microsoft's support web site officially lists port 139 as handling the following services: directory replication, event viewer, file sharing, logon sequence, pass through validation, performance monitor, printing, registry editor, server manager, trusts, user manager, WinNT diagnostics, WinNT secure channel (Q150543)[14] and nbsession (Q204279).[15] The local server kindly responded with a SYN-ACK packet and again the attacker reset the connection. This transaction is shown in the figure below.

```
18:06:51.908697 some.bad.guy.ip.2700 > my.local.ip.net.139: S [tcp sum ok]
1809807568:1809807568(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 35796, len 48)

18:06:51.908873 my.local.ip.net.139 > some.bad.guy.ip.2700: S [tcp sum ok]
1921816835:1921816835(0) ack 1809807569 win 17520 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id
5042, len 48)

18:06:52.459715 some.bad.guy.ip.2700 > my.local.ip.net.139: R [tcp sum ok]
1809807569:1809807569(0) win 0 (ttl 110, id 35859, len 40)
```

Figure 7 - TCP Connection to Port 139

At the same time that this was happening, a TCP connection was attempted to port 135, with similar results. Port 135, according to the Neohapsis list, hosts DCE endpoint administration and NCS local location broker.[16] On the other hand, Microsoft's support site lists port 135 as handling: DHCP manager, DNS administration, and WINS manager on Windows NT as well as managing client/server communications, exchange administrator, and remote procedure calls on Microsoft Exchange Server (which this particular server happened to be running).[17] To recap, in six seconds the attacker had at a bare minimum learned that the host was alive, running Microsoft terminal services, NetBIOS and that it possibly acted as an exchange sever with no connections to these ports blocked into or out of the host network, but that was only the beginning.

[11] Neohapsis.
[12] Microsoft Product Support Services. (Q150543).
[13] Neohapsis.
[14] Microsoft Product Support Services. (Q150543).
[15] Microsoft Product Support Services. (Q204279).
[16] Neohapsis
[17] Microsoft Product Support Services. (Q150543).

1.7 Houston, We Have a Problem

Probably what were the most significant connections came next in the WinDump logs. Starting within a second of the connection attempts to ports 139 and 135 respectively came another connection attempt to port 139 and one to port 445. Upon establishment of the three-way handshake to port 139, a NetBIOS session was immediately requested (confirmed through RFC1002 that the proper flag setting for a NetBIOS session request is 0x81[18]). The response from the server contained flag 0x82, which again according to RFC1002 is the proper response for a "positive session response". This transaction is shown below:

```
18:06:52.597170 some.bad.guy.ip.2721 > my.local.ip.net.139: S [tcp sum ok]
1811733967:1811733967(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 35876, len 48)

18:06:52.597334 my.local.ip.net.139 > some.bad.guy.ip.2721: S [tcp sum ok]
1922094127:1922094127(0) ack 1811733968 win 17520 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id
5046, len 48)

18:06:53.105577 some.bad.guy.ip.2721 > my.local.ip.net.139: P [tcp sum ok] 1:73(72) ack 1 win 17520
>>> NBT Packet
NBT Session Request
Flags=0x81000044
Destination=*SMBSERVER      NameType=0x20 (Server)
Source=INTRA           NameType=0x00 (Workstation)
 (DF) (ttl 110, id 35932, len 112)

18:06:53.105793 my.local.ip.net.139 > some.bad.guy.ip.2721: P [tcp sum ok] 1:5(4) ack 73 win 17448
>>> NBT Packet
NBT Session Granted
Flags=0x82000000
 (DF) (ttl 128, id 5048, len 44)
```

Figure 8 - NetBIOS Session Establishment Packets

The hacker then reset the connection. On port 445 however, a connection was established and data began to flow back and forth. This was evident in the series of packets which followed with both the PSH and ACK flags set traveling in both directions as shown below (only a small sampling is provided to illustrate the point.):

```
18:06:52.606805 some.bad.guy.ip.2720 > my.local.ip.net.445: S [tcp sum ok]
1811672593:1811672593(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 35875, len 48)

18:06:52.606981 my.local.ip.net.445 > some.bad.guy.ip.2720: S [tcp sum ok]
1922139918:1922139918(0) ack 1811672594 win 17520 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id
5047, len 48)

18:06:53.116539 some.bad.guy.ip.2720 > my.local.ip.net.445: . [tcp sum ok] 1:1(0) ack 1 win 17520 (DF)
(ttl 110, id 35934, len 40)

18:06:53.140644 some.bad.guy.ip.2720 > my.local.ip.net.445: P [tcp sum ok] 1:138(137) ack 1 win 17520
(DF) (ttl 110, id 35938, len 177)
```

[18] Network Working Group, p.29.

```
18:06:53.141354 my.local.ip.net.445 > some.bad.guy.ip.2720: P [tcp sum ok] 1:179(178) ack 138 win
17383 (DF) (ttl 128, id 5050, len 218)
```

Figure 9 - Connection Established and Data Transfer on Port 445

Flipping back to the Neohapsis list, we find that port 445 handles Windows 2000 server message block (SMB) communication.[19] Microsoft refers to this port as running "direct hosted 'NetBIOS-less' SMB traffic" on Windows 2000. Microsoft claims that using SMB in this fashion allows "simplification in transport of SMB, removal of WINS and NetBIOS broadcast as a means of name resolution, and standardization of name resolution on DNS for file and printer sharing". [20]

The specifics of the data transfer moves us out of the incident analysis and detection scope of this paper and into incident handling. That said, I will summarize this connection and the string of subsequent connections to port 445 over the course of several hours by stating that the Windows domain user names, groups, machine names and shared files were enumerated and transferred to the attacking IP. A total of twelve megabytes of information was acquired from the network.

1.8 Lessons Learned – Detecting and Preventing Future Attempts

1.8.0 Firewall Policy Change Recommendations

The first and most obvious problem on this network was the lack of an adequate firewall policy. Todd Sabin, a member of the RAZOR team at BindView, gave a presentation on "Windows 2000, Null Sessions and MSPC" in which he stated that the default firewall rules should be to deny any access not specifically authorized. Sabin also recommended that the following ports, several of which were seen in this attack, should be blocked: 135/UDP/TCP, 137/UDP, 138/UDP, 139/TCP, 445/TCP and 593/TCP.[21] Additionally, it would have been helpful to block external ICMP pings and their associated responses at the firewall. Had these measures been in place, these particular vulnerabilities could not have been exploited in this fashion.

1.8.1 Network Intrusion Detection Updates

The traffic, as shown within this paper does not violate the rules of the RFCs, which pertain to NetBIOS and SMB. Thus it is my recommendation to create a network intrusion rule that detects the end packet of the TCP three-way handshake from an external network as well as any SMB data being sent into or out of the network. In Snort, the way to achieve this would be:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg: "LOCAL ALERT External
WIN2K SMB Access"; flags:A+;)
```

[19] Neohapsis.
[20] Microsoft Product Support Services. (Q315267).
[21] Sabin, p.21.

Figure 10 - Recommended Additional Snort SMB Traffic Rule

This rule has been tested against this particular attack and generated a significant number of positive alerts, which would have lent weight to the situation in a real-time environment.

Another technique, which I strongly recommend, is the use of IP watch lists. An IP watch list is a rule set up to fire an alert if a specific IP address or range of IPs are seen upon your network. A sample rule from this incident would be:

```
alert tcp some.bad.guy.ip any -> any any (msg: "WATCHLIST Previous Successful
Hacker";)
```

Figure 11 - Snort IP Watch List Example

This rule will ensure that even if our hacker is not doing anything particularly suspicious, Snort will inform us so as to be aware. Also note that this rule only encompasses TCP traffic. It can be duplicated to include UDP and ICMP packets as necessary. We certainly want to know if our hacker returns.

1.8.2 Disabling NetBIOS Over TCP/IP

Microsoft presents several different knowledgebase archives that explain how to disable NetBIOS over TCP/IP. Two of which are: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q204279 and http://support.microsoft.com/default.aspx?scid=kb;en-us;Q315267.

However, Microsoft does present a warning about performing this operation. Both of these articles state that “the Windows-based computer will be unable to communicate with earlier operating systems using SMB traffic.”[22]

1.8.3 Microsoft Baseline Security Analyzer

Microsoft also provides a free baseline security analyzer, which can aid in ensuring your machines are properly patched and secured. It is available via the web at: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/Tools/mbsahome.asp.

1.9 Conclusions

The size of your network and the sensitivity of your data are not the only factors that need to be taken into consideration when determining if a hacker might be interested in your network. To recap, this particular network had no sensitive data and ten to fifteen machines, sharing an Internet connection. Have no doubt that if you have machines connected to the Internet, someone is interested in them. You should consider them at

[22] Microsoft Product Support Services. (Q315267).

risk and a potential tool for someone else to use for purposes other than those which you have intended.

Setting up a network intrusion detection system and knowing the typical traffic you should be sending into and out of your network lends a great deal in understanding and determining when something is wrong or out of the ordinary. As illustrated in this paper however, we must ensure that we do not become complacent and fail in our duty to pursue even those small anomalies that seem to be benign. Complacency almost caused me to miss a successful hack of a network, all because the attacker only set off a single, seemingly benign Snort alert. The entire attack was found because it began with nothing more than a ping.

1.10 Works Cited

Microsoft Product Support Services. “Windows NT, Terminal Server, and Microsoft Exchange Services Use TCP/IP Ports (Q150543).” 8 August 2001. URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;Q150543 (4 June 2002).

---. “Direct Hosting of SMB Over TCP/IP (Q204279).” 22 December 2001. URL: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q204279 (4 June 2002).

---. “The Advantages of Direct Hosting of SMB over TCP/IP (Q315267).” 24 January 2002. URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;Q315267 (13 June 2002).

Neohapsis. “Ports List.” 12 June 2002. URL: http://www.neohapsis.com/neolabs/neo-ports/ (4 June 2002).

Network Working Group. “RFC 1002: PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT: DETAILED SPECIFICATIONS.” March, 1987. URL: http://www.ietf.org/rfc/rfc1002.txt?number=1002 (13 June 2002).

Novak, Judy. “IP Behavior II: Frag men ta **tion**.” SANS Network Intrustion In-Depth. 2000, 2001: 3-4.

Northcutt, Stephen, et al. Intrusion Signatures and Analysis. Indianapolis: New Riders Publishing, January 2001. 2.

Piette, Frank. “Products: ICS.” Overbyte: Freeware by Frank Piette. URL: http://overbyte.alexid.fr/frame_index.html?redirTo=/products/ics.html (4 June 2002).

Politecnico di Torino. “WinDump: tcpdump for Windows.” 28 March 2002. URL: http://windump.polito.it/ (3 June 2002).

----. “Windump Manual.” Windump: tcpDump for Windows. 14 March 2002. URL: http://windump.polito.it/docs/manual.htm (3 June 2002).

Roesch, Martin. “Intrusion Detection Snort Style.” SANS Network Intrusion In-Depth. 2000, 2001: 39.

Roesch, Martin, et al. “More Info About Snort.“ Snort - The Open Source Network Intrusion Detection System. URL: http://www.snort.org/about.html (4 June 2002).

Sabin, Todd. “Windows 2000, Null Sessions and MSPC.” February 2001. URL: http://razor.bindview.com/publish/presentations/files/nullsess.ppt (4 June 2002).

Whitehat. “IDS155/ICMP_PING DELPHI-PIETTE WINDOWS.” arachNIDS - The Intrusion Event Database. URL: http://www.whitehats.com/info/IDS155 (4 June 2002).

2 Network Detects

2.0 Detect #1 – DNS Spoof False Positive

2.0.0 Event Trace

The following is the Snort alert to be analyzed:

```
[**] [1:254:2] DNS SPOOF query response with ttl: 1 min. and no authority [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/31-14:19:33.347677 my.external.dns.server:53 -> 192.168.15.100:2689
UDP TTL:117 TOS:0x0 ID:32888 IpLen:20 DgmLen:76
Len: 56
```

Figure 12 - Snort DNS Spoof Alert

Below is the correlating WinDump output of the event to be analyzed:

```
14:19:33.347677 my.external.dns.server.53 > 192.168.15.100.2689:  [udp sum ok] 1 q: A?
www.google.com. 1/0/0 www.google.com. A 216.239.33.101 (48) (ttl 117, id 32888, len 76)

                0x0000  4500 004c 8078 0000 7511 b6ed xxxx xxxx     E..L.x..u...xxxx
                0x0010  xxxx xxxx 0035 0a81 0038 1cbf 0001 8180     xxxx.5...8......
                0x0020  0001 0001 0000 0000 0377 7777 0667 6f6f     .........www.goo
                0x0030  676c 6503 636f 6d00 0001 0001 c00c 0001     gle.com.........
                0x0040  0001 0000 003c 0004 d8ef 2165               .....<....!e
```

Figure 13 - DNS Spoof Packet

2.0.1 Source of Trace

The source of this event was a Windows NT and Windows 2000 network using a single class-C reserved IP addressing scheme locally with no firewall. This local network connected to remote office DNS servers via a frame relay. The remote network used multiple public class-C IP addresses. The overall traffic on the local network was consistently low.

2.0.2 Detect Was Generated By

The alert for this event was generated by Snort version 1.86 with the DNS rules set last updated May 8, 2002. WinDump logs provided correlating data and an audit trail of packet traces for this event. The particular rule triggered by this packet is listed in figure 14. This particular Snort rule was written by Johan Augustsson in response to malicious uses of Dug Song's dsniff application which he stated produced two oddities (http://archives.neohapsis.com/archives/snort/2001-01/0034.html): "1. [Dsniff] sends at least two identical responses at a rapid pace. 2. The answers 'Time to live' (not the IP - TTL) is set to 1 minute - very short time."

```
alert udp $EXTERNAL_NET 53 -> $HOME_NET any (msg:"DNS SPOOF query response with ttl\: 1 min.
and no authority"; content:"|81 80 00 01 00 01 00 00 00 00|"; content:"|c0 0c 00 01 00 01 00 00 00 3c 00
04|"; classtype:bad-unknown; sid:254; rev:2;)
```

Figure 14 - Snort DNS Spoof Rule

As stated in the "Source of Trace" section above, the DNS server in question was actually part of the corporate WAN and not a part of the local LAN. Even though it was considered to be a trusted system, it was external to the local network and thus was not set in Snort's $HOME_NET variable, which lists internal IP addresses and ranges. This was done intentionally to ensure that in the event the frame relay or the remote DNS server were compromised, the traffic would still be monitored by Snort and the on-site analyst alerted to any suspicious activity.

The Snort alert can be broken down as follows:

Alert Message Name [**] [1:254:2] DNS SPOOF query response with ttl: 1 min. and no authority [**]					
Alert Message Classification [Classification: Potentially Bad Traffic] [Priority: 2]					
Timestamp 05/31-14:19:33.347677	Source IP Address : Port my.external.dns.server:53	Traffic Flow ->	Dest IP Address : Port 192.168.15.100:2689		
Protocol UDP	Time To Live TTL:117	Type of Service TOS:0x0	IP ID ID:32888	Header Length IpLen:20	Datagram Length DgmLen:76
UDP Header Length Len: 56					

Figure 15 - Snort DNS Spoof Alert Format

2.0.3 Probability the Source Address was Spoofed

Low. Given that the external server is a known system on the WAN and DNS traffic is expected from that machine, it is unlikely that the source address was spoofed.

2.0.4 Description of Attack

Based upon the Snort alert from Figure 15, we learn that the packet in question was an inbound UDP packet from a machine that was not part of the internal network. The offending packet originated from port 53, which is used for DNS communications and was bound for ephemeral port 2689 on the destination machine. It is difficult to garner further information about the attack without turning to the WinDump logs for correlation.

Analyzing the full packet decode of the DNS response, we see that there are two separate content sections which were matched by the Snort rule (as underlined in the figure below):

```
14:19:33.347677 my.external.dns.server.53 > 192.168.15.100.2689:  [udp sum ok] 1 q: A?
```

```
www.google.com. 1/0/0 www.google.com. A 216.239.33.101 (48) (ttl 117, id 32888, len 76)
                 0x0000  4500 004c 8078 0000 7511 b6ed xxxx xxxx      E..L.x..u...xxxx
                 0x0010  xxxx xxxx 0035 0a81 0038 1cbf 0001 8180      xxxx.5...8......
                 0x0020  0001 0001 0000 0000 0377 7777 0667 6f6f      .........www.goo
                 0x0030  676c 6503 636f 6d00 0001 0001 c00c 0001      gle.com.........
                 0x0040  0001 0000 003c 0004 d8ef 2165                .....<....!e
```

Figure 16 - DNS Spoof Content of Interest

Reviewing pages 26-28 of RFC 1035, “Domain Names – Implementation and Specification” (http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1035.html), the first content section decodes as shown in Figure 17. Because DNS response flags are set at the bit level, it is necessary to further break down some portions of this content from hexadecimal notation into binary notation.

DNS Response Flags	0x8180 = 1000 | 0001 | 1000 | 0000
Query Response	1 = Response Packet
Type of Query	0000 = Standard Query
Authoritative Answer	0 = Non-authoritative answer
Truncation	0 = No truncation
Recursion Desired	1 = Recursion is desired
Recursion Available	1 = Recursion is available
Reserved for Future Use	000
Response Code	0000 = No error condition
Number of Entries in the Question Section	0001 = 1 entry
Number of Resource Records in the Answer Section	0001 = 1 entry
Number of Resource Records in the Authority Records Section	0000 = No entries
Number of Resource Records in the Additional Records Section	0000 = No entries

Figure 17 - DNS Spoof Packet Decode for Content Section #1

From this data, we learn that this packet claims to be the response to a query initiated from the local machine. Since we have an audit trail with WinDump, we can build an appropriate filter to verify this fact.

```
14:19:33.307343 192.168.15.100.2689 > my.external.dns.server.53:  [udp sum ok] 1+ A?
www.google.com. [|domain] (ttl 128, id 40288, len 60)
                 0x0000  4500 003c 9d60 0000 8011 8f15 xxxx xxxx      E..<.`......xxxx
```

```
0x0010  xxxx xxxx 0a81 0035 0028 5804 0001 0100      xxxx...5.(X.....
0x0020  0001 0000 0000 0000 0377 7777 0667 6f6f      .........www.goo
0x0030  676c 6503 636f 6d00 0001 0001                gle.com.....
```

Figure 18 - DNS Query Packet Decode

As shown above, the connection was indeed initiated from the local machine. This appears to be a valid DNS query and response, which implies that the triggered Snort alert may be a false positive.

Continuing with the analysis, the second content section of interest is the response record portion of the offending packet. It decodes as follows:

Name:	c00c
Type:	0001 = DNS Query Type A
Class:	0001 = IN
Time to Live (in seconds):	0000 003c = 1 minute
RDLength (length in octets):	0004 = 32 bits
Rdata :	d8ef 2165 = (IP Address 216.239.33.101)

Figure 19 - DNS Spoof Packet Decode for Content Section #2

A quick check of the Arin Whois network on "216.239.33.101" returns that the IP address belongs to Google.com. Using our audit trail and checking the original DNS query initiated, we can confirm that Google.com was the requested lookup. Therefore, it stands to reason that this traffic was not malicious and we are dealing with a false positive stemming from a server that had a non-authoritative answer to the DNS request and a very low time to live for the DNS response.

2.0.5 Attack Mechanism

According to a newsgroup post made by Johannes Erdfelt, a network security professional for Mindspring Enterprises, DNS spoofing is simply "tricking the DNS system into believing [a] domain name is something other than it really is." His article, entitled "Everything you ever wanted to know about DNS Spoofing" can be found at: http://www.the-project.org/admins/0797/msg00070.html.

DNS spoofing is accomplished by causing a machine that has initiated a DNS lookup to accept falsified information. This information will then be used by the requestor to direct the client to the site of the malicious users' choice. In his paper "The Achilles Heel of DNS," (http://rr.sans.org/DNS/achilles.php) Christopher Irving writes at length about DNS security, stating:

> "…the DNS protocol has virtually no authentication method built into it. There is nothing in the protocol that provides a means to ensure that the requesting client is who it says it is or that the replying name server is a

> real name server. The message headers of a DNS query and response do contain a 16-bit identification field but it is mostly used for matching queries with responses. If an attacker can successfully predict future values of the identification number, he could fool a DNS client into accepting a false reply as the real one"

There are several purposes for which malicious users would want to spoof a DNS query. By setting up mock sites that appear to be the same as the actual sites and then redirecting traffic to the fake site, malicious hackers can trick unsuspecting users into providing passwords, credit card information, or other sensitive information. This information can be harvested and then used against the site or the user at a later time. DNS spoofing attacks could also be used for political means by redirecting users away from legitimate political sites to sites brandishing anything from negative content and commentary about individual candidates, parties, or platforms to pornographic and other sundry material.

2.0.6 Correlations

According to the Dshield.org list of top 10 ports attacked at the time of this writing (http://www.dshield.org/topports.html), port 53 is the sixth most attacked port. This shows the importance and criticality of investigating suspicious DNS traffic.

Similar traffic which generated the same Snort alert:
RPG. "[Snort-users] help with "DNS SPOOF" incidents." Snort-users Mailing List. 30 May, 2001. URL: http://www.geocrawler.com/mail/thread.php3?subject=%5BSnort-users%5D+help+with+%22DNS+SPOOF%22+incidents&list=4890

Information about DNS spoofing and the particular rule fired by Snort:
Augustsson, Johan. "Identifying dnsspoff." 5 January 2001. URL: http://archives.neohapsis.com/archives/snort/2001-01/0060.html (17 June 2002).

More information about DNS spoofing:
Patrick, Michael. "Has Your Domain Been Hijacked Lately?" 15 February 2001. URL: http://rr.sans.org/DNS/hijacked.php (18 June 2002).

Information on the security problems with DNS:
Irving, Christopher. "The Achilles Heel of DNS." 2 August 2001. URL: http://rr.sans.org/DNS/achilles.php (17 June 2002).

Information on the DNS standard:
Mockapetris, P. "RFC 1035: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION." November 1987. URL: http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1035.html (17 June 2002).

Mitre Common Vulnerabilities and Exposures listings correlating to DNS spoofing:
CVE-1999-0024 – "DNS cache poisoning via BIND, by predictable query IDs."

CVE-2000-0335 – “The resolver in glibc 2.1.3 uses predictable IDs, which allows a local attacker to spoof DNS query results.”
CVE-2000-0517 – “Netscape 4.73 and earlier does not properly warn users about a potentially invalid certificate if the user has previously accepted the certificate for a different web site, which could allow remote attackers to spoof a legitimate web site by compromising that site's DNS information.”

2.0.7 Evidence of Active Targeting

Given that the request was initiated by a local machine to the remote DNS server, this alert was legitimate, active targeting of the local machine.

2.0.8 Severity

Criticality: 1 – The local machine is an end-user workstation and is not providing any DNS services to the network.

Lethality: 1 – Had this been an actual, successful attack against the end-user workstation, ramifications and damage would have been minimal. Given that the user was searching for Google.com, it is unlikely that if the site were hijacked the user could have been duped into submitting sensitive information. The larger concern at that point would be the security of the remote DNS server and the frame relay between the sites.

System Countermeasures: 4 – This end-user workstation was a Windows NT operating system which was patched regularly, running only necessary services, and up-to-date anti-virus software. A maximum security rating of 5 for the system could be achieved by implementing a host-based firewall.

Network Countermeasures: 1 – There was no firewall in place. The router allowed most traffic through, as it assumed a trusted relationship over the frame relay to the remote site. There was a sniffer and a network intrusion detection system running on the local network, but these were not active countermeasures.

Severity = (1+1) – (4+1) = **-3**

2.0.9 Defensive Recommendations

With the exception of a host-based firewall, this particular system is well maintained. The network however, is severely lacking in security. Preferably a firewall should be implemented. If a firewall is not going to be installed, then the router should be reconfigured to narrow the inbound and outbound traffic to only that which is necessary.

2.0.10 Multiple Choice Test Question

```
14:19:33.347677 my.external.dns.server.53 > 192.168.15.100.2689:  [udp sum ok] 1 q: A?
www.google.com. 1/0/0 www.google.com. A 216.239.33.101 (48) (ttl 117, id 32888, len 76)
```

```
                  0x0000  4500 004c 8078 0000 7511 b6ed xxxx xxxx      E..L.x..u...xxxx
                  0x0010  xxxx xxxx 0035 0a81 0038 1cbf 0001 8180      xxxx.5...8......
                  0x0020  0001 0001 0000 0000 0377 7777 0667 6f6f      .........www.goo
                  0x0030  676c 6503 636f 6d00 0001 0001 c00c 0001      gle.com.........
                  0x0040  0001 0000 003c 0004 d8ef 2165                .....<....!e
```

In the above DNS packet, the 0x81 in the 30th byte offset represents which of the following DNS flag sets?

a. Inverse Query / Response Packet
b. Standard Query / Response Packet
c. Inverse Query / Query Packet
d. Standard Query / Query Packet

The correct answer is b. Standard Query / Response Packet.

2.1 Detect #2 – ICMP Source Quench

2.1.0 Event Trace

The following are the Snort alerts to be analyzed:

```
[**] [1:477:1] ICMP Source Quench [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/17-07:53:47.313120 192.168.20.10 -> 192.168.50.88
ICMP TTL:254 TOS:0x0 ID:58692 IpLen:20 DgmLen:56
Type:4  Code:0  SOURCE QUENCH

[**] [1:477:1] ICMP Source Quench [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/17-07:53:49.393151 192.168.20.10 -> 192.168.50.88
ICMP TTL:254 TOS:0x0 ID:59061 IpLen:20 DgmLen:56
Type:4  Code:0  SOURCE QUENCH

[**] [1:477:1] ICMP Source Quench [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/17-07:53:52.194037 192.168.20.10 -> 192.168.50.88
ICMP TTL:254 TOS:0x0 ID:59564 IpLen:20 DgmLen:56
Type:4  Code:0  SOURCE QUENCH
```

Figure 20 - Snort ICMP Source Quench Alerts

Below is the correlating WinDump output of the event to be analyzed:

```
07:53:47.313120 192.168.20.10 > 192.168.50.88: icmp: source quench for 192.168.50.88.5518 >
192.168.40.8.691: [|tcp] (DF) (ttl 126, id 30802, len 64) (ttl 254, id 58692, len 56)

                  0x0000  4500 0038 e544 0000 fe01 5692 xxxx xxxx      E..8.D..xxxx
                  0x0010  xxxx xxxx 0400 309a 0000 0000 4500 0040      xxxx..0.....E..@
                  0x0020  7852 4000 7e06 721a xxxx xxxx xxxx xxxx      xR@.~.r.xxxxxxxx
                  0x0030  158e 02b3 6313 5011                          ....c.P.
```

```
07:53:49.393151 192.168.20.10 > 192.168.50.88: icmp: source quench for 192.168.50.88.5518 >
192.168.40.8.691: [|tcp] (DF) (ttl 126, id 30816, len 64) (ttl 254, id 59061, len 56)
                 0x0000  4500 0038 e6b5 0000 fe01 5521 xxxx xxxx        E..8......U!xxxx
                 0x0010  xxxx xxxx 0400 309a 0000 0000 4500 0040        xxxx..0.....E..@
                 0x0020  7860 4000 7e06 720c xxxx xxxx0c6d 3205         x`@.~.r.xxxxxxxx
                 0x0030  158e 02b3 6313 5011                            ....c.P.
07:53:52.194037 192.168.20.10 > 192.168.50.88: icmp: source quench for 192.168.50.88.5518 >
192.168.40.8.691: [|tcp] (DF) (ttl 126, id 30829, len 64) (ttl 254, id 59564, len 56)
                 0x0000  4500 0038 e8ac 0000 fe01 532a xxxx xxxx        E..8......S*xxxx
                 0x0010  xxxx xxxx 0400 309a 0000 0000 4500 0040        xxxx..0.....E..@
                 0x0020  786d 4000 7e06 71ff xxxx xxxx xxxx xxxx        xm@.~.q.xxxxxxxx
                 0x0030  158e 02b3 6313 5011                            ....c.P.
```

Figure 21 - ICMP Source Quench Packets

2.1.1 Source of Trace

The source of this event was a Windows and Linux hybrid network using a single class-C reserved IP addressing scheme that connected to a mail server on the WAN via frame relay. The remote network used multiple public class-C IP addresses. 192.168.20.10 was a border router on the far end of the WAN. 192.168.50.88 was a Windows 2000 machine on the LAN. 192.168.40.8 was the remote mail server.

2.1.2 Detect Was Generated By

The alert for this event was generated by Snort version 1.86 with the ICMP rules set last updated October 30, 2001. WinDump logs provided correlating data and an audit trail of packet traces for this event. The particular rule triggered by this packet is listed in figure 22.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Source Quench"; itype: 4; icode: 0;
classtype:bad-unknown; sid:477; rev:1;)
```

Figure 22 - Snort ICMP Source Quench Rule

As stated in the "Source of Trace" section above, the mail server was part of the corporate WAN and not a part of the local LAN. Even though it was considered to be a trusted system, it was external to the local network and thus was not set in Snort's $HOME_NET variable, which lists internal IP addresses and ranges. This was done intentionally to ensure that in the event either the frame relay or the mail server were compromised, the traffic would still be monitored by Snort and the on-site analyst alerted to any suspicious activity.

The Snort alert can be broken down as follows:

```
Alert Message Name
[**] [1:477:1] ICMP Source Quench [**]
```

Alert Message Classification					
[Classification: Potentially Bad Traffic] [Priority: 2]					
Timestamp	Source IP Address	Traffic Flow	Dest IP Address		
05/17-07:53:47.313120	192.168.20.10	->	192.168.50.88		
Protocol	Time To Live	Type of Service	IP ID	Header Length	Datagram Length
ICMP	TTL:254	TOS:0x0	ID:58692	IpLen:20	DgmLen:56
ICMP Type & Code					
Type:4 Code:0 SOURCE QUENCH					

Figure 23 - Snort ICMP Source Quench Alert Format

2.1.3 Probability the Source Address was Spoofed

None. As shown in the correlating audit trail below, the TCP three-way handshake had already been completed and data was passing between the two before the source quench message was received.

```
07:38:47.204994 192.168.50.88.5518 > 192.168.40.8.691: P [tcp sum ok] 456:480(24) ack 1 win 17520
(DF) (ttl 128, id 28181, len 64)

07:38:47.411999 192.168.40.8.691 > 192.168.50.88.5518: . [tcp sum ok] 1:1(0) ack 480 win 17040 (DF)
(ttl 117, id 25520, len 40)

07:43:47.228209 192.168.50.88.5518 > 192.168.40.8.691: P [tcp sum ok] 480:504(24) ack 1 win 17520
(DF) (ttl 128, id 29109, len 64)

07:43:47.518257 192.168.40.8.691 > 192.168.50.88.5518: . [tcp sum ok] 1:1(0) ack 504 win 17016 (DF)
(ttl 117, id 31246, len 40)

07:48:47.251443 192.168.50.88.5518 > 192.168.40.8.691: P [tcp sum ok] 504:528(24) ack 1 win 17520
(DF) (ttl 128, id 30003, len 64)

07:48:47.491793 192.168.40.8.691 > 192.168.50.88.5518: . [tcp sum ok] 1:1(0) ack 528 win 16992 (DF)
(ttl 117, id 36168, len 40)
```

Figure 24 - Data Transfer Packet Trace

2.1.4 Description of Attack

Based upon the Snort alert in Figure 23, we are told that we are dealing with ICMP traffic coming from a border router on the WAN, bound for a machine inside the LAN. The border router is attempting to inform the LAN machine that it needs to slow down in the sending of traffic to some machine on the WAN.

Double-checking this information in the WinDump packets logged, we find that all three packets contain a 0x01 in the 9th byte offset field. This confirms that the protocol was set for ICMP. Starting in the 20th byte offset, we find 0x0400. These two bytes refer to the ICMP type (0x04) and the ICMP code (0x00). Together they lead us to believe that we are looking at an ICMP source quench message.

RFC 792, “INTERNET CONTROL MESSAGE PROTOCOL” (http://www.faqs.org/rfcs/rfc792.html pgs 10-11) tells us that the internet header and 64 bits of the original data causing this packet should be included as part of the source quench message. Therefore, we pull from the correlating WinDump logs this data contained within all three packets. It appears as shown in the figure below:

Packet #1 (07:53:47.313120)

```
4500 0040 7852 4000 7e06 721a xxxx xxxx
xxxx xxxx 158e 02b3 6313 5011
```

Packet #2 (07:53:49.393151)

```
4500 0040 7860 4000 7e06 720c xxxx xxxx
xxxx xxxx 158e 02b3 6313 5011
```

Packet #3 (07:53:52.194037)

```
4500 0040 786d 4000 7e06 71ff xxxx xxxx
xxxx xxxx 158e 02b3 6313 5011
```

Figure 25 – ICMP Source Quench Additional Data

Decoding the information available, we first notice that in the 8th byte offset we find 0x7e, which gives us a high time to live value of 126. According to Project Honeynet’s list of operating system fingerprints (http://project.honeynet.org/papers/finger/traces.txt), the default time to live value for a Microsoft Windows 2000 machine is 128. We can thus confirm that the traffic is being spoofed, these packets only traveled the expected two hops through the LAN router and out to the WAN router. In the 9th byte offset, we see 0x06. This tells us that the protocol of the packets the LAN machine was sending was TCP.

Jumping down to the TCP portion of these packets, we shall skip the source and destination ports momentarily and look at the 24th through 27th byte offsets. In all three packets we find 0x6313 and 0x5011. These octets represent the TCP sequence number of the packet. Given that all three are the same, combined with the spacing over time, we can infer that this is most likely a series of retries.

Moving back to the 20th through 23rd byte offsets, we find 0x158e and 0x02b3 in all three packets. These four octets tell us the source and destination ports are 5518 and 691, respectively. 5518 is an ephemeral port, and according to the Neohapsis ports list (http://www.neohapsis.com/neolabs/neo-ports/neo-ports.csv), no standard applications or Trojans utilize it. The destination port of 691 for both TCP and UDP however, is listed as handling MS Exchange routing (msexch-routing). Thus, it would appear the alerts were triggered on expected TCP traffic through the WAN border router, appropriately directed for the external Microsoft Exchange mail server.

The more important question to ask at this point should be what caused the WAN border router to become overloaded? Unfortunately, correlating data from a sensor monitoring the traffic traveling to and from the WAN border router is not available. While it is entirely possible that the WAN border router just became overloaded with the amount of normal traffic it was receiving, the network in question typically generated very little traffic. Thus, I am more inclined to believe that it was subjected to a denial of service attack.

2.1.5 Attack Mechanism

The intended purpose of ICMP source quench messages, as described in RFC 792, is to be used as a method by which routers can inform hosts of a need to slow down the speed with which they are sending data. This can be triggered by a full or nearly full buffer capacity on the router. It continues on to state that routers may send one source quench packet for each packet it receives. The sending host should then reduce its sending rate until it no longer receives these messages.

If used maliciously, ICMP source quench messages with spoofed IP addresses are a simple means of a denial of service attack. In the case of the network in this example, if someone were able to tap into the frame relay, they could spoof the IP address of the border router on the WAN and send ICMP source quench messages to the border router on the LAN. This would cause the LAN border router to slow any communications going out over the WAN, which in this case included mail, DNS, and file exchanges.

2.1.6 Correlations

Information on Handling Network Congestion in TCP/IP:
Nagle, John. "RFC 896: Congestion Control in IP/TCP Internetworks" January 1984. URL: http://www.faqs.org/rfcs/rfc896.html (23 June 2002).

Information on ICMP and Source Quench Standards:
Network Sorcery, Inc. "ICMP type 4, Source quench message." URL: http://www.networksorcery.com/enp/protocol/icmp/msg4.htm (22 June 2002).

Postel, J. "RFC 792: INTERNET CONTROL MESSAGE PROTOCOL" September 1981. URL: http://www.faqs.org/rfcs/rfc792.html (23 June 2002).

Wollman, Garrett. "ICMP source quench - deprecated?" net@FreeBSD.org Mailing List July 2001. URL: http://docs.freebsd.org/cgi/getmsg.cgi?fetch=90138+0+archive/2001/freebsd-net/20010722.freebsd-net (23 June 2002).

Other Examples of Source Quench Alerts:
Cyberarmy Project, The. "SnortSnarf signature page – ICMP Source Quench" June 2002. URL: http://www.cyberarmy.co.kr/snort/sig/sig11.html (22 June 2002).

Vianna, Vinicius. "ICMP Source Quench - Can it be some flood attack?" Neohapsis Incidents Mailing List. September 2000. URL: http://archives.neohapsis.com/archives/incidents/2000-09/0017.html (22 June 2002).

Source Quench as a Denial of Service Attack:
Oquendo, J. "Theories on new DoS Attacks v.1." URL: http://www.antioffline.com/TID/ (22 June 2002).

Source Quench Evidence of a DOS Attack:
Bleek, Thomas. "[FW1] intrusion/scan with icmp-source-quench?" FW-1 Mailing List. July 1999. URL: http://www.shmoo.com/mail/fw1/jul99/msg00408.html (23 June 2002).

There are currently no CVE entries involving generalized ICMP source quench attacks.

2.1.7 Evidence of Active Targeting

The traffic came from the WAN border router and was destined for a machine on the local LAN attempting to communicate with a mail server on the WAN. The packets were indeed actively targeted for the LAN machine, but they are legitimate. The larger question is whether or not the WAN border router was actively subjected to a denial of service attack, which cannot be determined with the available evidence.

2.1.8 Severity

There are two different scenarios here for which we can calculate the severity. The first is the severity of a source quench attack against the LAN machine and the second is a denial of service attack against the WAN border router.

2.2.9.1 Severity of a Source Quench Attack Against the LAN Machine

Criticality: 1 – The LAN machine is an end-user workstation.

Lethality: 2 – Assuming that the LAN machine noted and respected the ICMP source quench message, that individual workstation could have had its ability to gain email and web access significantly impacted.

System Countermeasures: 4 – This end-user workstation was a Windows 2000 operating system which was patched regularly, running only necessary services, and up-to-date anti-virus software. A maximum security rating of 5 for the system could be achieved by implementing a host-based firewall.

Network Countermeasures: 1 – There was no firewall in place. The router allowed most traffic through, as it assumed a trusted relationship over the frame relay to the remote site. There was a sniffer and a network intrusion detection system running on the local network, but these were not active countermeasures.

Severity of Source Quench Attack Against the LAN Machine = (1+2) – (4+1) = **-2**

2.2.9.2 Severity of a Source Quench Attack Against the WAN Border Router

Criticality: 5 – The WAN border router provides access to the company's DNS, secondary DNS, mail server, intranet web servers and corporate file sharing for LAN users.

Lethality: 4 – 90% of the work performed on the LAN required access to remote customers web sites, intranets, FTP servers, terminal services, database servers and mail servers. All of this data funneled through a single router. From a LAN perspective this is more likely a lethality of 5, but from an overall corporate view the lethality of DoSing the router would only affect one branch of users and thus be only a 4.

System Countermeasures: 1 – The router itself was not overly secure. It was not using secure or encrypted password schemes and passed this information in clear text down along the frame relay. The router was allowing most traffic to pass through. The operating system was not up to date and logging was either not enabled or not checked.

Network Countermeasures: 1 – There was no firewall in place. The router allowed most traffic through, as it assumed a trusted relationship over the frame relay to the remote site. There was a sniffer and a network intrusion detection system running on the local network, but these were not active countermeasures.

Severity of Source Quench Attack Against the LAN Machine = (5+4) – (1+1) = **7**

2.1.9 Defensive Recommendations

Because of the highly severe nature of a successful attack against the WAN border router or the LAN border router, it is recommended that:

a. Update the router's operating system
b. Enable secure passwords
c. Enable logging
d. Allocate personnel to monitor logs
e. Establish firewalls inside the network and significantly restrict access

If possible, it would also seem prudent to establish IPSec or VPN communications between the two routers. This way, if the frame relay was to become compromised it would be possible to have another layer of security protecting data.

2.1.10 Multiple Choice Test Question

When an ICMP Source Quench packet is sent, the internet header and what number bits are provided from the original packet is:

a. 16
b. 32
c. 64
d. 128

The correct answer is c. 64 bits.

2.2 Detect #3 – Network Misconfiguration

2.2.0 Event Trace

This trace is actually a series of events tracked on the network. The detects shown below are one set of traces in the series that mirrored the behavior of these packets.

The following is the Snort alert to be analyzed:

```
05/15-04:24:29.046034  [**] [1:473:1] ICMP redirect net [**] [Classification: Potentially Bad Traffic]
[Priority: 2] {ICMP} 172.19.1.1 -> 192.168.79.23

05/15-04:24:29.050017  [**] [1:473:1] ICMP redirect net [**] [Classification: Potentially Bad Traffic]
[Priority: 2] {ICMP} 172.19.1.1 -> 192.168.79.23

05/15-04:24:29.057979  [**] [1:473:1] ICMP redirect net [**] [Classification: Potentially Bad Traffic]
[Priority: 2] {ICMP} 172.19.1.1 -> 192.168.79.23

… (duplicate alerts advancing in milliseconds removed for brevity)

05/15-04:24:29.337950  [**] [1:473:1] ICMP redirect net [**] [Classification: Potentially Bad Traffic]
[Priority: 2] {ICMP} 172.19.1.1 -> 192.168.79.23

05/15-04:24:29.345923  [**] [1:449:4] ICMP Time-To-Live Exceeded in Transit [**] [Classification: Misc
activity] [Priority: 3] {ICMP} 172.19.1.1 -> 192.168.79.23
```

Figure 26 - Snort ICMP Redirect and TTL Exceeded Alerts

Below is the correlating WinDump output of the event to be analyzed:

```
04:24:29.046034 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 28, id 59357, len
44) (ttl 254, id 3331, len 56)

04:24:29.050017 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 26, id 59357, len
44) (ttl 254, id 3332, len 56)

04:24:29.057979 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 24, id 59357, len
44) (ttl 254, id 3334, len 56)

04:24:29.065954 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 22, id 59357, len
44) (ttl 254, id 3336, len 56)

04:24:29.073916 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 20, id 59357, len
44) (ttl 254, id 3338, len 56)

04:24:29.081884 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 18, id 59357, len
44) (ttl 254, id 3340, len 56)
```

```
04:24:29.186750 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 16, id 59357, len
44) (ttl 254, id 3342, len 56)

04:24:29.194718 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 14, id 59357, len
44) (ttl 254, id 3344, len 56)

04:24:29.202679 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 12, id 59357, len
44) (ttl 254, id 3346, len 56)

04:24:29.210640 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 10, id 59357, len
44) (ttl 254, id 3348, len 56)

04:24:29.218601 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 8, id 59357, len
44) (ttl 254, id 3350, len 56)

04:24:29.226569 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 6, id 59357, len
44) (ttl 254, id 3352, len 56)

04:24:29.234553 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 4, id 59357, len
44) (ttl 254, id 3354, len 56)

04:24:29.337950 172.19.1.1 > 192.168.79.23: icmp: redirect ext.norton.server.net to net
ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293:  udp 16 (ttl 2, id 59357, len
44) (ttl 254, id 3356, len 56)

04:24:29.345923 172.19.1.1 > 192.168.79.23: icmp: time exceeded in-transit for 192.168.79.23.1313 >
ext.norton.server.net.38293:  udp 16 [ttl 1] (id 59357, len 44) [tos 0xc0]  (ttl 254, id 3358, len 56)
```

Figure 27 – WinDump ICMP Redirect and TTL Exceeded Packets

2.2.1 Source of Trace

The source of this event was a Windows hybrid network using a single class-C reserved IP addressing scheme that connected to a WAN via frame relay. The remote network used multiple public class-C IP addresses. 172.19.1.1 was a border router on the far end of the WAN. 192.168.79.23 was a machine on the LAN. Ext.norton.server.net was a Norton Antivirus server on the WAN.

2.2.2 Detect Was Generated By

The alert for these events were generated by Snort version 1.86 with the ICMP rules set last updated October 30, 2001 and the ICMP-INFO rules set last updated October 30, 2001. WinDump logs provided correlating data and an audit trail of packet traces for this event. The particular rules triggered by these packets are listed in figure 28.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP redirect net";itype:5;icode:0;
reference:arachnids,199; reference:cve,CVE-1999-0265; classtype:bad-unknown; sid:473; rev:1;)

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Time-To-Live Exceeded in Transit";
itype: 11; icode: 0; sid:449;  classtype:misc-activity; rev:4;)
```

Figure 28 - Snort ICMP Source Quench Rule

As stated in the “Source of Trace” section above, the Norton Antivirus server and the border router were part of the corporate WAN and not a part of the local LAN. Even though they were considered to be trusted systems, they were external to the local network and thus was not set in Snort’s $HOME_NET variable, which lists internal IP addresses and ranges. This was done intentionally to ensure that in the event the frame relay or any remote systems were compromised, the traffic would still be monitored by Snort and the on-site analyst alerted to any suspicious activity.

The Snort alert, shown in fast alert mode, can be broken down as follows:

Alert Message Time 05/15-04:24:29.046034			
Alert Message Name [**] [1:473:1] ICMP redirect net [**]			
Alert Message Classification [Classification: Potentially Bad Traffic] [Priority: 2]			
Protocol {ICMP}	Source IP Address 172.19.1.1	Traffic Flow ->	Dest IP Address 192.168.79.23

Figure 29 - Snort ICMP Source Quench Alert Format

The associated WinDump data can be broken down as follows:

Alert Message Time 05/15-04:24:29.046034			
Source IP Address 172.19.1.1		Traffic Flow ->	Dest IP Address 192.168.79.23:
Protocol icmp:	ICMP Message Type redirect	ICMP Message ext.norton.server.net to net ext.norton.server.net for 192.168.79.23.1313 > ext.norton.server.net.38293: udp 16 (ttl 28, id 59357, len 44)	
Time to Live 254		IP Identification id 3331	Datagram Length len 56

Figure 30 – WinDump ICMP Packet Dump Format

2.2.3 Probability the Source Address was Spoofed

None. While it is not difficult to spoof ICMP packets, the frame relay would have to be compromised in order to receive a packet that is only one hop away (as evident by the ICMP time-to-live value of 254). This would imply that the malicious machine or router were actually on the same segment of the frame relay. It would also be possible, once the number of hops to the LAN machine were known, to craft packets with non-standard time-to-live values and spoof the WAN border router's IP address. These packets could be built in such a way as to appear to only be coming from a single hop away.

Because of the LAN audit trail provided by WinDump however, it was possible to build a filter to search for the original packet which the ICMP redirect messages claim to have initiated the communications. The original packet did exist as shown in the figure below:

```
04:24:28.977026 192.168.79.23.1313 > ext.norton.server.net.38293:  [udp sum ok] udp 16 (ttl 32, id
59357, len 44)
```

Figure 31 - WinDump UDP Packet Log

2.2.4 Description of Attack

In RFC 792, "INTERNET CONTROL MESSAGE PROTOCOL" (http://www.faqs.org/rfcs/rfc792.html pgs 12-13), we learn that the "ICMP redirect" message serves as a gateway or router's method to inform a host that there is a more efficient path to use in the future order to get packets to an intended destination. The router however, still forwards the initial packet on to its proper location. This message should only be sent by gateway devices.

The same document describes "ICMP time-to-live exceeded in transit", as a message sent to the originating host when a gateway finds a packet with a TTL value of "0" or when a remote host times out on fragment reassembly.

Staring with the Snort alerts, we are only told that the WAN border router is telling the LAN machine to redirect its traffic over and over again. The time that elapses while this is happening is approximately one second.

```
icmp: redirect ext.norton.server.net to net ext.norton.server.net for 192.168.79.23.1313 >
ext.norton.server.net.38293:  udp 16 (ttl 28, id 59357, len 44)
```

Figure 32 - ICMP Redirect Message Specific Information

Focusing on the ICMP redirect message-specific information provided by the WinDump packet logs, we add to the Snort alert information that the LAN machine was being told to redirect all traffic to the Antivirus server directly to that machine, as opposed to through the WAN border router. This implies that the router believes the two machines are on the same segment. We also learn that this message is in response to a UDP packet sent to port 38293, which a quick check of the Neohapsis ports list (http://www.neohapsis.com/neolabs/neo-ports/neo-ports.csv) tells us is the port used for Norton Anti-Virus Host Discovery. This is in line with traffic we expect to see traveling to that particular WAN machine.

Next we note that the IP identification and datagram length of the copied packet in the ICMP redirect message does not change throughout the entire series. We can also note the pattern of the time-to-live value of the copied packets. The time-to-live decrements with every ICMP redirect message. This implies that the WAN border router kept seeing the packet over and over again. When the time-to-live value finally expired (reached zero) WinDump logged the ICMP time-to-live exceeded in transit packet.

What appears to be happening is that the configuration of the WAN border router has changed. For some reason it now falsely believes that the Anti-virus server is part of the LAN and the following steps are taking place:

1. As a single packet from the LAN attempts to reach the Anti-virus server, it passes through the LAN border router and the time-to-live is decremented by one.
2. The packet reaches the WAN border router, where the time-to-live is decremented again.
3. The WAN border, incorrectly believing that the Anti-virus server is on the LAN, sends an ICMP redirect message to the originating LAN host.
4. The ICMP redirect message, since it is destined for the originating LAN host, passes through the LAN border router and the time-to-live is decremented by one. This gives us a time-to-live value of 254 in the ICMP packet.
5. The original redirected packet reaches the LAN border router and is decremented by one.
6. This router however, knows that the Anti-virus server is on the WAN and forwards the packet back to the WAN border router.
7. Return to step #2 until the time-to-live value is 0, then send an ICMP time-to-live exceeded in transit.

This erroneous configuration would indeed generate a pattern similar to the one shown here. This does not appear to be a direct attack. However, it is possible that this could be the result of a successful hack and reconfiguration of the WAN border router. The logs for the WAN border router were not available. As a side observation, this problem also created a denial of service situation for users on the LAN, whose access to the WAN were significantly reduced because of the volume of packets traveling back and forth.

2.2.5 Attack Mechanism

An ICMP redirect attack can be mounted for several reasons. By successfully convincing a host or router to use an alternate gateway first instead of its intended gateway, malicious users could sniff traffic off the wire before forwarding it along its way. A denial of service attack could be accomplished by reconfiguring a host or router to believe that external addresses are internal to its network, as was the case in this trace or to believe that packets needed to be routed to an internal, non-existent IP address.

2.2.6 Correlations

Winfreez is a sample tool which uses ICMP redirect messages to create a denial of service attack. As shown in the figure below (taken from Glenn Pitcher's post located at: http://lists.insecure.org/incidents/2002/Apr/0126.html), the Winfreez attack attempts to reroute the destination IP to a different IP. Additionally, the TTL value of the initial packet does not change and the IP ID does not remain constant. These are all significant differences from the information presented in this detect.

```
04/25/2002 00:23:17.764138 0:60:8:a:69:c1 0:60:8:93:91:c8 ip 70: xxx.xxx.173.2 > xxx.xxx.173.8: icmp:
redirect 10.13.0.1 to host  xxx.xxx.173.1 for xxx.xxx.173.8 > 10.13.0.1: icmp: echo request (ttl 126, id
38819, len 84) (ttl 128, id 63088, len 56)
04/25/2002 00:23:17.773256 0:60:8:a:69:c1 0:60:8:93:91:c8 ip 70: xxx.xxx.173.2 > xxx.xxx.173.8: icmp:
redirect 10.13.0.4 to host xxx.xxx.173.1 for xxx.xxx.173.8 > 10.13.0.4: icmp: echo request (ttl 126, id
39075, len 84) (ttl 128, id 63344, len 56)
04/25/2002 00:23:17.774036 0:60:8:a:69:c1 0:60:8:93:91:c8 ip 70: xxx.xxx.173.2 > xxx.xxx.173.8: icmp:
redirect 10.13.0.4 to host  xxx.xxx.173.1 for xxx.xxx.173.8 > 10.13.0.4: icmp: echo request (ttl 126, id
39331, len 84) (ttl 128, id 63600, len 56)
```

Figure 33 - Winfeez Packet Dump

Information on ICMP Redirect Denial of Service Attacks and Windows NT:
Microsoft Product Support Services. "ICMP Redirect Attack Causes Windows NT Server and Workstation to Hang (Q225344)" 11 June 2002. URL: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q225344 (24 June 2002).

Information on ICMP Redirect Behavior:
Farrow, Rick. "ICMP Stands for Trouble" 15 September 2000. URL: http://www.networkmagazine.com/article/NMG20000829S0003 (24 June 2002).

Microsoft Product Support Services. "Explanation of ICMP Redirect Behavior (Q195686)" 10 August 2001. URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;Q195686 (24 June 2002).

Postel, J. "RFC 792: INTERNET CONTROL MESSAGE PROTOCOL" September 1981. URL: http://www.faqs.org/rfcs/rfc792.html (23 June 2002).

Mitre Common Vulnerabilities and Exposures listings correlating to ICMP Redirect:
CVE-1999-0265 - ICMP redirect messages may crash or lock up a host.
CAN-1999-1254 - Windows 95, 98, and NT 4.0 allow remote attackers to cause a denial of service by spoofing ICMP redirect messages from a router, which causes Windows to change its routing tables.

2.2.7 Evidence of Active Targeting

There is no evidence of malicious targeting in this detect.

2.2.8 Severity

Criticality: 1 – The LAN machine is an end-user workstation.

Lethality: 1 – Had this been a successful, targeted attack against the local LAN system, it would not have been able to communicate with one of several Norton Anti-virus servers. There were other Anti-virus servers which acted as backups.

System Countermeasures: 4 – This end-user workstation was a Windows 2000 operating system which was patched regularly, running only necessary services, and up-to-date anti-virus software. A maximum security rating of 5 for the system could be achieved by implementing a host-based firewall.

Network Countermeasures: 1 – There was no firewall in place. The router allowed most traffic through, as it assumed a trusted relationship over the frame relay to the remote site. There was a sniffer and a network intrusion detection system running on the local network, but these were not active countermeasures.

Severity = (1+1) – (4+1) = **-3**

2.2.9 Defensive Recommendations

If practical, a host-based firewall should be implemented. A network-based firewall should absolutely be implemented.

2.2.10 Multiple Choice Test Question

According to the ICMP standards, who is authorized to send ICMP redirect messages?

a. Firewalls
b. Hosts
c. Gateways
d. All of the Above

The correct answer is: c. Gateways.

3 Analyze This

3.0 Executive Summary

Following is a report and summary analysis of the University's combined intrusion detection system logs over the five-day periods from 16 June through 20 June, 2002 and 4 April through 8 April, 2002. In total, there were over 6.6 million events logged by the intrusion detection systems during this period.

Provided in this report can be found the critical alerts of interest and information about these alerts, the most active IP addresses internally and externally, as well as the most common services being logged. Amplifying information is provided about a number of internal and external IP addresses which seem to be demonstrating suspicious activity on your network. Also contained within this report is a profile of common host and network services, inferred from the event logs. This list should be provided to your system administrators to validate that these services are indeed approved and operating legitimately on the given hosts.

Finally, a brief description of the analysis process is provided for your review. Should you have any questions concerning any of this information, please do not hesitate to contact us.

3.1 Files to Be Analyzed

Out of Spec Files**	Alert Files	Scan Files
oos_Apr.4.2002.gz oos_Apr.5.2002.gz oos_Apr.6.2002.gz oos_Apr.7.2002.gz oos_Apr.8.2002.gz	alert.020616.gz alert.020617.gz alert.020618.gz alert.020619.gz alert.020620.gz	scans.020616.gz scans.020617.gz scans.020618.gz scans.020619.gz scans.020620.gz

** These were the most recent Out of Spec files available.

3.2 Network Profile

Following is a profile of services believed to be running on the network, as inferred from the type and number of alerts destined or originating from each host.

NETWORK AND HOST SERVICE PROFILE (based upon generated alerts)		
SUBNET	**HOSTS**	**SERVICES**
MY.NET.1	2,3,4,5	DNS
MY.NET.2	29	MSSQL
MY.NET.3	54 (Brian B.)	FTP
MY.NET.5	19, 67	FTP, MSSQL, SSH
	29	HTTPS

	44	FTP, MSSQL, SMTP
	80	DNS
	20, 64	MSSQL
	82	DNS, Unix RPC
	92, 95-96	HTTP
MY.NET.6	7	FTP, Telnet, rsh
	60	Unix RPC
	63	FTP
	34, 35, 40, 47	SMTP
MY.NET.11	4	FTP
MY.NET.53	33	MSSQL
	58	SMTP
MY.NET.60	11, 16, 38-39	FTP
MY.NET.70	40, 50 (HelpDesk), 69 (Beetle.UCS), 90, 181, 215, 218, 225	FTP
	133,134	DNS
	140, 177, 181	MSSQL
	10	FTP, MSSQL, SSH
	80, 149, 180, 198	Gnutella
MY.NET.75	102, 212	FTP, MSSQL
	114	DNS
MY.NET.86	19	FTP
	108	Gnutella
MY.NET.88	163	FTP
	88	DNS
	102, 212	MSSQL
	215, 240	Gnutella
	245	DNS, Unix RPC
MY.NET.97	73, 159	DNS
	16, 41, 80, 115, 129, 158, 170, 177, 196, 197	Gnutella
MY.NET.98	111, 146, 149, 188, 194, 228, 238, 242	Gnutella
MY.NET.99	85	FTP
	81	Unix RPC
MY.NET.100	157	Gnutella
	158	FTP
	165 (CS WEBSERVER)	FTP, SSH, HTTP, Finger
	203	Unix RPC
	217	SMTP (Queso), HTTP
	230	SMTP

MY.NET.102	2, 4, 9-10, 16-19, 24, 31, 37, 41-42, 51-.59, 64, 67, 72, 74, 76, 84, 87-88, 92, 94-95, 97, 100-118, 122, 125, 128, 132, 135, 141, 145, 153-165, 172-174, 200-217, 228, 234, 236-237, 239, 248-252	FTP
MY.NET.105	10, 120	FTP
MY.NET.109	96, 101	Unix RPC
MY.NET.110	92	HTTP
	110	SMTP
MY.NET.111	21	FTP
	140	SMTP
MY.NET.114	56	FTP
MY.NET.116	105	Gnutella
MY.NET.117	10	FTP
	25	DNS, Unix RPC
MY.NET.121	30	DNS
MY.NET.130	86	FTP, MSSQL
MY.NET.132	10	FTP
MY.NET.137	7	DNS
MY.NET.139	230	SMTP
MY.NET.143	84	FTP
MY.NET.145	74	FTP
MY.NET.150	98	MSSQL
	209	Gnutella
MY.NET.151	14	FTP
MY.NET.153	108, 168, 179	FTP
	142, 160	Gnutella
	203	Unix RPC
MY.NET.154	27	Unix RPC
MY.NET.157	108	Gnutella
	243, 247, 253	FTP, MSSQL
	250	Infected with NIMDA
	242	FTP, TFTP or BackGate Trojan, IRC
	248	HTTP
	252	FTP, MSSQL, IRC
MY.NET.158	53, 75	FTP, MSSQL
MY.NET.162	67, 235,	FTP
	90	FTP, Gnutella, HTTP, PCAnywhere
MY.NET.163	135	DNS
MY.NET.165	20	FTP
MY.NET.167	2	MSSQL

MY.NET.168	142	Gnutella
MY.NET.178	168, 199	MSSQL
MY.NET.179	78, 80	SMTP
MY.NET.182	135, 56, 94	Gnutella
MY.NET.185	48	MSSQL, Gnutella
MY.NET.253	20, 105	FTP
	20, 41	MSSQL
	41, 43, 51, 53	SMTP
	114	MSSQL, HTTP

Figure 34 – Network and Host Service Profile

Following is a list of ports used to determine the above Network profile, in conjunction with review of the alerts to ensure applicability.

- FTP Servers (Ports: 20, 21)
- Telnet Servers (Port: 23)
- Web Servers (Ports: 80, 8080, 1080)
- SMTP Servers (Port: 25)
- Printers (Port: 515) - There were a total of 9123 printer alerts during the reporting period. 9091 of these alerts originated from IP 211.114.0.252 as it scanned multiple subnets for printers. 32 of these alerts originated from the broadcast address of 255.255.255.255, with a source port of 31337 (spelling "eleet" in the hacker community). Because responses from printers would not have normally generated alerts, it is impossible to gauge the number of actual printers on the network.
- MSSQL Servers (Port: 1433)
- DNS Servers (Port: 53)
- Other Servers of Interest
    - CS Webservers
    - beetle.ucs
    - MY.NET.3 54 & 64 (Bryan B.)
    - HelpDesk

3.3 Detects By Severity or Number of Occurrences

Out of the 6.6 million events logged (including scans and out-of-spec packets) the university IDS captured 309 unique alerts over this five-day period. The complete list of alerts can be found in Appendix B. The following are what I believe to be the most significant alerts.

3.3.0 Watchlist 000222 NET-NCFC

Number of alerts: 15,677 (16th most frequent alert)
Number of Unique Source IPs: 54
Number of Unique Destination IPs: 59

Destination Ports: 54 different destination ports (one per source IP). Primarily ephemeral with a number of high ports in the 20-64K range. Well-known destination ports were: 113 (ident authorization service), 21 (FTP and trojans), 23 (telnet and trojans), 25 (SMTP and trojans), 80 (HTTP and trojans),

Description: Watchlist alerts reference hosts which are known to be problematic. This watchlist in particular monitors traffic originating from the 159.226.0.0/16 network which belongs to the Institute of Computing Technology Chinese Academy of Sciences which is located in Bejing, China.

In addition to the Watchlist alert, 45 other events were logged from this network with the following six message types:

- CS Webserver – external web traffic
- INFO Inbound GNUTella connect request
- MISC source port 53 to <1024
- NMAP TCP ping!
- SCAN SYN ******S*
- SCAN UDP

Defensive Recommendations: Monitoring of this block of IPs should continue. Consideration should be given as to whether the entire IP block could be restricted from accessing the University's servers.

The University security policy should also be analyzed as to whether to allow inbound GNUTella requests. If improperly configured, file-sharing tools such as GNUTella, Kazaa and Morpheus can allow malicious users access to all of the files on a user's system.

Correlations: This alert has been documented in the following practicals:
Lorraine Weaver, http://www.giac.org/practical/Lorraine_Weaver_GCIA.zip
Hee So, http://www.giac.org/practical/Hee_So_GCIA.doc
Martha Flick, http://www.giac.org/practical/Martha_Flick_GCIA.zip
Paul Asadoorian, http://www.giac.org/practical/Paul_Asadoorian_GCIA.zip
Chris Lethaby, http://www.giac.org/practical/Chris_Lethaby_GCIA.zip

3.3.1 Watchlist 000220 IL-ISDNNET-990517

Number of Alerts: 12,427 (19th most frequent alert)
Number of Unique Source IPs: 55
Number of Unique Destination IPs: 55
Destination Ports: Various ranging from 1030 to 4789, all ephemeral with the exception of ports 25 (SMTP and trojans) and 80 (HTTP and trojans). Other well-known ephemeral ports attempting to connect to were: 6347 (Gnutella) and 1214 (KaZaa/Morpheus/Grokster).

Description: All 55 unique IP addresses correlate to an Israeli company called ISDN Net Ltd in the 212.176.0.0/17 subnet. These hosts are well known for malicious activity. It is interesting to note that each source IP only connected to a single destination IP. This is indicative of packet crafting.

In addition to the Watchlist alert, 73 other events were logged from this network with the following six message types:

- ICMP Destination Unreachable (Communication Administratively Prohibited)
- INFO FTP anonymous FTP
- INFO Inbound GNUTella Connect request
- MISC source port 53 to <1024
- SCAN SYN ******S*
- suspicious host traffic

Defensive Recommendations: Monitoring of this block of IPs should continue. Consideration should be given as to whether the entire IP block could be restricted from accessing the University's servers.

The University security policy should also be analyzed as to whether to allow inbound GNUTella requests. If improperly configured, file-sharing tools such as GNUTella, Kazaa and Morpheus can allow malicious users access to all of the files on a user's system.

Correlations: This alert has been well documented by other GCIA practicals, such as:
Lorraine Weaver, http://www.giac.org/practical/Lorraine_Weaver_GCIA.zip
Hee So, http://www.giac.org/practical/Hee_So_GCIA.doc
Martha Flick, http://www.giac.org/practical/Martha_Flick_GCIA.zip
Paul Asadoorian, http://www.giac.org/practical/Paul_Asadoorian_GCIA.zip
Chris Lethaby, http://www.giac.org/practical/Chris_Lethaby_GCIA.zip

3.3.2 Attempted Sun RPC high port access / SUNRPC highport access!

Number of Alerts: 731 (41st most frequent alert) / 213 (54th most frequent alert)
Number of Unique Source IPs: 274 (4 Internal: MY.NET.100.230, MY.NET.153.164, MY.NET.6.49, MY.NET.99.120) / 21 (7 Internal: MY.NET.100.40, 85, 203 & 227, MY.NET.109.101, MY.NET.99.85 & 174)
Number of Unique Destination IPs: 9 (MY.NET.100.203, MY.NET.109.101, MY.NET.109.96, MY.NET.117.25, MY.NET.153.203, MY.NET.154.27, MY.NET.5.82, MY.NET.6.60, MY.NET.88.245) / 16 (MY.NET.60.16, MY.NET.82.54, MY.NET.83.13, 15, 20, 23, MY.NET.99.51, 81, 120, MY.NET.100.21, 65, 230, MY.NET.139.42, MY.NET.179.78, MY.NET.253.53)
Destination Ports: 32771 – This port is typically used for remote procedure calls on Sun Unix systems and is often referred to as the rpc.ghost portmapper port.

Description: Internet Security Systems suggests that many Sun operating system boxes listen on this port for portmapper calls (http://www.iss.net/security_center/advice/Exploits/Ports/32771/default.htm). Portmapper provides a list of which remote services are running on which ports. This is a fairly common exploit, which is often not blocked at the firewall because it is an ephemeral port.

Portmapper runs on port 111 by default. Checking all logged entries for Sun RPC highport access from source port 111, two alerts are returned. Both of these alerts originated from MY.NET.100.230 and were bound for MY.NET.100.203.

A significant amount of this traffic appears destined for machines that have been inferred to be DNS servers. Traffic bound for these machines seems to be originating from port 53 (DNS).

Unlike many other practicals, no traffic could be found originating from port 4000, inferring ICQ connections.

Defensive Recommendations: Each of the 9 internal destination hosts causing this alert should be checked for whether they respond to requests on port 32771 and whether or not they are running portmapper. If they indeed are running portmapper and there is a necessity for it, these machines should be thoroughly checked to ensure no compromises have occurred. Internal hosts communicating to port 32771 should be monitored for suspicious traffic.

Externally, UDP traffic to port 32771 should not be allowed through the firewall when it has not been initiated by a local machine. This is assuming that the firewalls in place allow stateful detection.

Correlations: This alert has been documented in the following other GCIA practicals, such as:
CVE-1999-0189, Solaris rpcbind listens on a high numbered UDP port, which may not be filtered since the standard port number is 111.
Guy Bruneau, http://www.giac.org/practical/Guy_Bruneau_GCIA.doc
Jeffrey Holland, http://www.giac.org/practical/Jeff_Holland_GCIA.doc
Kevin Orkin, http://www.giac.org/practical/Kevin_Orkin_GCIA.doc
Lenny Zeltser, http://www.giac.org/practical/Lenny_Zeltser.htm
PJ Goodwin, http://www.giac.org/practical/PJ_Goodwin_GCIA.doc

3.3.3 SNMP public access

Number of Alerts: 10,271 (20th most frequent alert)
Number of Unique Source IPs: 36 (94% these alerts originated from internal machines)
Number of Unique Destination IPs: 157
Destination Ports: 161 (SNMP – 6th most frequent destination port)

Description: This non-standard alert implies an attempt to access SNMP services via the default “public” access string. According to Cisco (http://www.cisco.com/warp/public/707/cisco-malformed-snmp-msgs-pub.shtml), by using public community strings, it is possible to bypass SNMP access controls. Cisco also states that public community string access vulnerabilities can be used to initiate denial of service attacks.

Defensive Recommendations: Public access strings should not be allowed for SNMP. Wherever possible, they should be disabled. If it is necessary to have public strings, it is recommended that the University change the string from the default.

Correlations:
CAN-2002-0012, Vulnerabilities in a large number of SNMP implementations
CAN-2002-0013, Vulnerabilities in the SNMPv1 request handling
Scott Shinberg, http://www.giac.org/practical/Scott_Shinberg.doc
PJ Goodwin, http://www.giac.org/practical/PJ_Goodwin_GCIA.doc
Markus DeShon, http://www.giac.org/practical/Markus_DeShon.html

3.3.4 connect to 515 from outside

Number of Alerts: 9,115 (22nd most frequent alert)
Number of Unique Source IPs: 2 (211.114.0.252, 255.255.255.255)
Number of Unique Destination IPs: 7,324
Destination Ports: 515 (printer, lpdw0rm, Ramen trojan)

Description: IP 211.114.0.252 appears to be scanning a large number of subnets on the University’s network. This could be a scan for printers, printer vulnerabilities common to Red Hat Linux operating systems or simply a network-mapping scan.

Defensive Recommendations: Connections to port 515 from sources external to the network should be blocked at the firewall. Also, anything sent to a broadcast address from the outside world should be stopped by the router or firewall.

Correlations:
CERT Advisory, http://www.cert.org/advisories/CA-2000-22.html
CAN-2000-0917, Format string vulnerability in LPRng
Jeffrey Holland, http://www.giac.org/practical/Jeff_Holland_GCIA.doc
Lorraine Weaver, http://www.giac.org/practical/Lorraine_Weaver_GCIA.zip
Scott Shinberg, http://www.giac.org/practical/Scott_Shinberg.doc

3.3.5 SMB Name Wildcard

Number of Alerts: 85,891 (5th most frequent alert)
Number of Unique Source IPs: 4,274 (96% originated from external sources)
Number of Unique Destination IPs: 4,170
Destination Ports: 137 (NetBIOS)

Description: SMB traffic can be used by malicious users to elicit Windows domain information such as user and host identification.

Defensive Recommendations: Internally, a significant number of these alerts were logged between MY.NET.152.216 – MY.NET.11.7 and MY.NET.11.6 – MY.NET.152.179. These four hosts should be checked for possible NetBIOS compromises. If NetBIOS is not a necessity on these machines, it should be disabled.

Unless providing external file-sharing services, inbound traffic on ports 135-139 should be stopped at the firewall.

Correlations:
Jeffrey Holland, http://www.giac.org/practical/Jeff_Holland_GCIA.doc
Lorraine Weaver, http://www.giac.org/practical/Lorraine_Weaver_GCIA.zip

3.3.6 UDP SRC and DST outside network

Number of Alerts: 104,095 (4th most frequent alert)
Number of Unique Source IPs: 66
Number of Unique Destination IPs: 4 (229.55.150.208, 233.28.65.148, 233.28.65.61, 239.255.255.250)
Destination Ports: 1345 (vpjp, sysmond), 1900 (ssdp), 5779 (unknown)

Description: This traffic appears to be spoofed UDP traffic in which someone inside the network is attempting to appear as the Georgia Institute of Technology, Internet Assigned Numbers Authority (IANA), The University of Freiburg in Germany, and Yahoo! Broadcast Services.

All four of the destinations IP addresses correlate to multicast network addresses.

In his GCIA practical, Scott Shinberg suggests that this traffic may in fact be legitimate streaming media which has been assigned a valid multicast address by the University's multicasting system.

Defensive Recommendations: In order to keep users inside the network from successfully spoofing IP addresses and launching external network denial-of-service attacks, it is strongly recommended that egress filtering be implemented, if it is not already.

Use of IDS tagging or a network sniffer would provide full packet decodes of the spoofed network traffic, which could be used to help track down the offending user(s) based upon router hops and time to live values.

If the university does have a multicast system, then given the significant number of alerts, the university may wish to consider adding the multicast addresses to the home network in Snort to reduce false positives.

Correlations:
Jeffrey Holland, http://www.giac.org/practical/Jeff_Holland_GCIA.doc
Lorraine Weaver, http://www.giac.org/practical/Lorraine_Weaver_GCIA.zip
Scott Shinberg, http://www.giac.org/practical/Scott_Shinberg.doc

3.4 Top Talkers

Top 10 Source IP Addresses		
Rank	**IP Address**	**# of Alerts**
1	65.107.178.34	376510
2	65.120.161.122	239162
3	65.114.45.134	235415
4	207.162.20.121	194209
5	MY.NET.5.89	175023
6	212.194.136.49	157498
7	80.143.254.161	136486
8	67.68.163.9	134749
9	MY.NET.60.43	112417
10	66.205.196.200	88485

The number of times a particular machine was logged as the Source IP Address.

Top 10 Destination IP Addresses		
Rank	**IP Address**	**# of Alerts**
1	MY.NET.162.90	2827200
2	MY.NET.157.248	860867
3	MY.NET.157.252	708987
4	MY.NET.157.242	155881
5	233.28.65.148	67848
6	MY.NET.157.243	58290
7	MY.NET.130.69	50793
8	MY.NET.140.9	44897
9	MY.NET.100.165	37337
10	233.28.65.61	31533

The number of times a particular machine was logged as the Destination IP Address.

Top 10 Source IP Addresses By Number of Unique Destination IP Addresses		
Rank	**IP Address**	**# of Unique IPs**
1	210.83.17.179	26086
2	67.242.168.58	25230
3	202.224.237.106	24028
4	130.75.89.31	21500
5	63.173.94.20	18208
6	217.150.33.178	18202
7	212.182.119.141	17339
8	132.235.75.1	15973
9	68.11.184.35	15478
10	149.169.44.155	14476

The number of unique destination IP addresses for each IP.

Top 10 Destination IP Addresses By Number of Unique Source IP Addresses		
Rank	**IP Address**	**# of Unique IPs**
1	MY.NET.162.90	14080
2	MY.NET.100.165	7506
3	MY.NET.100.157	4895
4	MY.NET.70.149	4854
5	MY.NET.70.80	3816
6	MY.NET.1.3	3724
7	MY.NET.185.48	3386
8	MY.NET.153.142	3149
9	MY.NET.1.4	2984
10	MY.NET.1.5	2896

The number of unique source IP addresses seen for each IP.

Top 10 Destination Ports			
Rank	Port	Typical Service on this Port	# of Alerts
1	6346	Gnutella	2773688
2	30200	Unknown	851160
3	80	http	363772
4	6970	GateCrasher Trojan	311450
5	2062	ICG SWP Port	188998
6	161	SNMP	183521
7	3842	Unknown	157436
8	1082	AMT-ESD-PROT	134978
9	137	NetBIOS	106092
10	5779	Unknown	99381

The number of alerts targeted at a specific destination port.

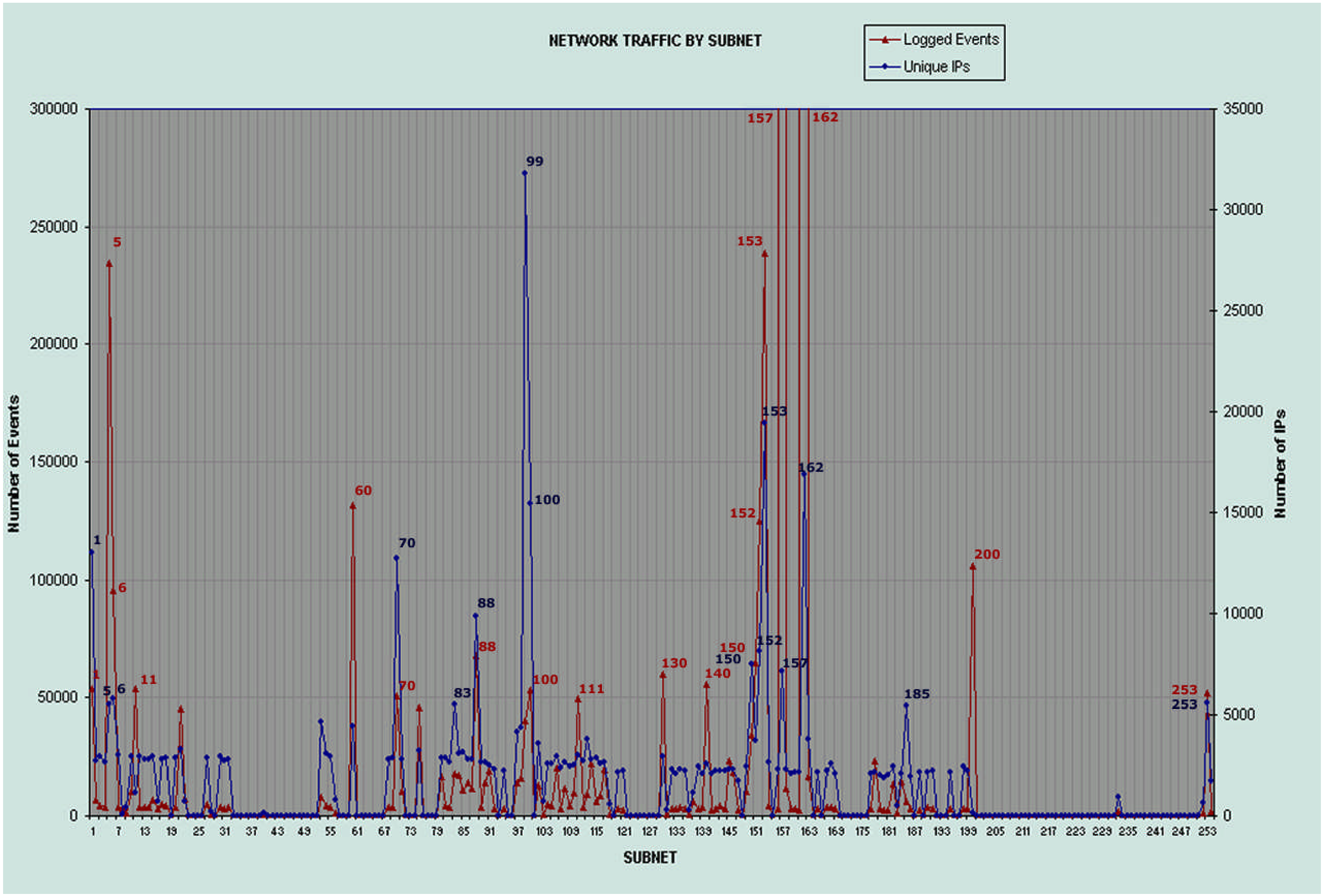
3.5 *External Source Information*

IP Address	Information	Reason for Inclusion
65.107.178.34	XO Communications (NET-XOXO-BLK-15), 1400 Parkmoor Avenue, San Jose, CA 95126-3429, US Netname: XOXO-BLK-15 Netblock: 65.104.0.0 - 65.107.255.255 Maintainer: XOXO Coordinator: DNS and IP ADMIN (DIA-ORG-ARIN) hostmaster@CONCENTRIC.NET Phone: (408) 817-2800 Fax: (408) 817-2630	This IP was the top alert-generating source IP address, producing a total of 376,510 alerts. Communications from this IP were logged to the #1 and #2 destination IPs as well as the #1 machine to which the most source IPs were seen. Additional alerts from this IP were Inbound GNUTella connect requests. Al traffic was bound for either ports 30200 or 6346.
210.83.17.179	inetnum: 210.82.0.0 - 210.83.255.255 netname: CNCNET descr: China Netcom Corp. descr: New Telecommunication Carrier Based on IP Backbone country: CN	This IP sent to the largest number of internal IP addresses, which was a total of 26,086. It is strongly recommended that

	admin-c: YZ213-AP tech-c: YZ213-AP mnt-by: APNIC-HM mnt-lower: MAINT-CN-ZM28 changed: hostmaster@apnic.net 20001011 changed: hm-change@apnic.net 20020703 source: APNIC person: yanping zhao address: 15/F, Building A, Corporate Square,No address: 35 Financial Street,Xicheng District, address: Beijing country: CN phone: +86-010-88093588 fax-no: +86-010-88091442 e-mail: tech-group@china-netcom.com nic-hdl: YZ213-AP mnt-by: MAINT-CN-ZM28 changed: daihy@china-netcom.com 20020618 source: APNIC	this net block 210.82.0.0/16 and 210.83.0.0/16 be put on a watchlist for suspicious activity.
65.120.161.122	CENTRAL DISTRIBUTION INC (NETBLK-Q0115-65-120-161-0), 2832 ROE LANE, KANSAS CITY, KS 66103, US Netname: Q0115-65-120-161-0 Netblock: 65.120.161.0 - 65.120.161.127 Coordinator: Donakey, Coy (CD733-ARIN) coy.donakey@centraldistribution.com Phone: 913-677-1666	This IP ranked #2 on the list of top alert-generating source IPs. The 239,162 alerts originating from this IP were logged as traffic to internal hosts deemed suspicious. The traffic was primarily bound for ephemeral port 30200.
65.114.45.134	SELWAY PARTNERS (NETBLK-Q1105-65-114-45-128), 52 FOREST AVE 2ND FLOOR, PARAMUS, NJ 07652, US Netname: Q1105-65-114-45-128 Netblock: 65.114.45.128 - 65.114.45.159 Coordinator: Paulison, Harry (HP178-ARIN)	Ranking #3 on the list of top source IPs, this IP logged 235,415 suspicious host alerts to the #2 ranked destination host. All traffic was bound for port 30200.

	kristeena@comserv1.com Phone: 973-812-3832	
207.162.20.121	Universite du Quebec (NETBLK-UQUEBEC-CA), 2875 Bld. Laurier, Ste-Foy, Quebec G1V 2M3, CA Netname: UQUEBEC-CA Netblock: 207.162.1.0 - 207.162.50.255 Coordinator: Goutier, Bernard (BG181-ARIN) Bernard_Goutier@UQSS.UQUEBEC.CA Phone: (418)657-4469 FAX: (418)657-2132	The #4 ranked top alerting source IP logged 194,209 alerts to the #3 ranked destination IP. All alert messages were “suspicious host traffic” and were bound for ports 2062 and 3899.

3.6 Link Graph



3.6.0 Link Graph Analysis

When operating a network as large as the university's, it is often necessary to be able to quickly assess large volumes of data in order to rapidly determine potential problem areas. By presenting the number of unique IP addresses seen by each subnet in perspective with the number of IDS alert and scan events logged graphically, our attention is immediately drawn to specific sub-nets of computers. In the graph above, subnets seeing over 5,000 unique IP addresses or logging over 50,000 alerts over the 5-day period were marked as potentially containing compromised hosts. It should be noted that it was necessary to scale the events axis of the graph in order to better understand the data. Thus, the two highest alerting subnets, 157 and 162, are shown as being off the graph.

The theories behind this link graph are that subnets offering common services, such as HTTP, FTP, Telnet, SMTP, POP-3, and DNS will likely show up on the graph with a large number of unique IP addresses (seen in dark blue). If a subnet registers a spike on the number of unique IPs but should not be offering large-scale common services, we are immediately alerted to investigate. Without knowing anything about the overall network and looking at just the unique IPs seen, our attention should be drawn to subnets 1, 5, 6, 70, 83, 88, 99, 100, 150, 152, 153, 157, 162, 185 and 253.

When the quantity of logged IDS event data is graphed in perspective, we narrow our focus even further. This is not to say that all networks generating alerts should not be investigated. A small group of 10 "trojan server activity" messages could be far worse than 1,000 Windows Pings. For quick reference of the largest potential problems though, we often fall back on the adage that "more is worse" and look to numbers of alerts. Thus looking at number of alerts alone, our attention would be drawn to subnets 1, 5, 6, 11, 60, 70, 88, 100, 111, 130, 140, 150, 152, 153, 157, 162, 200 and 253.

Theoretically, if we are seeing a large number of unique IPs into a subnet AND a large number of alerts, we are more likely to have a problem. Combining these two lists together, our focus is drawn to subnets: 1, 5, 6, 70, 88, 100, 150, 152, 153, 157, 162, and 253. Correlating this with our detailed analysis data, ALL of the subnets represented in this combined list are noted as having one or more possible compromised hosts in the *"Insights into Internal IP"* section below.

Obviously this is not a foolproof scheme. However, it can be very helpful in narrowing focus down to specific problem areas in a relatively short amount of time. Instead of subnets, it can be very helpful to graph similar data for all hosts within a specific subnet. This can aid in narrowing down problems from a subnet to an individual host.

3.7 Insights Into Internal Machines

3.7.0 IP: MY.NET.1.3

During the reporting period, this IP logged 20,949 alerts. This machine should be carefully checked for legitimate DNS usage. All but 300 of the alerts involve either a source or destination port (or both) of 53. This includes possible myserver activity, over 14,000 UDP port 53 alerts, and the receiver of possible Red Worm activity (also to port 53).

3.7.1 IPs: MY.NET.5.20 / MY.NET.5.64

Based upon the type and number of alerts logged, these web and ms-sql servers appear to be compromised to allow remote execution of cmd, ms-sql command shells, and be susceptible to Unicode attacks. It is recommended that these machines be taken off the network and all available logs for them be carefully examined. Additionally, complete virus and Trojan scans should be run. File monitoring software, such as Tripwire, should be implemented on visible external hosts to ensure that administrators are made aware if files have been altered without authorization.

3.7.2 IP: MY.NET.5.29

This IP appears to be either offering up information or have a virus installed on port 137. 902 of the 1,402 alerts logged revolve around port 137 access (primarily SMB Name Wildcard). This machine is receiving a large number of ICMP Destination Unreachable and MISC Large ICMP messages, which could be indicative of ICMP tunnel communications or IP spoofing. NetBIOS should be disabled if unnecessary and should be blocked at the firewall.

3.7.3 IP: MY.NET.5.83

The conversations originating from MY.NET.70.177 bound for this IP logged an average of 8 SMB public access alerts on port 161 every hour from 00:16 on the 16th through 13:17 on the 17th. The source port of the packets never changed, implying either packet crafting or a single 37-hour connection. The relationships between these two machines should be evaluated to determine if NetBIOS is running and is a necessary service. Both of these machines should be checked for virii and Trojans. Possible SubSeven Trojan activity was briefly logged between this IP and MY.NET.5.42, which is also on the list of compromised internal machines.

Another interesting note from the scan events shows that this IP regularly scanned machines on ports 7938 and 7937. These ports imply that this server may be running NetWorker backup and recovery Server software, release 5.5 or later.

3.7.4 IP: MY.NET.6.16

During the reporting period, this IP logged 40 SMB Name Wildcard Access events from 6 different external IPs. High ephemeral port traffic bound for local port 8765 (Ultraseek HTTP) originated from the NET-NCFC watchlist. Traffic to this same port was later logged from IP 198.200.181.204 as possible Trojan server activity (because it originated from port 27374). While no common Trojans are listed as using this port, it is possible that a modified Trojan or some other exploit could be running on this system. This port is also the default for the Inktomi search engine. NetBIOS should be disabled where unnecessary.

3.7.5 IP: MY.NET.6.34

87.5% of the alerts to this IP involved ICMP communications. Primarily, these were miscellaneous large ICMP packets and destination unreachable messages. All of the ICMP events logged were from sources external to the network implying one of two things: 1. This IP address is being spoofed or 2. This IP is actually attempting to connect to a large number of external hosts to which it is not allowed to connect. Given the lack of correlating scan events, I am more inclined to believe the former.

The remainder of the traffic to this IP revolved around port 25 (SMTP). If this machine is offering SMTP services, logs and configuration should be checked for malicious use. If not, Trojans and virii should be scanned for thoroughly.

3.7.6 IP: MY.NET.157.248

This web server appears on the suspicious host traffic list. It is difficult to analyze exactly what part of the traffic is suspicious without knowing more about the non-standard “suspicious host” rule that seems to be prevalent throughout these alerts. If we assume, based upon the volume of suspicious host alerts, that all traffic is logged when a machine is considered suspicious, then a large deal of IP spoofing can be inferred from packets returning from various web server IP addresses with no initiating requests. Early in the logs, this machine appears to be scanning multiple subnets in the range of 18.29 for web servers.

Other anomalous traffic to this machine shows IP 161.58.90.250 logged a significant number of port 80 traffic with this machine as well as a number of IIS Unicode attacks. Additionally, a large number of packets destined for port 30200 from IPs 65.120.161.122, 65.114.45.134, and 65.107.178.34 as well as traffic to ports 3196 and 30201 traffic from 80.8.81.239 were logged. Given the number of connections, but not knowing of any service or Trojan that typically runs on these ports, I would recommend that this machine be thoroughly checked for server software and Trojans running on these ports.

3.7.7 IP: MY.NET.162.90

Apparently this IP has come under previous scrutiny as it is on the “suspicious host alert list”. Assuming that the host’s services are legitimate, the system does not appear to be compromised. It should be noted that this IP saw the highest number of unique source IP addresses in the log files. I believe this is because the host appears to be receiving the backscatter from IP spoofing, as is evident by the significant number of ICMP Source Quench, Host and Network Unreachable alert messages from a plethora of networks.

3.7.8 Additional IPs to Review

In addition to the above-mentioned recommendations, the following IPs should be reviewed for possible compromises:

SUBNET	HOSTS
MY.NET.1	3
MY.NET.10	55
MY.NET.100	165
MY.NET.110	92
MY.NET.111	140
MY.NET.116	126
MY.NET.117	25
MY.NET.132	37
MY.NET.133	20
MY.NET.143	84
MY.NET.144	52
MY.NET.149	15
MY.NET.150	198, 232, 241
MY.NET.151	95
MY.NET.152	13, 15, 157, 158, 159, 16, 160, 162, 165, 166, 169, 170, 171, 174, 175, 176, 178, 179, 18, 183, 185, 186, 19, 20, 21, 216, 22, 245, 246, 248, 250, 251, 45, 46
MY.NET.153	108, 141, 142, 144, 146, 147, 148, 149, 150, 152, 153, 154, 159, 160, 162, 163, 164, 165, 166, 167, 169, 172, 174, 175, 176, 179, 180, 184, 185, 186, 188, 189, 193, 194, 195, 196, 197, 198, 199, 202, 203, 208, 209, 210, 211
MY.NET.157	250
MY.NET.16	42
MY.NET.163	235
MY.NET.167	87
MY.NET.168	98
MY.NET.178	168, 199, 76
MY.NET.179	35, 78
MY.NET.195	245
MY.NET.199	146, 72
MY.NET.2	177
MY.NET.253	114, 125, 24, 43, 52
MY.NET.53	160, 33, 51
MY.NET.6	40, 45, 48, 49, 50, 51, 52, 53, 60, 7
MY.NET.60	10, 151, 43
MY.NET.70	134, 177, 38
MY.NET.75	114
MY.NET.84	178
MY.NET.88	162, 235, 245
MY.NET.90	15
MY.NET.97	216, 221
MY.NET.98	238, 254
MY.NET.99	238, 254

Figure 35 - Recommended IPs for Review

3.8 Analysis Process

I began my analysis by downloading the files from http://www.incidents.org/logs. Right off the bat, I noticed something odd. The alerts data from 16 June to 20 June jumped exponentially in terms of the amount of data. This is shown in the chart below.

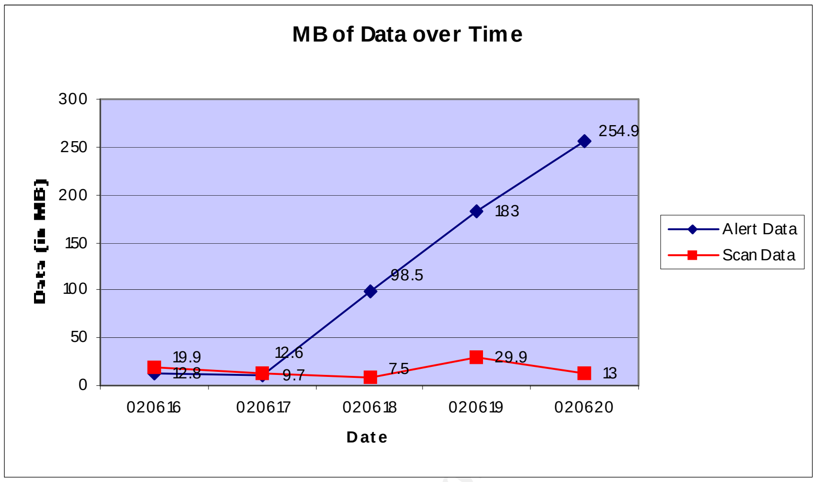


Figure 36 - Alert and Scan Data Over Time

I checked the calendar to see the specific days of the week and noted that many schools were preparing to let out for the summer, if they had not already. This could mean plenty of eager youth with too much time on their hands. These alert and scan file dates fell on a Sunday through Thursday. The largest set of scan data fell on Wednesday, followed by the largest amount of alert data on Thursday.

Looking at over 500MB of alert data seemed a pretty daunting challenge, but I was up to the task. I quickly scanned through the several of the files to see what I was dealing with. The alerts files, which took up by far the most space, were created with Snort in fast alert mode in the format of:

Timestamp	Alert	Source / Direction / Destination
06/16-00:00:02.972487	[**] ICMP Echo Request L3retriever Ping [**]	MY.NET.152.159 -> MY.NET.11.7

Figure 37 - Snort Fast Alert Output Sample

The scan files were standard Snort portscan output:

Timestamp	Source / Direction / Destination	Scan Type
Jun 18 13:50:53	205.188.228.65:7532 -> MY.NET.85.58:6970	UDP

Figure 38 - Snort Portscan Output Sample

For the purposes of larger picture analysis, the word “SCAN” was added to the “Scan Type” field when preparing the portscan logs for the database.

There were only a total of 33 out-of-spec packets in the five OOS files. It appeared that they were captured in Snort’s sniffer mode, as shown below:

Timestamp 04/08-14:15:51.056448	Source IP Address : Port 212.211.86.7:23	Traffic Flow ->	Dest IP Address : Port MY.NET.5.79:23
Protocol TCP	Time To Live TTL:20	Type of Service TOS:0x0	IP ID ID:39426
TCP Flags **SF****	TCP Sequence ID Seq: 0xC4BAAFF	TCP Ack Ack: 0x5AC2A143	TCP Window Size Win: 0x404

Figure 39 - Snort Out Of Spec Output Sample

I determined that the best approach when dealing with such a large amount of data would be to get all of the files into a similar format and import them into a database. In this fashion, it would be possible to quickly review the data in many different ways.

I began by using the Cygwin toolset and removing all portscan data from the alert files. Given that this information was present in the scan files, I wanted to avoid redundant data sets. This was accomplished by using the Cygwin toolset for Windows with the command:

```
grep –v “spp_portscan” <alert_file> > alert_file_noscan
```

From there, I set off to write a Perl script which would convert both the alerts and scan files to a similar delimited file format which could be imported into a database. The format I had determined to use was:

```
Month;Date;Time;Alert;Source_IP;Source_Port;Destination_IP;Destination_Port;
```

Figure 40 - Common Delimited Log Format

It was during the conversion process that I discovered corruption in the alerts file format. Approximately halfway through the alert data on the 18th of June and continuing through the 20th, I found a number of lines in which the alerts ran together. It appeared almost as though there were too many alerts coming in at some points and as multiple sensors tried to log data, some data was cut off or overwritten. I determined that there were two courses of action. I could try to manually fix every error, or throw out the erroneous data. I chose the latter, being that I did not wish to inject flawed data into my analysis.

The criteria for determining a valid alert line was as follows:

1. Line was greater than 50 characters
2. Line was less than 150 characters
3. Line possessed only one traffic direction symbol (->)
4. Line possessed only two Snort Alert symbols ([**])

5. The source and destination IP had four dots (x.y.z.a)

Using these criteria to parse the log files, 27,879 alert lines or .5% were determined to be invalid. This was determined by using the wc command as shown below:

$ cat alert*_noscan | wc –l 5357698	$cat parsed_alert* | wc –l 5329819
$ cat scan* | wc –l 1277500	$cat parsed_scan* | wc –l 1277500

Figure 41 - Parsed Line Count Comparison

Once converted, I concatenated the files together with the command:

```
cat parsed* > events_list.txt
```

Just to try and understand how many IP addresses I was dealing with, I wrote another quick Perl script to grab the source and destination IPs from each parsed alert line and output them each to their own line in a new file. I then used the unique sort command to produce a complete list of IP addresses in my events list. The sort command was run as follows:

```
cat all_src_dst_ips.txt | sort –u > final_hosts
```

Doing another line count on the final_hosts file, I learned that I was dealing with 100,385 unique IP addresses. The OOS alerts were considered somewhat apart from the larger data, as there were only 33 alerts and the timeframe was two months prior.

After several unsuccessful attempts to import the data into mySQL, I fell back on Microsoft Access and created a linked table from the events_list file. This was probably a less efficient choice, but it would allow me to adequately perform the queries required to analyze the data. Checking the number of records in the linked table, I confirmed that there were 6,607,319 records. This was the same as the number of lines in the parsed scan and alert files combined.

From here I decided the best course of action would be to try to understand the network behind the scene. I began to generate SQL queries to view the data in different formats. These queries included the number of alerts generated by source IP, destination IP, by source and destination IP pairs, by the number of machines connected to each IP, the number of machines connected from each IP, and the source and destination ports for each machine. This generated approximately 40 unique IP addresses of interest.

Moving to the broader scope, queries were generated to count the total number of alerts across the network, the number of IPs within each 24-bit subnet that showed up in the alerts files, and the top destination ports generating alerts. Pulling all of the alerts targeted for specific destination ports hosting common services (such as FTP, SMTP, SSH, telnet, etc) allowed the construction of a basic network profile. This allowed identification of “high value” targets running multiple common services and began to allow the drawing of insights into some of the host machines. As questionable traffic was found in the generation of the network profile, the IP addresses were noted for review under the *“Insight into Internal IPs”* section of the practical.

A search was then done for all IPs generating alerts containing the keywords: Possible Trojan Server activity, Red Worm, Nimda, Back Orifice, NetMetro, Backdoor, MS-SQL, and Virus. This pulled together a list of 153 internal IPs in need of investigation. Due to space constraints, only a small sampling of IPs were chosen to review.

The link graph and associated analysis were done last. Looking at it now, I would recommend to others analyzing large amounts of data to use the link graph as a starting point, instead of an ending point.

4 Appendix A

4.1 Automated Rotating of log files in a Windows Environment

Given the volume of information collected, rotating WinDump log files manually is a significant hassle. At the time of this writing, the current version of WinDump does not support automated log rotation. In response, I have developed a method to perform this task using Perl, the Window's "at" command, "pskill", a freeware command-line process killer for windows, and "pulist", a free command-line process status lister from the Windows resource center. The script assumes that it is named windump_rotate.pl. It assumes that both pskill and pulist are in the same directory as the Perl script and that the WinDump executable is in the system path. Logs will be rotated every hour.

The global section in this script should be configured to your individual environment prior to execution. To initiate the script, use the "at" command to set the first execution time. An example of this would be: "at 08:00 Perl c:\home\perl\windump_rotate.pl".

This script has only been tested on a Windows 2000 platform.

```
-----------------------
############################
# INCLUDES
############################
use Time::localtime;

############################
# GLOBALS
############################
$LOG_DIR = "c:\\home\\snort\\log";           # Path to write log files
$ROT_LOC = "c:\\home\\perl\\rotate_windump";      # Location of directory
containing the rotation script
$ROT_EXEC = "$ROT_LOC\\windump_rotate.pl";              # Location of
rotation script
$WD_OPTS = "-s 1500 -n";                                # Windump command
line options

############################
# SUBROUTINES
############################
sub zero_pad {
 $myNum = shift;

 if ($myNum < 10) {
         $myNum = "0$myNum";
 }

 return $myNum;
}

############################
# MAIN CODE EXECUTION
############################
$i = 0;
```

```
# Set time variables
$hour = localtime->hour;
$min = localtime->min;
$day = localtime->mday;
$month = localtime->mon;
$year = localtime->year;

# Format the date with Zeros to fill the log file name.
$year = sprintf("%02d", $year % 100);   # Only want the last two digits of
the year
$month += 1;
$month = zero_pad($month);
$day = zero_pad($day);
$hour = zero_pad($hour);
$min = zero_pad($min);

# Scheduled next log rotation
if ($hour == 23) {
        $nextHour = 0;
} else {
        $nextHour = $hour + 1;
}
$cmd = "at $nextHour:00 perl $ROT_EXEC 1>&2";
system $cmd;

# Find old windump process to kill and schedule it to be killed at current
minute +1
open (PROCESS, "$ROT_LOC\\pulist.exe |");
while (<PROCESS>) {
        @words = split(" ","$_");
        if ("$words[0]" eq "WinDump.exe") {
                $nextMin = zero_pad($min + 1);
                $cmd = "at $hour:$nextMin $ROT_LOC\\pskill.exe $words[1]
1>&2";
                system "$cmd";
        }
}

# Start new windump program
$newLogFile = "$LOG_DIR\\$year$month$day-$hour$min.cap";  # Timestamp for new
logfile
$wd_exec = "windump $WD_OPTS -w $newLogFile";
exec $wd_exec;

exit;
-----------------------
```

4.2 Parsing Unique Errors from a Snort Full Alert File

The following script is designed to read in a Snort full alerts file and parse out unique error messages in a quick text format. It is written in perl and uses the grep command which is native to *nix systems and can be implemented in Windows environments through the use of the Cygwin toolset. The $OUT variable should be set to the directory where the full alerts file can be located prior to execution. The program looks for full*.ids, which is the name I typically give my full alerts files.

The script should be executed as: “perl get_summary.pl” Brief usage instructions will be printed from there.

```
-----------------------
$DIR = shift;                                   # Specific Summary Directory
$OUT = "/cygdrive/c/home/snort/snort1_86/log";  # Main Logging Directory

if ($DIR) {
        chdir "$OUT/$DIR";

        $cmd = "grep \"\\[\\*\\*\\]\" full*.ids | grep -v \"spp_port\" | sort
-u > sum.txt";
        exec $cmd;
} else {
        print "\nget_sum.pl v1.0\n";
        print "Written By: Jason Tant\n\n";
        print "USAGE: perl get_sum.pl <alert_dir>\n\n";
        print "Where alert_dir is a directory under\n";
        print "$OUT\n";
        print "containing a full alert file from snort\n";
        print "with the title full*.ids\n\n";
}
-----------------------
```

5 Appendix B

5.1 University Log File Alerts

ALERT	TIMES LOGGED	ALERT	TIMES LOGGED
suspicious host traffic	4683696	SCAN FULLXMAS **UAPRSF	2
SCAN UDP	854791	SCAN INVALIDACK ***APRS*	2
SCAN SYN ******S*	419769	SCAN INVALIDACK **UA*RS*	2
UDP SRC and DST outside network	104095	SCAN INVALIDACK *2*A*RSF RESERVEDBITS	2
SMB Name Wildcard	85891	SCAN INVALIDACK *2*AP*SF RESERVEDBITS	2
ICMP Destination Unreachable (Communication Administratively Prohibited)	70771	SCAN INVALIDACK *2*APR*F RESERVEDBITS	2
MISC traceroute	41745	SCAN INVALIDACK *2UA**S* RESERVEDBITS	2
CS WEBSERVER - external web traffic	35043	SCAN INVALIDACK *2UA*RS* RESERVEDBITS	2
INFO Inbound GNUTella Connect request	30853	SCAN INVALIDACK 1*UA**S* RESERVEDBITS	2
MISC Large UDP Packet	29919	SCAN INVALIDACK 12UA*R*F RESERVEDBITS	2
ICMP Echo Request Sun Solaris	24112	SCAN NMAPID *2U*P*SF RESERVEDBITS	2
ICMP Destination Unreachable (Host Unreachable)	21720	SCAN NOACK ****P*S*	2
MISC source port 53 to <1024	20525	SCAN NOACK ****PRS*	2
spp_http_decode: IIS Unicode attack detected	18471	SCAN NOACK **U***S*	2
WEB-MISC prefix-get //	18210	SCAN NOACK **U***SF	2
Watchlist 000222 NET-NCFC	15677	SCAN NOACK **U**R**	2
WEB-MISC Attempt to execute cmd	15457	SCAN NOACK **U**RS*	2
AFS - Off-campus activity	14386	SCAN NOACK *2**PRS* RESERVEDBITS	2
Watchlist 000220 IL-ISDNNET-990517	12427	SCAN NOACK 1****R*F RESERVEDBITS	2
SNMP public access	10271	SCAN NOACK 1****RS* RESERVEDBITS	2
INFO MSN IM Chat data	9144	SCAN NOACK 1***P*SF RESERVEDBITS	2
connect to 515 from outside	9115	SCAN NOACK 1***PR** RESERVEDBITS	2
ICMP Echo Request L3retriever	8885	SCAN NOACK 1***PR*F	2

Ping		RESERVEDBITS	
ICMP Echo Request BSDtype	6505	SCAN NOACK 12U**R** RESERVEDBITS	2
ICMP Echo Request Windows	5351	SCAN SPAU **UAP*S*	2
ICMP Destination Unreachable (Network Unreachable)	5190	SCAN SYNFIN *2****SF RESERVEDBITS	2
INFO FTP anonymous FTP	3304	SCAN SYNFIN 1*****SF RESERVEDBITS	2
ICMP Echo Request Nmap or HPING2	3138	SCAN UNKNOWN *2*A**** RESERVEDBITS	2
MISC Large ICMP Packet	3085	SCAN UNKNOWN *2*A***F RESERVEDBITS	2
SCAN Proxy attempt	2940	SCAN UNKNOWN *2*A*R** RESERVEDBITS	2
Queso fingerprint	2455	SCAN UNKNOWN *2UAP**F RESERVEDBITS	2
SCAN SYN 12****S* RESERVEDBITS	2324	SCAN UNKNOWN 1**A**S* RESERVEDBITS	2
ICMP Source Quench	2263	SCAN UNKNOWN 1**AP*** RESERVEDBITS	2
SMB C access	1959	SCAN UNKNOWN 1*UAP**F RESERVEDBITS	2
CS WEBSERVER - external ftp traffic	1055	SCAN UNKNOWN 12UA**** RESERVEDBITS	2
ICMP Echo Request Broadscan Smurf Scanner	916	SCAN VECNA 1***P*** RESERVEDBITS	2
INFO - Possible Squid Scan	876	SCAN VECNA 12U*P*** RESERVEDBITS	2
High port 65535 udp - possible Red Worm - traffic	831	SCAN XMAS	2
FTP DoS ftpd globbing	804	SCAN XMAS *2U*P**F RESERVEDBITS	2
Port 55850 tcp - Possible myserver activity - ref. 010313-1	792	SCAN XMAS 12U*P**F RESERVEDBITS	2
Attempted Sun RPC high port access	731	SYN-FIN scan!	2
IDS552/web-iis_IIS ISAPI Overflow ida nosize	669	TFTP - Internal UDP connection to external tftp server	2
Incomplete Packet Fragments Discarded	536	Virus - Possible scr Worm	2
High port 65535 tcp - possible Red Worm - traffic	475	WEB-CGI w3-msql access	2
EXPLOIT x86 NOOP	442	WEB-IIS .cnf access	2
WEB-IIS view source via	405	x86 NOOP - unicode BUFFER	2

translate header		OVERFLOW ATTACK	
ICMP Destination Unreachable (Fragmentation Needed and DF bit was set)	395	DDOS shaft client to handler	1
ICMP Router Selection	354	External FTP to HelpDesk MY.NET.70.50	1
ICMP traceroute	350	ICMP Redirect (Network)	1
WEB-MISC http directory traversal	336	IRC evil - running XDCC	1
Null scan!	234	SCAN - wayboard request - allows reading of arbitrary files as http service	1
NIMDA - Attempt to execute cmd from campus host	214	SCAN FIN 1******F RESERVEDBITS	1
SUNRPC highport access!	213	SCAN FIN 12*****F RESERVEDBITS	1
INFO Outbound GNUTella Connect request	186	SCAN FULLXMAS 1*UAPRSF RESERVEDBITS	1
WEB-MISC count.cgi access	159	SCAN FULLXMAS 12UAPRSF RESERVEDBITS	1
SCAN NULL ********	146	SCAN INVALIDACK **UA**SF	1
WEB-CGI formmail access	133	SCAN INVALIDACK **UA*R**	1
NMAP TCP ping!	122	SCAN INVALIDACK **UA*R*F	1
WEB-MISC Lotus Domino directory traversal	120	SCAN INVALIDACK **UA*RSF	1
SCAN VECNA ****P***	116	SCAN INVALIDACK **UAP*SF	1
WEB-CGI rsh access	114	SCAN INVALIDACK **UAPRS*	1
WEB-IIS SAM Attempt	111	SCAN INVALIDACK *2*A**SF RESERVEDBITS	1
ICMP Fragment Reassembly Time Exceeded	109	SCAN INVALIDACK *2*A*R*F RESERVEDBITS	1
WEB-IIS _vti_inf access	107	SCAN INVALIDACK *2*A*RS* RESERVEDBITS	1
WEB-FRONTPAGE _vti_rpc access	102	SCAN INVALIDACK *2*AP*S* RESERVEDBITS	1
WEB-MISC compaq nsight directory traversal	102	SCAN INVALIDACK *2*APRS* RESERVEDBITS	1
INFO Outbound GNUTella Connect accept	87	SCAN INVALIDACK *2UA**SF RESERVEDBITS	1
Possible trojan server activity	79	SCAN INVALIDACK *2UA*R** RESERVEDBITS	1
SMTP chameleon overflow	76	SCAN INVALIDACK *2UA*R*F RESERVEDBITS	1
ICMP Address Mask Request	70	SCAN INVALIDACK *2UAP*SF RESERVEDBITS	1

INFO Napster Client Data	62	SCAN INVALIDACK *2UAPR** RESERVEDBITS	1
RPC tcp traffic contains bin_sh	60	SCAN INVALIDACK *2UAPR*F RESERVEDBITS	1
MS-SQL xp_cmdshell - program execution	52	SCAN INVALIDACK *2UAPRS* RESERVEDBITS	1
X11 outgoing	52	SCAN INVALIDACK 1**A**SF RESERVEDBITS	1
INFO Possible IRC Access	48	SCAN INVALIDACK 1**A*R*F RESERVEDBITS	1
WEB-CGI redirect access	45	SCAN INVALIDACK 1**A*RS* RESERVEDBITS	1
SCAN INVALIDACK ***A*R*F	39	SCAN INVALIDACK 1**AP*S* RESERVEDBITS	1
beetle.ucs	37	SCAN INVALIDACK 1**AP*SF RESERVEDBITS	1
WEB-CGI finger access	36	SCAN INVALIDACK 1**APR*F RESERVEDBITS	1
EXPLOIT x86 setuid 0	34	SCAN INVALIDACK 1*UA**SF RESERVEDBITS	1
SCAN Synscan Portscan ID 19104	34	SCAN INVALIDACK 1*UA*RS* RESERVEDBITS	1
NETBIOS NT NULL session	30	SCAN INVALIDACK 1*UA*RSF RESERVEDBITS	1
WEB-MISC whisker head	30	SCAN INVALIDACK 1*UAPR** RESERVEDBITS	1
spp_http_decode: CGI Null Byte attack detected	29	SCAN INVALIDACK 12*A**SF RESERVEDBITS	1
EXPLOIT x86 setgid 0	25	SCAN INVALIDACK 12*A*R*F RESERVEDBITS	1
ICMP Destination Unreachable (Protocol Unreachable)	24	SCAN INVALIDACK 12*A*RSF RESERVEDBITS	1
WEB-IIS Unicode2.pl script (File permission canonicalization	24	SCAN INVALIDACK 12*AP*S* RESERVEDBITS	1
BACKDOOR NetMetro Incoming Traffic	22	SCAN INVALIDACK 12*AP*SF RESERVEDBITS	1
WEB-MISC nc.exe attempt	22	SCAN INVALIDACK 12*APRS* RESERVEDBITS	1
WEB-FRONTPAGE fpcount.exe access	17	SCAN INVALIDACK 12UA*R** RESERVEDBITS	1
SCAN INVALIDACK ***APR*F	16	SCAN INVALIDACK 12UA*RS* RESERVEDBITS	1
Notify Brian B. 3.54 tcp	15	SCAN INVALIDACK 12UA*RSF RESERVEDBITS	1

SCAN NOACK *2U**R*F RESERVEDBITS	14	SCAN INVALIDACK 12UAP*SF RESERVEDBITS	1
Notify Brian B. 3.56 tcp	13	SCAN NOACK *****R*F	1
WEB-MISC ICQ Webfront HTTP DOS	13	SCAN NOACK ****P*SF	1
ICMP Echo Request CyberKit 2.2 Windows	12	SCAN NOACK ****PR**	1
WEB-CGI csh access	12	SCAN NOACK ****PR*F	1
MISC PCAnywhere Startup	11	SCAN NOACK ****PRSF	1
Port 55850 udp - Possible myserver activity - ref. 010313-1	11	SCAN NOACK **U**RSF	1
Tiny Fragments - Possible Hostile Activity	11	SCAN NOACK **U*PR**	1
ICMP Parameter Problem (Unspecified Error)	8	SCAN NOACK **U*PRSF	1
SCAN FIN *2*****F RESERVEDBITS	8	SCAN NOACK *2**P*S* RESERVEDBITS	1
WEB-IIS Unauthorized IP Access Attempt	8	SCAN NOACK *2**P*SF RESERVEDBITS	1
WEB-MISC 403 Forbidden	8	SCAN NOACK *2**PRSF RESERVEDBITS	1
ICMP redirect (Host)	7	SCAN NOACK *2U*PR** RESERVEDBITS	1
WEB-MISC handler access	7	SCAN NOACK *2U*PR*F RESERVEDBITS	1
ICMP Destination Unreachable (Destination Host Unknown)	6	SCAN NOACK *2U*PRSF RESERVEDBITS	1
SCAN UNKNOWN 1**A*R** RESERVEDBITS	6	SCAN NOACK 1****RSF RESERVEDBITS	1
WEB-CGI survey.cgi access	6	SCAN NOACK 1***P*S* RESERVEDBITS	1
SCAN NOACK *2U**R** RESERVEDBITS	5	SCAN NOACK 1***PRS* RESERVEDBITS	1
SCAN NOACK 1*U**RS* RESERVEDBITS	5	SCAN NOACK 1*U***S* RESERVEDBITS	1
SCAN VECNA *2U*P*** RESERVEDBITS	5	SCAN NOACK 1*U**R** RESERVEDBITS	1
WEB-FRONTPAGE shtml.exe	5	SCAN NOACK 1*U**RSF RESERVEDBITS	1
CS WEBSERVER - external ssh traffic	4	SCAN NOACK 1*U*P*S* RESERVEDBITS	1
External RPC call	4	SCAN NOACK 1*U*PR** RESERVEDBITS	1
IDS475/web-iis_web-webdav-	4	SCAN NOACK 1*U*PR*F	1

propfind		RESERVEDBITS	
RFB - Possible WinVNC - 010708-1	4	SCAN NOACK 12***RS* RESERVEDBITS	1
SCAN INVALIDACK ***AP*S*	4	SCAN NOACK 12**P*S* RESERVEDBITS	1
SCAN INVALIDACK 12UAPR*F RESERVEDBITS	4	SCAN NOACK 12**PR*F RESERVEDBITS	1
SCAN NMAPID 1*U*P*SF RESERVEDBITS	4	SCAN NOACK 12U***S* RESERVEDBITS	1
SCAN NOACK *2U**RS* RESERVEDBITS	4	SCAN NOACK 12U*P*S* RESERVEDBITS	1
SCAN SPAU *2UAP*S* RESERVEDBITS	4	SCAN NOACK 12U*PR** RESERVEDBITS	1
SCAN UNKNOWN 12*A**S* RESERVEDBITS	4	SCAN NOACK 12U*PR*F RESERVEDBITS	1
WEB-CGI glimpse access	4	SCAN SPAU 12UAP*S* RESERVEDBITS	1
WEB-CGI ksh access	4	SCAN SYN ******S*Jun 19 04:15:44 202.224.237.106:4235 -> MY.NET.254.193:80 SYN ******S*	1
WEB-CGI tsch access	4	SCAN SYN *2****S* RESERVEDBITS	1
FTP .forward	3	SCAN SYNFIN ******SF	1
SCAN FIN	3	SCAN SYNFIN 12****SF RESERVEDBITS	1
SCAN INVALIDACK ***A*RS*	3	SCAN UNKNOWN *2***R** RESERVEDBITS	1
SCAN INVALIDACK *2*APRSF RESERVEDBITS	3	SCAN UNKNOWN *2*AP*** RESERVEDBITS	1
SCAN INVALIDACK 12UA**S* RESERVEDBITS	3	SCAN UNKNOWN *2*APR** RESERVEDBITS	1
SCAN NOACK *****RS*	3	SCAN UNKNOWN *2UAP*** RESERVEDBITS	1
SCAN NOACK **U*P*S*	3	SCAN UNKNOWN 1**A***F RESERVEDBITS	1
SCAN NOACK *2***RS* RESERVEDBITS	3	SCAN UNKNOWN 1**AP**F RESERVEDBITS	1
SCAN NOACK *2**PR** RESERVEDBITS	3	SCAN UNKNOWN 1*UA**** RESERVEDBITS	1
SCAN NOACK *2U**RSF RESERVEDBITS	3	SCAN UNKNOWN 12*A*R** RESERVEDBITS	1
SCAN NOACK *2U*PRS* RESERVEDBITS	3	SCAN UNKNOWN 12*AP**F RESERVEDBITS	1
SCAN NOACK 1*U**R*F	3	SCAN UNKNOWN 12UAP***	1

RESERVEDBITS		RESERVEDBITS	
SCAN NOACK 12***R*F RESERVEDBITS	3	SCAN VECNA **U****F	1
SCAN NOACK 12U**RSF RESERVEDBITS	3	SCAN VECNA **U*P***	1
SCAN UNKNOWN *2*AP**F RESERVEDBITS	3	SCAN VECNA *2U***** RESERVEDBITS	1
SCAN UNKNOWN *2UA**** RESERVEDBITS	3	SCAN VECNA *2U****F RESERVEDBITS	1
SCAN UNKNOWN *2UA***F RESERVEDBITS	3	SCAN VECNA 1***P**F RESERVEDBITS	1
SCAN UNKNOWN 12***R** RESERVEDBITS	3	SCAN VECNA 1*U***** RESERVEDBITS	1
SCAN VECNA **U*****	3	SCAN VECNA 12U***** RESERVEDBITS	1
SCAN VECNA *2**P**F RESERVEDBITS	3	SCAN XMAS 1*U*P**F RESERVEDBITS	1
SCAN XMAS **U*P**F	3	TFTP - External UDP connection to internal tftp server	1
WEB-CGI scriptalias access	3	Virus - Possible pif Worm	1
WEB-IIS scripts-browse	3	WEB-CGI archie access	1
Back Orifice	2	WEB-MISC .htaccess access	1
EXPLOIT NTPDX buffer overflow	2	WEB-MISC ~root	1
FTP passwd attempt	2	WEB-MISC ftp attempt	1
MISC Source Port 20 to <1024	2	WEB-MISC L3retriever HTTP Probe	1
SCAN FIN *******F	2	WEB-MISC whisker splice attack	1