GIAC Certified Intrusion Analyst, GCIA

Mike Shannon
Los Angeles, October 2003

Submitted: February 19, 2004
GCIA Practical v3.4 (revised September 24, 2003)

Table of Contents

List of Figures:

List of Tables:

Section 1: Describe the State of Intrusion Detection

Intrusion Detection for PPPoE using a Linksys router and LinkLogger

Abstract:

This paper discusses the benefits of utilizing a Linksys router as an Intrusion Detection System with the adjunct use of LinkLogger for data analysis. The paper additionally compares LinkLogger to syslog and other standard IDS systems.

Introduction:

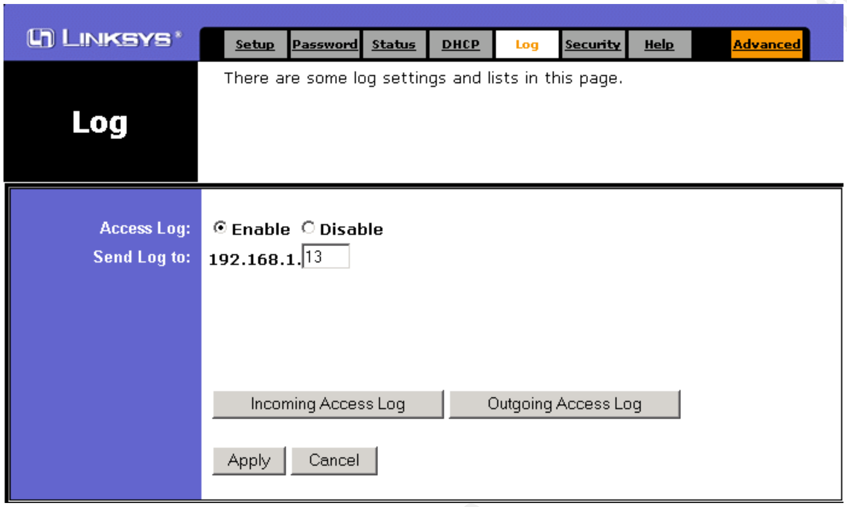
DSL and Cable connectivity being almost ubiquitous, as having a continuous connection to the internet is, for most people, a relatively common occurrence. In the majority of instances, security for these round-the-clock connections is non-existent for the average Small Office Home Office (SOHO) user.[1] Many individual SOHOs do install antivirus software, while others add a personal firewall. These are prudent and effective security practices, as far as they go. In some cases, individuals take their security more seriously, and acquire a router to manage their continuous connection, which will, in effect, sit between the internet and their SOHO Local Area Network (LAN). Additionally, it will use some form of Network Address Translation (NAT) to further disguise its presence. Combining these three strategies can lead to a fairly secure SOHO environment which will deter all but the most determined attacker. Having said this, however, while using a combination of router, personal firewall, and antivirus may supply reasonable security, it cannot provide information regarding the source of malicious entities probing the connection seeking entry into the SOHO network. For security concerned professionals, this is an unacceptable circumstance.

One of the more common of today's ADSL connectivity protocols is PPP over Ethernet (PPPoE) with a DHCP provided IP address. This type of ADSL requires some form of authentication which the router typically handles for the end user of PPPoE. The router performs its intended task with efficiency, and the vast majority of SOHOs are content in the knowledge that their network is secure. However, this is insufficient for the discriminating number of SOHOs with a more inquiring bent, thirst for knowledge, and factual information. In order to determine the incoming activity prior to reaching its destination and handling within the router, the use of an IDS placed outside the router in a location that has access to the external internet is an effective solution. Thus, sniffing the connection outside of the router is possible by placing a hub in front of it, and, then, placing an IDS on the external connection. Most SOHO users do not have the facility to place hardened IDS on their external network. Frankly, to enable the ability to see who is "knocking" on your door, in a manner of speaking, it would be more convenient to have a sniffer or IDS internal to the PPPoE device. Many of the routers currently available do have the capability of logging the incoming traffic to the router and, in some cases, allow the logs to be forwarded to a syslog, SNMP or third party software for log analysis.

The purpose of this discussion is to describe the use of a Linksys BEFW11S4 wireless router in combination with Binary Visions' LinkLogger product for intrusion detection on the SOHO ADSL connection and to discuss the pros, cons, and other options available to a SOHO implementation.

Linksys Configuration:

The Linksys BEFW11S4 utilizes a browser interface to setup its configuration.[2] On the version of the firmware used for purposes of this discussion, the Log setting was found on the initial setup screen, following the user login to the router. To find it, simply select the Log Tab (see below). Following this step, select: Access Log: enable and Send Log to the IP of your logging device. In this case, 192.168.1.13 (which is an IP not in the DHCP scope) was chosen, and the logging device was then prepared to utilize a static IP address of 192.168.1.13.



There are some log settings and lists in this page.

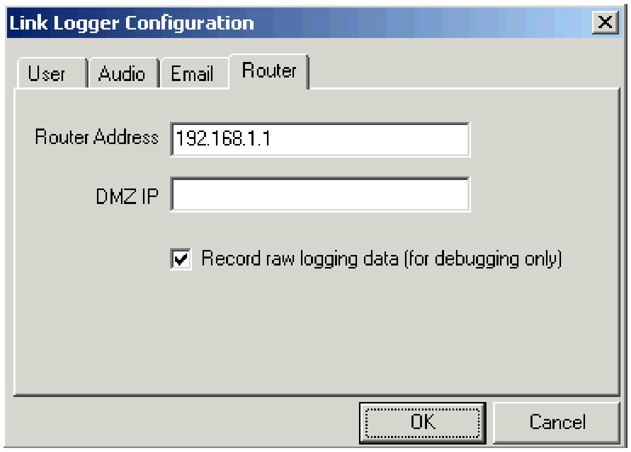
Access Log: ● Enable ○ Disable

Send Log to: 192.168.1.13

LinkLogger Configuration:

LinkLogger's intrusion detection program can be downloaded from http://www.linklogger.com. It comes in different formats, and it is important to download the correct format for the router in use. The version used in this example was 1.6.2.314 for Linksys routers other than model BEFSX41. Linklogger comes as a single executable. There is a thirty day demonstration option for the software, which allows some leeway in determining whether the program is suitable for the user's needs. Following the thirty days, the product needs to be purchased for continuing its use. Once the software is installed, click on LinkLogger, and it will run in the system tray as well as popping up a LinkLogger screen.

To configure LinkLogger simply open the program, and select Edit/Setup from the menu bar. From that point, the various settings and the router's IP address can be configured. In the example, and, in most cases for Linksys routers, 192.168.1.1 is the router's IP address.

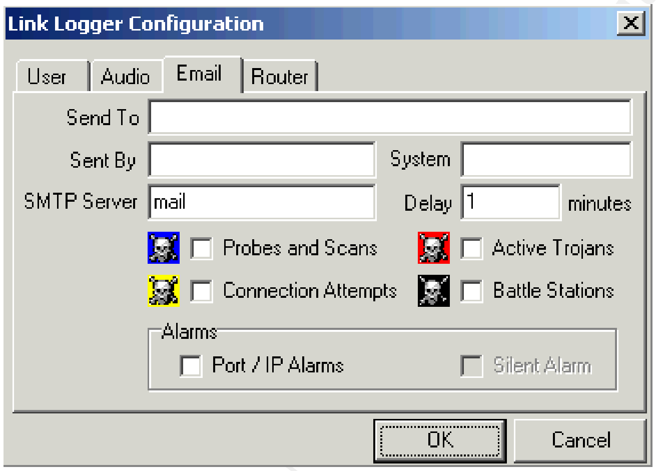


Router Address: 192.168.1.1

DMZ IP:

☑ Record raw logging data (for debugging only)

As with most IDS, the alerts are passive in nature. It takes a trained engineer to examine the logs and determine whether the alerts are of real concern or are false positives. LinkLogger has a notification facility which can be used to create a type of Alerting system. In order to use this, the Email option needs to be enabled so that the user is notified when particular alerts require attention, for example, the 'Battle Stations' alert. It is important to know that the blue and yellow alerts are inbound connections, and that the red and black are outbound connection attempts, see below. The delay feature prevents the user from mail bombing their own mail server with alerts.



Send To:

Sent By:

System:

SMTP Server: mail

Delay: 1 minutes

□ Probes and Scans

□ Active Trojans

□ Connection Attempts

□ Battle Stations

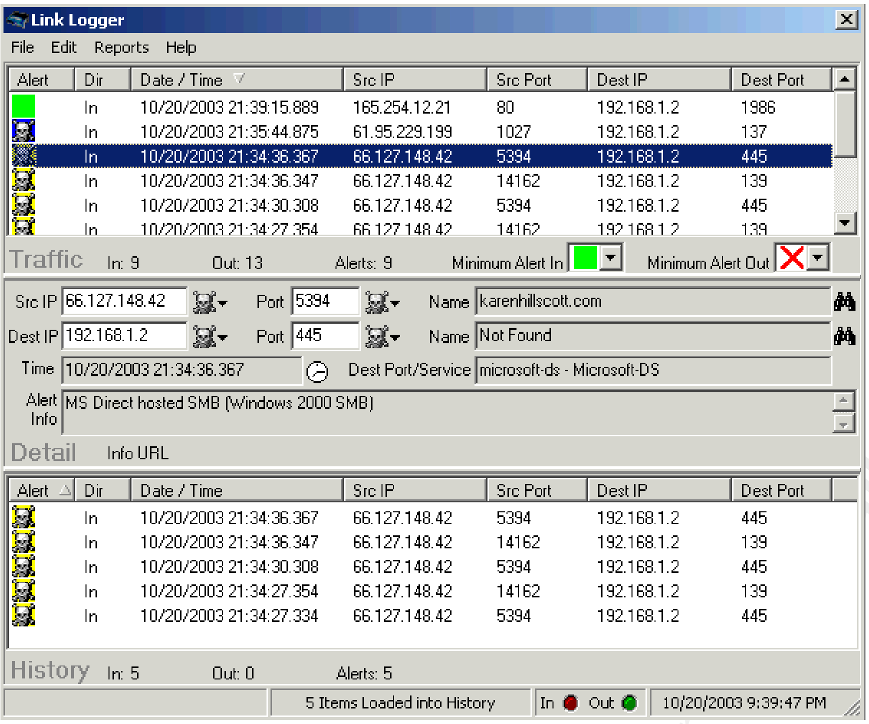
Alarms

□ Port / IP Alarms

□ Silent Alarm

The Active Trojan and Battle Stations alerts are used to indicate a potential response to a Trojan inside your network.[3] This is extremely useful information if, indeed, a Trojan has infiltrated the network.

Once the LinkLogger program is configured, and the router points logs to the logging device, all incoming external activity will be shown in the log window as seen below. Logging capabilities can be tested by utilizing a free vulnerability scanning web site; their service scans the network you request for insecurities. [4 or 5]



Alert	Dir	Date / Time	Src IP	Src Port	Dest IP	Dest Port
🟩	In	10/20/2003 21:39:15.889	165.254.12.21	80	192.168.1.2	1986
	In	10/20/2003 21:35:44.875	61.95.229.199	1027	192.168.1.2	137
	In	10/20/2003 21:34:36.367	66.127.148.42	5394	192.168.1.2	445
	In	10/20/2003 21:34:36.347	66.127.148.42	14162	192.168.1.2	139
	In	10/20/2003 21:34:30.308	66.127.148.42	5394	192.168.1.2	445
	In	10/20/2003 21:34:27.354	66.127.148.42	14162	192.168.1.2	139

Traffic In: 9 Out: 13 Alerts: 9 Minimum Alert In 🟩 Minimum Alert Out ✗

Src IP 66.127.148.42 Port 5394 Name karenhillscott.com

Dest IP 192.168.1.2 Port 445 Name Not Found

Time 10/20/2003 21:34:36.367 Dest Port/Service microsoft-ds - Microsoft-DS

Alert Info MS Direct hosted SMB (Windows 2000 SMB)

Detail Info URL

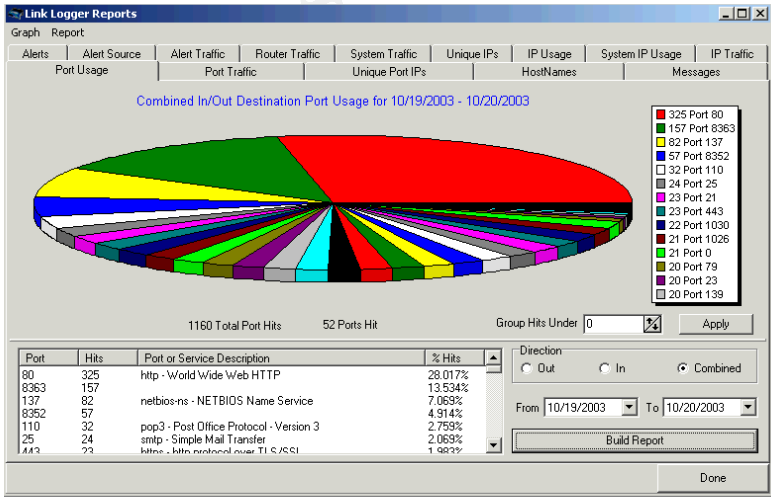
Alert	Dir	Date / Time	Src IP	Src Port	Dest IP	Dest Port
	In	10/20/2003 21:34:36.367	66.127.148.42	5394	192.168.1.2	445
	In	10/20/2003 21:34:36.347	66.127.148.42	14162	192.168.1.2	139
	In	10/20/2003 21:34:30.308	66.127.148.42	5394	192.168.1.2	445
	In	10/20/2003 21:34:27.354	66.127.148.42	14162	192.168.1.2	139
	In	10/20/2003 21:34:27.334	66.127.148.42	5394	192.168.1.2	445

History In: 5 Out: 0 Alerts: 5

5 Items Loaded into History In ● Out ● 10/20/2003 9:39:47 PM

It is also possible to filter the types of alerts the user is interested in seeing. If there is no interest in tracking the details relating to what and where the internal users are going, then the Minimum Alert Out can be set to None. Conversely, the Minimum Alert In can be set to conform to the user's individual requirements. Alerts can be filtered in priority, from benign to harmful. When viewing the logs, it is possible to select a particular item for detailed information regarding the attacking/scanning host, and which port was used as the vector of attack (both from and to). Again, it is possible to select a port or IP address to receive summary details on either the port or the IP, and all incidents of that port or IP being used as a vector of attack are viewable.

In addition, there is a reporting feature built into the product; it allows for the selection of various reports which sort and analyze the SOHOs network traffic.



Group Hits Under 0

Port	Hits	Port or Service Description	% Hits
80	325	http - World Wide Web HTTP	28.017%
8363	157		13.534%
137	82	netbios-ns - NETBIOS Name Service	7.069%
8352	57		4.914%
110	32	pop3 - Post Office Protocol - Version 3	2.759%
25	24	smtp - Simple Mail Transfer	2.069%
443	23	https - http protocol over TLS/SSL	1.983%

Direction ○ Out ○ In ● Combined

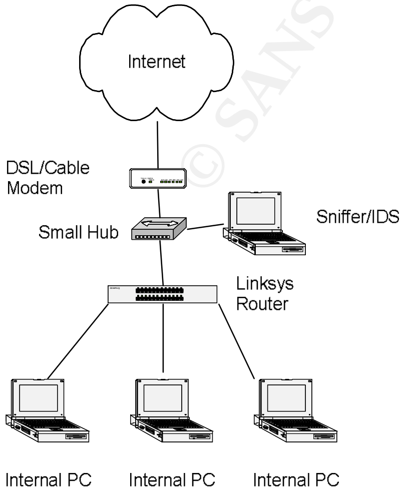
From 10/19/2003 To 10/20/2003

One of the better features of the product is the ability for the SOHO to send logs to DShield.org. DShield contributes data to the collective knowledge relating to intrusion detection on the internet. It's a superb grass roots effort, if you will, to better police and secure the internet from within. To participate in this, there is a second program, called Dshieldup, which requires downloading. As the Dshieldup configuration has been documented, thoroughly and separately, it does not warrant a repetitive short-shrift account of the product due to the constraints of space and focus.[6]

Conclusion:

The combination of router, personal firewall, antivirus and IDS provides the discerning SOHO with a secure setup, and one which generates comprehensive information regarding potential intrusions, their sources, and avenues of attempted infiltration. On a personal level, I have utilized this setup to good effect in investigating issues occuring with a SOHO setup. In particular, the SOHO setup was struggling with DSL connectivity issues during the MSBlaster worm outbreak, and I was having difficulty in troubleshooting the setup's connection. At that time, there was no IDS in the SOHO, so I used the logging facility in the Linksys router to the LinkLogger program to track down a PC that had become infected with MSBlaster. It was fortunate that the logs were available to review, because the logs allowed identification of the PC by search and review of the pertinent logs. The logs indicated that the malicious worm was attempting to connect to every PC on the internet over TCP 135. Without the logging facility, an internal sniffer would have been necessary to find the anomalous traffic.

Those SOHOs having an ADSL connection with PPPoE have a limited number of venues with which to view external traffic on their connection. For complete packet analysis and the best way to view traffic, the ideal design includes an IDS/Sniffer placed outside of the router, as is illustrated in the diagram below. This type of setup is often prohibitively expensive for the majority of SOHO users, which is the reason that the router becomes a viable option. While the Linksys router / LinkLogger combination should not be used for an enterprise level IDS, it remains sufficient for the SOHO user looking to see who's knocking on their internet door.



Additionally, there are other options to consider for remote logging. There is always a form of syslog or SNMP management station. If Linksys is the router in use, it is important to know that Linksys does not have a function to output to UDP 514 (standard syslog port), therefore, if a syslog system is chosen, it will have to listen on UDP 162 (SNMP). The issue that limits a syslog solution is that there is no filtering, inbound or outbound, when looking at the logs. In the syslog solution, all the logged information is the inbound and outbound traffic from the router. Another possibility to use for remote logging is Wallwatcher. Wallwatcher is compared to LinkLogger comprehensively and in detail, elsewhere, and, in encapsulated form, it is another type of logging system, perhaps not as comprehensive as LinkLogger.[7] An additional possibility for remote logging is the use of a Linux system for logging the messages. There are several configurations/applications available that automate logging in Linux to greater and lesser degrees.[8] Finally, the use of Kiwi's Syslog Daemon is a possible choice.[9] However, in my opinion, the logs in the Kiwi solution logs are difficult to interpret and do not sort easily; it's necessary to utilize some form of Log parser or anomaly detection tool to sort out the alerts from everyday traffic.

The use of the LinkLogger software as a form of IDS is a good choice for the interested above average SOHO, however, there are far more comprehensive and powerful options available for the professional to choose from. LinkLogger may fall short for the more professional Intrusion Analyst because it does not capture the entire packet for true packet analysis. Further, there is no binary dump of the packet which can be read into Tcpdump, WinDump, Snort or Ethereal. LinkLogger does, however; provide information about the traffic that is attempting to enter your network from the outside, which is usually masked by the router, providing the discriminating SOHO with fairly comprehensible information and additional surety of protection.

References:

1. CERT, Home Computer Security. 2002. http://www.cert.org/homeusers/HomeComputerSecurity/
2. Linksys Support, Linksys BEFW11S4-AT Manual, http://www.linksys.com/support/support.asp?spid=95
3. LinkLogger Program Help
4. Gibson Research Corporation, http://grc.com/
5. PC Flank, http://www.pcflank.com/
6. DShield.org Distributed Intrusion Detection System, http://www.dshield.org/clients/linksys_kiwi_setup.php
7. Sydney Jensen, GSEC Practical. 2002. http://www.sans.org/rr/papers/30/364.pdf
8. Michael J. Wohlgemuth, Linksys Monitor. 2002. http://woogie.net/linksysmon/
9. Kiwi Syslog. http://www.kiwisyslog.com/

Section 2: Three Network Detects.

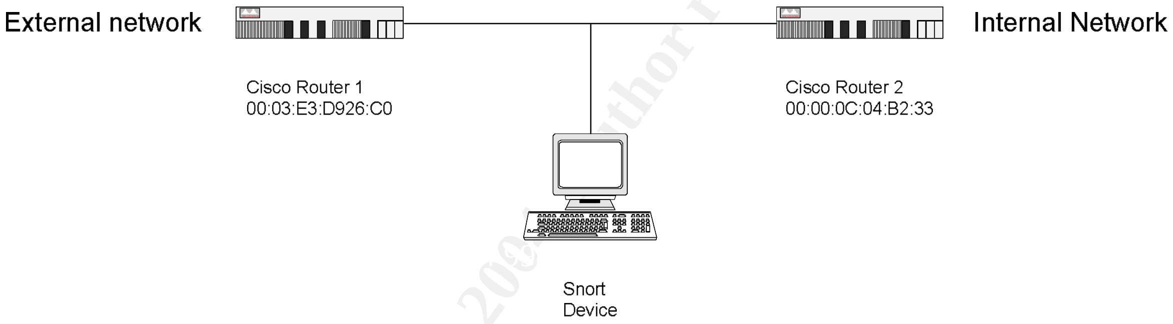
Detect #1: Backdoor Q

1. *Source of the trace:* In this particular case, the trace was downloaded from http://www.incidents.org/logs/raw/index.html. The specific file being utilized is 2002.10.15, and, to quote from the website, “The log files are the result of a Snort instance running in binary logging mode”.

   MAC address analysis indicates the following two MAC addresses: 00:03:E3:D926:C0 and 00:00:0C:04:B2:33. Each of these was extracted by using the following command:

   ```
   tcpdump -nn -e -r 2002.10.15 | cut -d ' ' -f 2,3 | sort –u
   ```

   Additionally, both of the MAC addresses are registered to Cisco. Having no direct knowledge of the exact network layout, it can, nevertheless, be surmised that the actual network layout would appear to be something along the lines of the following:



2. *Detect was generated by*: The alerts were generated by Snort Version 2.0.4 (Build 96) running on a Linux Red Hat 8 system using a fresh rule file named Snortrules-stable dated November 19, 2003. I have modified the default snort.conf by turning on all the rules, specifically by uncommenting them in the default snort.conf file.

   The command used to detect the alert was:
   snort –r 2002.10.15 –c ./snort.conf –l 20021015 –k none –A full –edNUX

   The following are the Snort options used above:
   -r which file to read.
   -c which configuration file to use
   -l where to log the output
   -k none ignore Checksums
   -A full write the alert file with full decoded header as well as the alert message
   -e displays the second layer header info
   -d dumps the application layer
   -N turn off logging
   -U uses UTC for timestamps

-X dump the raw packet date starting at the link layer

The alert was triggered by the snort rule listed hereafter:

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

The Snort rule seeks to trigger an alert on any TCP activity issuing from the source IP address of 255.255.255.0/24 from any port going to the Home Network (in this case Any was used as the HOME_NET). The alert will contain the message 'BACKDOOR Q access'. Additionally, the A+ statement indicates that as long as the Ack bit is set and any other flag is set the rule conditions are met. Finally, the last segment of the rule looks for any data greater than 1 byte.

What follows is a sample of the alerts generated by the Snort rule; of which, only three are included for brevity:

```
[**] [1:184:3] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
11/15-00:34:10.596507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
255.255.255.255:31337 -> 170.129.94.129:515 TCP TTL:15 TOS:0x0 ID:0 IpLen:20
DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20

[**] [1:184:3] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
11/15-00:38:10.756507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
255.255.255.255:31337 -> 170.129.181.145:515 TCP TTL:15 TOS:0x0 ID:0
IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]

[**] [1:184:3] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
11/15-01:56:56.236507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
255.255.255.255:31337 -> 170.129.72.205:515 TCP TTL:15 TOS:0x0 ID:0 IpLen:20
DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
```

Additionally, the following is a description of the fields in the Snort Alert:
[Classification: Misc activity]: Miscellaneous Activity
[Priority: 3]: Considered a Low Priority classification
11/15-00:34:10.596507: Date and time of the alert
0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33: Source and destination MAC address
type:0x800: Encapsulated Protocol is IP
len:0x3C: Length of the frame is 60 bytes

255.255.255.255:31337 -> 170.129.94.129:515: Source IP of 255.255.255.255 with a source port of 31337 and destination IP address of 170.129.94.129 and destination port of 515
TCP: TCP packet
TTL:15: Time To Live of 15
TOS:0x0: Type of Services bits are 0
ID:0: IP ID is 0
IpLen:20: IP Packet header Length is 20 bytes
DgmLen:43: Total Datagram length is 43 bytes
***A*R**: Ack and Reset bit are set
Seq: 0x0 Ack: 0x0 Win: 0x0: Sequence Number, Acknowledgement ID and TCP Window size of 0
TcpLen: 20: TCP Header Length is 20 bytes
[Xref => http://www.whitehats.com/info/IDS203]: This is reference to Whitehats.com.

In this instance, the alert was triggered as a result of the source IP of 255.255.255.255, Ack/Rst bits set and Datagram length > 1 byte.

For further packet analysis, a tcpdump of the binary file for a packet that caused the alert is listed hereafter:

```
[root@localhost raw-logs]# tcpdump -r 2002.10.15 -Xnnevc5 'host 255.255.255.255'
16:34:10.596507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 255.255.255.255.31337 >
170.129.94.129.515: R [tcp sum ok] 0:3(3) ack 0 win 0 [RST cko] (ttl 15, id 0, len
43)
0x0000   4500 002b 0000 0000 0f06 a2cb ffff ffff        E..+............
0x0010   aa81 5e81 7a69 0203 0000 0000 0000 0000        ..^.zi..........
0x0020   5014 0000 57f3 0000 636b 6f00 0000             P...W...cko...
16:38:10.756507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 255.255.255.255.31337 >
170.129.181.145.515: R [tcp sum ok] 0:3(3) ack 0 win 0 [RST cko] (ttl 15, id 0, len
43)
0x0000   4500 002b 0000 0000 0f06 4bbb ffff ffff        E..+......K.....
0x0010   aa81 b591 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 00e3 0000 636b 6f00 0000             P.......cko...
17:56:56.236507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 255.255.255.255.31337 >
170.129.72.205.515: R [tcp sum ok] 0:3(3) ack 0 win 0 [RST cko] (ttl 15, id 0, len
43)
0x0000   4500 002b 0000 0000 0f06 b87f ffff ffff        E..+............
0x0010   aa81 48cd 7a69 0203 0000 0000 0000 0000        ..H.zi..........
0x0020   5014 0000 6da7 0000 636b 6f00 0h000             P...m...cko...
18:10:50.476507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 255.255.255.255.31337 >
170.129.156.91.515: R [tcp sum ok] 0:3(3) ack 0 win 0 [RST cko] (ttl 15, id 0, len
43)
0x0000   4500 002b 0000 0000 0f06 64f1 ffff ffff        E..+......d.....
0x0010   aa81 9c5b 7a69 0203 0000 0000 0000 0000        ...[zi..........
0x0020   5014 0000 1a19 0000 636b 6f00 0000             P.......cko...
18:29:40.696507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 255.255.255.255.31337 >
170.129.161.211.515: R [tcp sum ok] 0:3(3) ack 0 win 0 [RST cko] (ttl 15, id 0, len
43)
0x0000   4500 002b 0000 0000 0f06 5f79 ffff ffff        E..+......_y....
0x0010   aa81 a1d3 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 14a1 0000 636b 6f00 0000             P.......cko...
[root@localhost raw-logs]#
```

Tcpdump Options:

-r 2002.10.15 -Read data from this file 2002.10.15
-X -Full ASCII translation of HEX
nn -No resolution of host name or port numbers
e -Dump MAC Layer information
v -verbose output
c 5 -only want the first 5 packets
'host 255.255.255.255' –find packets with IP address = 255.255.255.255

Tcpdump output description:
16:34:10.596507: Date and Time
0:3:e3:d9:26:c0 0:0:c:4:b2:33: Source and Destination MAC addresses
0800: Encapsulated Protocol is IP
60: Length of the frame is 60 bytes
255.255.255.255.31337: Source IP and Port pair
> Going to
170.129.94.129.515: Destination IP and Port pair
R: Reset TCP flag set
[tcp sum ok]: Checksum is ok:
0:3(3): TCP Starting Sequence # : TCP Ending Sequence # (bytes of data)
ack 0 win 0: Acknowledgement # is 0 and TCP Window size is 0
[RST cko]: RST data is cko
ttl 15: Time To Live is 15
id 0: IP ID is 0
len 43: Datagram length is 43 bytes

3. *Probability the source address was spoofed*: The address of 255.255.255.255 is a broadcast address. Normal internet traffic should not come from a broadcast address; specifically, 255.255.255.255 is called a limited broadcast address, and should not be seen on the internet.[1] This address is one that should, in fact, be limited to the local LAN. Thus, assuming that the traffic sent to the destination address did, indeed, come from the internet, it would be rational to deduce that that the traffic is spoofed and crafted.

4. *Description of the attack*: This type of malicious attack is used to send commands to a compromised Q backdoor server. The Q program is a client/server backdoor which features a remote shell for root and regular users with encryption. There is also an on-demand TCP relay/bouncer with encryption. The program can be activated with raw packets.[2] Analysis indicates that the TCP source port, of all the attack packets in this case, are 31337 (ELEET to hacker's), and the destination port is 515, LPD port. Of equal importance, the Acknowledgement and Reset bit are also set in all attack packets which caused the alert. Thus, a packet with the Acknowledgement and Reset bit would terminate the communication as soon as the packet is received. Even though it appears to be a response to a SYN packet, it is highly unlikely that the packet is a response packet coming from a broadcast address of 255.255.255.255. A complete packet trace that contained all traffic to and from the source and destination would be necessary for absolute certainty.

Additionally, all the alert packets have a TTL of 14 or 15 and an IP ID of 0. While an IP ID of 0 is possible, the number should normally increase at some point and the TTL of 14 or 15 indicates a similar source for all of these packets.

There is a CVE, CVE CAN-1999-0660[3] currently under review, which relates to Trojan Backdoors in general. This attack targets random IP addresses on the monitored 170.129.0.0 /16 subnet at seemingly random times. Revealingly, the times are not consistent with any form of retry, which could indicate either a scan or a random attempt at eliciting a response to a planted Q Backdoor.

5. *Attack Mechanism*: Due to the fact that the packet has a spoofed or malformed source IP address and the reset flag is set, there can be no response expected over the current communication channel for this stimulus.

   The Q program uses a client/server arrangement, in which the Client connects to a server that has been already compromised and running the Q server program. The Q program then spawns a command shell back to the Client. The alerts under discussion could be a result of a scan searching for systems running Q. In particular, it is entirely probable that there was previous scanning on the filtering firewall/router, which discovered that port 515 was open, and, therefore, available for use as the attack vector. The analysis of the alerts indicates that the destination port 515 is open on the filtering firewall/router. Thus, the infiltrating packets would then be used as vectors to search for and trigger the backdoor mechanism on the compromised system(s) in order to "phone home" to receive further instructions. Regrettably, the complete communication from the "compromised" host to the Q client is not in existence, thus it is impossible to tell with any degree of certainty whether there was a response to the 31337 Q probe with the cko payload.

   Another possibility is that the probe is seeking a specific buffer overflow vulnerability in LPR. Analyses of the packet contents do not seem to justify this possibility, due to the fact that the packet does not conform to a standard buffer overflow appearance. A "standard" buffer overflow packet, if there is such a thing, would contain some type of repeated character such as: 41 41, 61 61 or 90 90 trying to fill the vulnerable buffer.[4] In this case, there were no repeated characters, NOP instructions or /bin/sh present in the packets. Interestingly, there are a number of vulnerabilities associated with the LPD service.[5] However, with the presence of the Reset flag and the absence of buffer overflow signs, it is unlikely that these packets are specifically designed for a buffer overflow infiltration.

   In essence, the Q backdoor is used to gain root shell access to a compromised system running the Q daemon program. These packets are probably an attempt to get a system infected with Q to initiate the backdoor function. In 2002, L. Gordon pursued extensive research into Q backdoor activity, and Mixter wrote various forms of the Q code and TFN and TFN2K DDoS tools.[6] Gordon speculates that this type of trace could be a result of hping, TFN, TFN2K or Q probes. He also mentions that we may never really know definitively what this trace is; it is a reasonable conclusion to draw, and one in which I concur.

6. *Correlations*: There are numerous reports of Q Backdoor activity from a multitude of sources. Primarily, there are several security related lists which have hosted discussions of Q Backdoor, and among which the following is a specific example: http://lists.jammed.com/incidents/2001/05/0039.html.[7] This particular thread provides no conclusive answer to the source of the traces that have shown up in their logs, and which correlate the behavior to IRC connectivity. One note speculated, in a convincing fashion, that this activity may have been related to a broken worm.[8] Numerous other GCIA practicals have been written about this particular Snort Signature/trace, specifically, Mario Ricci, Al Maslowski-Yerges and Les Gordon.[9,10&11] L. Gordon propounds that this probe is used primarily for reconnaissance, seeking potential vulnerable DDoS or Q servers waiting for commands. Additionally, M. Ricci in his GCIA paper tends to agree with the viewpoint expressed by L. Gordon. Maslowski-Yerges assert their hypothesis that they are Q probes. Finally, the CVE related to this vulnerability, CVE CAN-1999-0660, remains under review at present, and, is broad in scope, simply listing a few names of Trojans and other backdoor programs.[3]

7. *Evidence of active targeting*: There does not appear to be any active targeting of a specific host. In this example, the probes exist in a bookend fashion: a couple of days before, and a couple days after the specific trace to this subnet. They are spread across the entire class B network range (170.129.0.0/16) initiating at 170.129.1.20 up to and including 170.129.230.201 (log files examined 2002.10.13-2002.10.17). Of noteworthy interest, 170.129.0.0/16 is registered to SMC.[12] By examining the range of logs, it is possible to see other destination sub-nets. They are also random and at the class B level. It seems reasonable to deduce that the other class B network is due to the obfuscation of the logs intended to hide the IP addresses of the host networks by GIAC for practical purposes and/or a new set of scans with identical patterns and behavior on a new class B subnet, again not with any active targeting.

8. *Severity*: Severity is calculated with the following formula:
   Severity = (criticality + lethality) - (system counter measures + network countermeasures)

   Criticality=4: Not knowing anything about the systems that are available on this network link I will give a 4 because there could be systems that are very critical (ecommerce or DNS). If one of those systems were compromised then there are severe consequences.

   Lethality=3: Since no systems are directly targeted, however, if a system is compromised there is direct root access.

System countermeasures=2: Again, not knowing what systems were patched or the specific operating systems being probed, or whether there are any wrappers, Host Based IDS, or anti virus present.

Network countermeasures=2: The probe to TCP 515 was detected by the NID. One would have to assume that this probe was allowed through the perimeter router/firewall. Because of this, there is no filtering present to block the attack on TCP 515. I did not see any probes or hits on NetBIOS (TCP/UDP 137-139) which implies that there is an occurrence of some type of filtering. NetBIOS probes are extremely prevalent on exposed networks.

Severity = (4+3) – (2+2) =3

9. *Defensive recommendations*: Direct and effective recommendations are to block all unnecessary ports inbound. Currently, TCP 515 is allowed through perimeter defenses, and this should be blocked. A more secure approach would be to allow inbound only to the ports necessary, e.g., TCP port 80 for HTTP traffic, port TCP/UDP 53 for DNS, etc., and those specific to the site's requirements. Additionally, blocking source IP addresses of 255.255.255.255 and any other addresses within the private range specific to RFC 1918. If not already in place, Host Based IDS is strongly recommended, and, as always, maintaining up to date patches on all systems, in combination with the addition of anti-virus software, again installed on all systems.

10. *Multiple choice test question*: What is the appropriate response of a perimeter device receiving a packet with a source IP address of 255.255.255.255 to port 515 which has been properly secured?
    a) Reply with ICMP Net unreachable.
    b) Reply with ICMP Port unreachable.
    c) Allow the packet.
    d) Drop the packet.
    e) Send a Reset packet.

    Answer: d, drop the packet. Properly configured perimeter devices should drop broadcast traffic and not allow it to pass to other networks.

The top three questions garnered from posting the Trace and Analysis to Incidents.org on 24 January 2004. There was only one response which included several questions is incorporated below with my replies.
http://cert.uni-stuttgart.de/archive/intrusions/2004/01/msg00144.html

Questions start with @@@@ and my replies start with &&&&:

@@@@
@@@@ How many addresses were targeted? How many packets per address were sent?

@@@@ Was there any pattern in the way the destination addresses were selected? Or random?

&&&& The answer to these questions relating to Item 7, Evidence of Active Targeting is that there were 32 IP address with only one packet per IP address. All were different and apparently random with no apparent pattern.

@@@@ Any hosts responded?

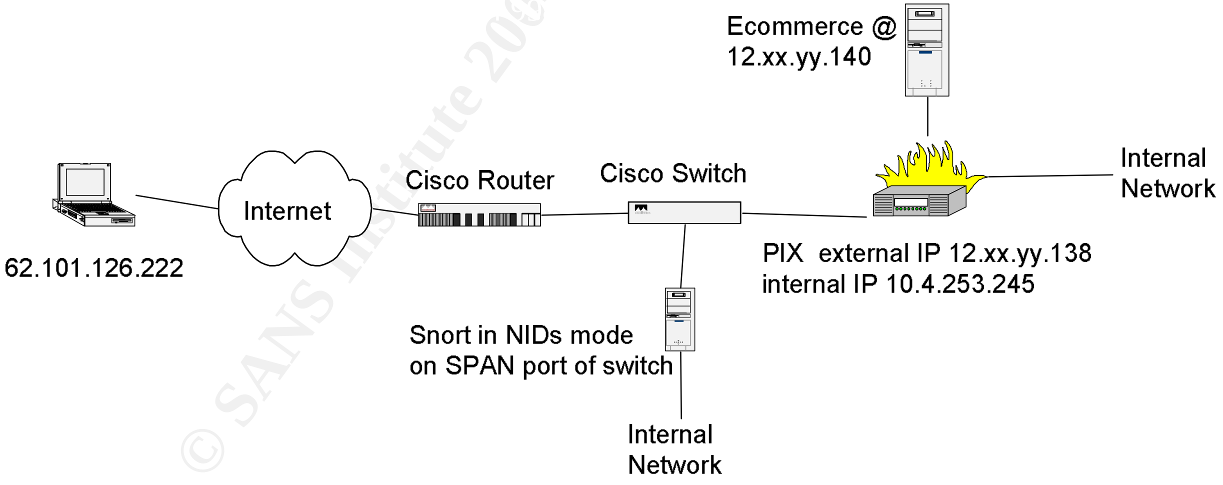
&&&& In response to this question, which corresponds with Item 4 is that I did mention within the body of the Trace and Analysis that I would have needed complete packet traces to determine whether any host responded ..

&&&& Additionally, it was also stated, at the beginning of the analysis, that "The log files are the result of a Snort instance running in binary logging mode". When Snort runs in binary logging mode the file that gets produced is only a log of the packet that caused the alert not the entire trace.

Detect #2: Proxy Scans

1. *Source of the trace*: In this case, the trace was extracted from my Employer's external ecommerce network from a binary capture of Snort on 26 October 2003.

   Network Diagram:



   The Snort Sensor that detected the alerts is a hardened Windows 2000 system with two network interface cards. The network interface card which faces the internet does not have an IP address. The other network interface card, the one facing the internal network is used for administrative purposes. The version of Snort utilized in this case is Version 2.0.0-ODBC-MySQL-FlexRESP-WIN32 (Build 72). The Snort rule set is that of a modified default rule set.

The following are excerpts from Firewall Logs from the firewall connecting the internet to the ecommerce webserver. These logs indicate that the scans to the proxy ports 8080, 1080 and 3182 were denied to the ecommerce server at 12.xx.yy.140.

```
2003-10-26 00:18:29        Local4.Error    10.4.253.245    Oct 26 2003 00:07:26: %PIX-3-106011:
Deny inbound (No xlate) icmp src outside:62.101.126.222 dst outside:12.xxx.yyy.139 (type 8, code 0)
2003-10-26 00:18:34        Local4.Warning 10.4.253.245    Oct 26 2003 00:07:31: %PIX-4-106023:
Deny tcp src outside:62.101.126.222/54071 dst dmz:12.xxx.yyy.140/8080 by access-group "acl_out"
2003-10-26 00:18:34        Local4.Warning 10.4.253.245    Oct 26 2003 00:07:31: %PIX-4-106023:
Deny tcp src outside:62.101.126.222/56318 dst dmz:12.xxx.yyy.140/3128 by access-group "acl_out"
2003-10-26 00:18:34        Local4.Warning 10.4.253.245    Oct 26 2003 00:07:31: %PIX-4-106023:
Deny tcp src outside:62.101.126.222/49162 dst dmz:12.xxx.yyy.140/1080 by access-group "acl_out"
2003-10-26 00:18:34        Local4.Warning 10.4.253.245    Oct 26 2003 00:07:31: %PIX-4-106023:
.
.
.
2003-10-26 00:19:30        Local4.Warning 10.4.253.245    Oct 26 2003 00:08:27: %PIX-4-106023:
Deny tcp src outside:62.101.126.222/15199 dst dmz:12.xxx.yyy.140/1080 by access-group "acl_out"
2003-10-26 00:19:36        Local4.Warning 10.4.253.245    Oct 26 2003 00:08:33: %PIX-4-106023:
Deny tcp src outside:62.101.126.222/15199 dst dmz:12.xxx.yyy.140/1080 by access-group "acl_out"
```

2. *Detect was generated by*: Initially, the detect was captured on an external system running Snort in NIDs mode. The detect was reanalyzed by Snort Version 2.0.4 (Build 96) running on a Linux Redhat 8 using the Snortrules-stable dated 19 November 2003. (Author’s Note: The time on the logs is out of sync.) A modified snort.conf was used, with all the rules turned on.

   The command used to detect the anomaly was:

   ```
   snort –r snort.log.1066671614 –c ./snort.conf –l output_log –k none –A full –edNUyX
   ```

   The following are the Snort options used:
   -r which file to read.
   -c which configuration file to use
   -l where to log the output
   -k none ignore Checksums
   -A full write the alert file with full decoded header as well as the alert message.
   -e displays the second layer header info
   -d dumps the application layer
   -N turn off logging
   -U uses UTC for timestamps
   -y puts the year in
   -X dump the raw packet date starting at the link layer

   The three snort rules that triggered the alerts were the following:

   ```
   alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS Proxy
   attempt"; flags:S,12; reference:url,help.undernet.org/proxyscan/;
   classtype:attempted-recon; sid:615; rev:4;)
   ```

alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"SCAN Squid Proxy attempt"; flags:S,12; classtype:attempted-recon; sid:618; rev:4;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy \(8080\) attempt"; flags:S,12; classtype:attempted-recon; sid:620; rev:3;)

These specific Snort rules are designed to trigger an alert on TCP activity from the External Network (in this case Any) from any port going to the Home Network (in this case Any) and specific ports 3128, 1080 or 8080. The alerts will include the messages: ‘SCAN Squid Proxy Attempt’, ‘SCAN SOCKS Proxy attempt’ or ‘SCAN Proxy (8080) attempt’ depending on which port was accessed. The rule also seeks any packet with the SYN flag bit set and find the SYN packets regardless of the values of the reserved bits with the flags:S,12 statement. All three rules have a class type of attempted-recon. The sid’s (Snort Rule Identifiers) 615, 618 and 620 are included in the Snort distribution, as presented hereafter:

Sid 615 states: “Improperly-configured SOCKS proxies can be abused to allow a hostile user to launch attacks and make them appear to come from your site. Additionally, if the proxy is behind a firewall or is a trusted host, it can be used to gain further access into your network and other hosts.”

Sid 618 states: “This event indicates that an attempt has been made to scan a host. This may be the prelude to an attack. Scanners are used to ascertain which ports a host may be listening on, whether or not the ports are filtered by a firewall and if the host is vulnerable to a particular exploit.”

Sid 620 states: The same as 618.
(http://www.snort.org/snort-db/sid.html)
Rev: denotes what revision each rule is in either 4 or 3.

The following are samples of the alerts generated by the Snort rule, only one of each is included for brevity. The destination IP addresses have been obfuscated:

```
[**] [1:620:3] SCAN Proxy (8080) attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/26/03-07:23:21.642533 0:D0:BC:30:98:1D -> 0:4:4D:4E:15:8C type:0x800
len:0x3E
62.101.126.222:54068 -> 12.xx.yy.138:8080 TCP TTL:104 TOS:0x0 ID:37666
IpLen:20 DgmLen:48 DF
******S* Seq: 0x31B716F4  Ack: 0x0  Win: 0xFAF0  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

[**] [1:618:4] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/26/03-07:23:21.652732 0:D0:BC:30:98:1D -> 0:4:4D:4E:15:8C type:0x800
len:0x3E
62.101.126.222:36862 -> 12.xx.yy.138:3128 TCP TTL:104 TOS:0x0 ID:37672
IpLen:20 DgmLen:48 DF
******S* Seq: 0x5BB168D5  Ack: 0x0  Win: 0xFAF0  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

```
[**] [1:615:4] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/26/03-07:23:21.656849 0:D0:BC:30:98:1D -> 0:4:4D:4E:15:8C type:0x800
len:0x3E
62.101.126.222:55198 -> 12.xx.yy.138:1080 TCP TTL:104 TOS:0x0 ID:37676
IpLen:20 DgmLen:48 DF
******S* Seq: 0xFF06EE3A  Ack: 0x0  Win: 0xFAF0  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
[Xref => http://help.undernet.org/proxyscan/]
```

Descriptions of the fields in the Snort Alert:
[Classification: Attempted Information Leak]
[Priority: 2]: low priority
10/26/03-07:23:21.642533: Date and time of the alert
0:D0:BC:30:98:1D -> 0:4:4D:4E:15:8C: Source and destination MAC address
type:0x800: Encapsulated Protocol is IP
len:0x3E: Length of the frame is 62 bytes:
62.101.126.222:54068 -> 12.xx.yy.138:8080: Source IP of 62.101.126.222 with a source port of 36862, 54068 or 55198 (and repeats during the scan, see Tcpdump output below) and destination IP address of 12.XX.YY.138 and destination port of 8080, 3128 or 1080.
TCP: TCP packet
TTL:104: Time To Live of 104
TOS:0x0: Type of Services bits are 0
ID:37666 or ID:37672 and ID:37676: IP ID is 37666, 37672 or 37676 (repeats)
IpLen:20: IP Packet header Length is 20 bytes
DgmLen:48: Total Datagram length is 48 bytes
DF: Don't Fragment bit is set
******S*: SYN bit are set
Seq: 0x31B716F4, Seq: 0x5BB168D5 and Seq: 0xFF06EE3A: Sequence numbers: (repeats)
Ack: 0x0: Acknowledgement ID 0:
Win: 0xFAF0: TCP Window size = 64240
TcpLen: 28: TCP Header Length is 28 bytes
TCP Options (4) => MSS: 1460 NOP NOP SackOK: Four TCP Options are set TCP Options: mss 1460 (maximum segment size 1460 bytes), nop, nop (no operations, filler), sackok (selective acknowledgement permitted):

[Xref) http://help.undernet.org/proxyscan/]: The last line of the SCAN SOCKS Snort alert gives a cross Reference.

The destination ports of 1080, 8080 and 3128 with the SYN bit set (regardless of the value of the reserved bits) are what triggered this alert.

Portscan PREPROCESSOR Config:
preprocessor portscan: $HOME_NET 5 4 portscan.log (5 Ports accessed 4 seconds)

Here is the pertinent portion of the PortScan Log:

```
Oct 26 07:23:24 62.101.126.222:54068 -> 12.XX.YY.138:8080 SYN ******S*
Oct 26 07:23:24 62.101.126.222:36862 -> 12.XX.YY.138:3128 SYN ******S*
Oct 26 07:23:24 62.101.126.222:55198 -> 12.XX.YY.138:1080 SYN ******S*
Oct 26 07:23:30 62.101.126.222:54071 -> 12.XX.YY.140:8080 SYN ******S*
Oct 26 07:23:24 62.101.126.222:56318 -> 12.XX.YY.140:3128 SYN ******S*
Oct 26 07:23:24 62.101.126.222:49162 -> 12.XX.YY.140:1080 SYN ******S*
Oct 26 07:24:14 62.101.126.222:63704 -> 12.XX.YY.138:1080 SYN ******S*
Oct 26 07:23:30 62.101.126.222:36862 -> 12.XX.YY.138:3128 SYN ******S*
Oct 26 07:23:30 62.101.126.222:54068 -> 12.XX.YY.138:8080 SYN ******S*
Oct 26 07:23:30 62.101.126.222:49162 -> 12.XX.YY.140:1080 SYN ******S*
Oct 26 07:23:30 62.101.126.222:56318 -> 12.XX.YY.140:3128 SYN ******S*
Oct 26 07:24:17 62.101.126.222:15199 -> 12.XX.YY.140:1080 SYN ******S*
Oct 26 07:24:23 62.101.126.222:63704 -> 12.XX.YY.138:1080 SYN ******S*
Oct 26 07:24:23 62.101.126.222:15199 -> 12.XX.YY.140:1080 SYN ******S*
```

A tcpdump of the binary file for a packet that caused the alert follows for further packet analysis (Destination IP addresses and Checksums have been obfuscated). Bolded texts are source and destination ports. Repeats are indicative of retries by their timing:

```
tcpdump -r snort.log.1066671614 -Xnnvvec 10 'host 62.101.126.222'
23:23:21.642533 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.54068 > 12.xx.yy.138.8080: S [tcp sum ok]
834082548:834082548(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl 104,
id 37666, len 48)
0x0000   4500 0030 9322 4000 6806 xxxx 3e65 7ede        E..0."@.h..B>e~.
0x0010   0cxx xx8a d334 1f90 31b7 16f4 0000 0000        .....4..1.......
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p....Z..........
23:23:21.652732 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.36862 > 12.xx.yy.138.3128: S [tcp sum ok]
1538353365:1538353365(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl
104, id 37672, len 48)
0x0000   4500 0030 9328 4000 6806 xxxx 3e65 7ede        E..0.(@.h..<>e~.
0x0010   0cxx xx8a 8ffe 0c38 5bb1 68d5 0000 0000        .......8[.h.....
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p...............
23:23:21.656849 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.55198 > 12.xx.yy.138.1080: S [tcp sum ok]
4278644282:4278644282(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl
104, id 37676, len 48)
0x0000   4500 0030 932c 4000 6806 xxxx 3e65 7ede        E..0.,@.h..8>e~.
0x0010   0cxx xx8a d79e 0438 ff06 ee3a 0000 0000        .......8...:....
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p...T...........
23:23:21.670591 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.54071 > 12.xx.yy.140.8080: S [tcp sum ok]
2986438417:2986438417(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl
104, id 37682, len 48)
0x0000   4500 0030 9332 4000 6806 xxxx 3e65 7ede        E..0.2@.h..0>e~.
0x0010   0cxx xx8c d337 1f90 b201 6f11 0000 0000        .....7....o.....
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p...............
23:23:21.697269 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.56318 > 12.xx.yy.140.3128: S [tcp sum ok]
3263356761:3263356761(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl
104, id 37722, len 48)
```

```
0x0000   4500 0030 935a 4000 6806 xxxx 3e65 7ede        E..0.Z@.h...>e~.
0x0010   0cxx xx8c dbfe 0c38 c282 df59 0000 0000        .......8...Y....
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p...............
23:23:21.706884 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.49162 > 12.xx.yy.140.1080: S [tcp sum ok]
770159858:770159858(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl 104,
id 37727, len 48)
0x0000   4500 0030 935f 4000 6806 xxxx 3e65 7ede        E..0._@.h...>e~.
0x0010   0cxx xx8c c00a 0438 2de7 b4f2 0000 0000        .......8-.......
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p...v...........
23:23:24.605516 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.54071 > 12.xx.yy.140.8080: S [tcp sum ok]
2986438417:2986438417(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl
104, id 39561, len 48)
0x0000   4500 0030 9a89 4000 6806 xxxx 3e65 7ede        E..0..@.h...>e~.
0x0010   0cxx xx8c d337 1f90 b201 6f11 0000 0000        .....7....o.....
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p...............
23:23:24.605744 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.54068 > 12.xx.yy.138.8080: S [tcp sum ok]
834082548:834082548(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl 104,
id 39555, len 48)
0x0000   4500 0030 9a83 4000 6806 xxxx 3e65 7ede        E..0..@.h...>e~.
0x0010   0cxx xx8a d334 1f90 31b7 16f4 0000 0000        .....4..1.......
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p....Z..........
23:23:24.606330 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.36862 > 12.xx.yy.138.3128: S [tcp sum ok]
1538353365:1538353365(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl
104, id 39558, len 48)
0x0000   4500 0030 9a86 4000 6806 xxxx 3e65 7ede        E..0..@.h...>e~.
0x0010   0cxx xx8a 8ffe 0c38 5bb1 68d5 0000 0000        .......8[.h.....
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p...............
23:23:24.606399 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.55198 > 12.xx.yy.138.1080: S [tcp sum ok]
4278644282:4278644282(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl
104, id 39559, len 48)
0x0000   4500 0030 9a87 4000 6806 xxxx 3e65 7ede        E..0..@.h...>e~.
0x0010   0cxx xx8a d79e 0438 ff06 ee3a 0000 0000        .......8...:....
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p...T...........
```

```
tcpdump -r snort.log.1066671614 -Xnnvvec 10 'host 62.101.126.222'
```

Tcpdump command line Options:
-r snort.log.1066671614 -Read data from this file snort.log.1066671614
-X -HEX dump of the packet with ASCII output
nn -No resolution of host name or port numbers
vv -very verbose output
e -MAC address output
c 10 -only want the first 10 packets
'host 62.101.126.222' -find packets with a host IP address = 62.101.126.222

```
tcpdump -r snort.log.1066671614 -Xnnvvec 10 'host 62.101.126.222'
23:23:21.642533 0:d0:bc:30:98:1d 0:4:4d:4e:15:8c 0800 62:
62.101.126.222.54068 > 12.xx.yy.138.8080: S [tcp sum ok]
834082548:834082548(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl 104,
id 37666, len 48)
```

```
0x0000   4500 0030 9322 4000 6806 xxxx 3e65 7ede        E..0."@.h..B>e~.
0x0010   0cxx xx8a d334 1f90 31b7 16f4 0000 0000        .....4..1.......
0x0020   7002 faf0 xxxx 0000 0204 05b4 0101 0402        p....Z..........
```

Tcpdump output description:
23:23:21.642533: Date and Time:
0:d0:bc:30:98:1d 0:4:4d:4e:15:8c: Source and Destination MAC addresses:
0800: Encapsulated Protocol is IP
62: Length of the frame is 62 bytes
62.101.126.222.54068: Source IP and Port pair
> Going to
12.xx.yy.138.8080**:** Destination IP and Port pair
S: SYN TCP flag set
[tcp sum ok]: Checksum is ok
834082548:834082548(0): TCP Starting Sequence #: TCP Ending Sequence # (bytes of data)
win 64240: TCP Window size is 64240
<mss 1460,nop,nop,sackOK>: TCP Options: mss 1460 (maximum segment size 1460 bytes), nop, nop (no operations, filler), sackok (selective acknowledgement permitted)
(DF): Don't Fragment bit set
ttl 104, Time To Live is 104
id 37666: IP ID is 37666
len 48: Datagram length is 48 bytes

3. *Probability the source address was spoofed*: In this case it is unlikely that the source address was spoofed. The purpose of the scan is to discover responsive proxy systems for use in further attacks. There would be no benefit to the malign instigator to spoof their address and send the replies to another system. As will be indicated in the correlation section, the detected IP address has been seen before.

4. *Description of the attack*: This type of attack is used as a preliminary scan of systems to determine their potential use as proxy hosts in order to launch further web-based attacks or anonymous browsing. The proxy is used as a mediator between the internet and the LAN. Its primary use would be to cache web pages to conserve bandwidth. The scans listed here are looking for Microsoft, Socks, Squid, Wingate, or some similar product on port 1080, 3128 or 8080.[13,14&15] For this event, this type of attack merely functions as a precursor to subsequent attacks. Alternatively, there are several known vulnerabilities found among the multitude of proxy implementations. It seems plausible to deduce that the perpetrator may be searching for a vulnerable system to compromise. All vulnerabilities were obtained from http://www.cve.mitre.org and http://www.cert.org.[16&17]

   - CVE-1999-0710 - The RedHat squid program installs cachemgr.cgi in a public web directory, allowing remote attackers to use it as an intermediary to connect to other systems.

- CVE-1999-1481 - Squid 2.2.STABLE5 and below, when using external authentication, allows attackers to bypass access controls via a newline in the user/password pair.
- CVE-2001-0142 - Squid 2.3 and earlier allows local users to overwrite arbitrary files via a symlink attack in some configurations.
- CVE-2002-0068 - Squid 2.4 STABLE3 and earlier allows remote attackers to cause a denial of service (core dump) and possibly execute arbitrary code with an ftp:// URL with a larger number of special characters, which exceed the buffer when Squid URL-escapes the characters.
- CAN-2002-1001 (under review) - Buffer overflows in AnalogX Proxy before 4.12 allows remote attackers to cause a denial of service and possibly execute arbitrary code via (1) a long HTTP request to TCP port 6588 or (2) a SOCKS 4A request to TCP port 1080 with a long DNS hostname.
- CVE-1999-0291 - The WinGate proxy is installed without a password, which allows remote attackers to redirect connections without authentication.
- CVE-2001-0239 - Microsoft Internet Security and Acceleration (ISA) Server 2000 Web Proxy allows remote attackers to cause a denial of service via a long web request with a specific type.
- CVE-2001-0658 - Cross-site scripting (CSS) vulnerability in Microsoft Internet Security and Acceleration (ISA) Server 2000 allows remote attackers to cause other clients to execute certain script or read cookies via malicious script in an invalid URL that is not properly quoted in an error message.
- CERT Vulnerability Note VN-98.03 - WinGate is a popular software package that allows a Local Area Network (LAN) to share a single Internet connection. The default configuration for WinGate allows an intruder to use a WinGate server to conceal his or her true location without the need to forge packets. In particular: WinGate enables all available network ports or services (this includes FTP, IRC, News, Telnet and WWW). And WinGate does not log connections.

After dumping the packets for 62.101.126.222, I ran p0f on the file to see whether I could determine the Operating System of the offending system; it turned out to potentially be a Windows system:

```
# p0f -s proxyscan
p0f: passive os fingerprinting utility, version 1.8.3
(C) Michal Zalewski <lcamtuf@gis.net>, William Stearns <wstearns@pobox.com>
p0f: file: '/etc/p0f.fp', 207 fprints, iface: 'eth0', rule: 'all'.
62.101.126.222 [25 hops]: Windows XP Pro, Windows 2000 Pro
62.101.126.222 [25 hops]: Windows XP Pro, Windows 2000 Pro
.
.
.
62.101.126.222 [25 hops]: Windows XP Pro, Windows 2000 Pro
62.101.126.222 [25 hops]: Windows XP Pro, Windows 2000 Pro
62.101.126.222 [25 hops]: Windows XP Pro, Windows 2000 Pro
```

5. *Attack Mechanism*: The packets seen in this attempted infiltration are specifically designed to scan for available proxy servers. The process of the scan is to send a SYN packet to a specific server via a particular port and wait for a SYN/ACK indicating an open proxy server.

6. *Correlations:* The broadly applied proxy scan in question was discovered and reported to DShield.org.[18] It is of interest to note that the scan on my Employer's network was found a day prior to DShield's report of the activity relating to this IP address. The timing inconsistencies could be due to non-synchronized logs synchronizing with an external time source.

IP Address: 62.101.126.222 **HostName:** 62-101-126-222.fastres.net **DShield Profile:**

Country:	IT
Contact E-mail:	abuse@fastweb.it
AS Number:	12874
Total Records against IP:	44735
Number of targets:	2395
Date Range:	2003-10-27 to 2003-11-30

Top 10 Ports Hit by this Source:

Port	Attacks	Start	End
1081	5574	2003-11-11	2003-11-30
1080	5030	2003-11-11	2003-11-30
8081	4885	2003-11-11	2003-11-30
3128	4411	2003-11-11	2003-11-30
8080	4185	2003-11-11	2003-11-30
80	3677	2003-11-11	2003-12-01
8000	3154	2003-11-13	2003-11-24
6882	69	2003-11-14	2003-11-14

5358	50	2003-11-28	2003-11-28
53	25	2003-11-11	2003-11-27

Last Fightback Sent: sent to abuse@fastweb.it on 2003-11-27 23:51:47
Whois:

```
inetnum:       62.101.126.208 - 62.101.126.223
netname:       FASTWEB-RESIDENTIAL-02
descr:         Infrastructure for Fastweb's main location
descr:         NAT IP addresses for residential customer, public subnet
country:       IT
admin-c:       IRS2-RIPE
tech-c:        IRS2-RIPE
status:        ASSIGNED PA
mnt-by:        FASTWEB-MNT
changed:       IP.RegistrationService@fastweb.it 20030821
remarks:       In case of improper use originating from our network,
remarks:       please mail customer or abuse@fastweb.it
remarks:       INFRA-AW
source:        RIPE

route:         62.101.96.0/19
descr:         Fastweb Networks block
origin:        AS12874
mnt-by:        FASTWEB-MNT
changed:       IP.RegistrationService@fastweb.it 20020404
remarks:       In case of improper use originating from our network,
remarks:       please mail customer or abuse@fastweb.it
source:        RIPE

person:        IP Registration Service
address:       Via Caracciolo, 51
address:       20155 Milano MI
address:       Italy
phone:         +39 02 45451
fax-no:        +39 02 45451
e-mail:        IP.RegistrationService@fastweb.it
nic-hdl:       IRS2-RIPE
remarks:
remarks:       In case of improper use originating from our network,
remarks:       please mail customer or abuse@fastweb.it
remarks:
notify:        IP.RegistrationService@fastweb.it
```

The CVE's related to the proxy scan are listed hereafter. These CVE's are specific to vulnerabilities relating to Microsoft, Socks, Squid and Wingate proxy servers and a CERT Vulnerability Note has additionally been included.[16&17]

- CVE-1999-0710 - The RedHat squid program installs cachemgr.cgi in a public web directory, allowing remote attackers to use it as an intermediary to connect to other systems.

- CVE-1999-1481 - Squid 2.2.STABLE5 and below, when using external authentication, allows attackers to bypass access controls via a newline in the user/password pair.
- CVE-2001-0142 - Squid 2.3 and earlier allows local users to overwrite arbitrary files via a symlink attack in some configurations.
- CVE-2002-0068 - Squid 2.4 STABLE3 and earlier allows remote attackers to cause a denial of service (core dump) and possibly execute arbitrary code with an ftp:// URL with a larger number of special characters, which exceed the buffer when Squid URL-escapes the characters.
- CAN-2002-1001 (under review) - Buffer overflows in AnalogX Proxy before 4.12 allows remote attackers to cause a denial of service and possibly execute arbitrary code via (1) a long HTTP request to TCP port 6588 or (2) a SOCKS 4A request to TCP port 1080 with a long DNS hostname.
- CVE-1999-0291 - The WinGate proxy is installed without a password, which allows remote attackers to redirect connections without authentication.
- CVE-2001-0239 - Microsoft Internet Security and Acceleration (ISA) Server 2000 Web Proxy allows remote attackers to cause a denial of service via a long web request with a specific type.
- CVE-2001-0658 - Cross-site scripting (CSS) vulnerability in Microsoft Internet Security and Acceleration (ISA) Server 2000 allows remote attackers to cause other clients to execute certain script or read cookies via malicious script in an invalid URL that is not properly quoted in an error message.
- CERT Vulnerability Note VN-98.03 - WinGate is a popular software package that allows a Local Area Network (LAN) to share a single Internet connection. The default configuration for WinGate allows an intruder to use a WinGate server to conceal his or her true location without the need to forge packets. In particular: WinGate enables all available network ports or services (this includes FTP, IRC, News, Telnet and WWW). And WinGate does not log connections.

Don Murdoch, Mark Embrich and Chris Calabrese all found similar type of scans triggering snort alerts all from various time periods.[19,20&21] Additionally, Mike Rondello found scans earlier this year.[22]

7. *Evidence of active targeting*: In this instance, there appears to be no active targeting of a specific host. This assertion is supported by the fact that there is no proxy service running on that subnet. Thus, it appears that an automated tool of some sort is utilized for the scan, which is in keeping with the conjecture that the scan was merely a fraction of a more encompassing scan. The possibility of a broader 'mother' scan was noted by the DShield report referred to in 6 above. In the case of my Employer's situation, there was activity only for the /29 subnet that the Snort sensor was attached to. Had additional IP addresses been routed to this network, I believe it is reasonable to assume that there would have been evidence of far more activity from the scanning host.

8. *Severity*: Severity is calculated with the following formula:

Severity = (criticality + lethality) - (system counter measures + network countermeasures)

Criticality=4: If there were a vulnerable proxy server available with anonymous proxy capability, bandwidth could be reduced and or there could be a buffer overflow vulnerability.

Lethality=4: DoS attack or buffer overflow could lead to no bandwidth or service and/or root compromise. Anonymous browsing is not considered in the lethality calculation.

System countermeasures=4: System is well patched and is a robust operating system with few known vulnerabilities. There is no proxy service running. However, there is no host based IDS.

Network countermeasures=4: The attack was stopped by a well configured firewall. The IDS detected the scan, but was allowed through at the perimeter router.

Severity = (4+4) – (4+4) = 0

9. *Defensive recommendations*: Recommendations in this case are straightforward and direct. Host based IDS, and block all unnecessary ports towards the firewall to the web site, i.e., block all but port 80 and 443 at the border router. Additionally, synchronize all log times with an external source.

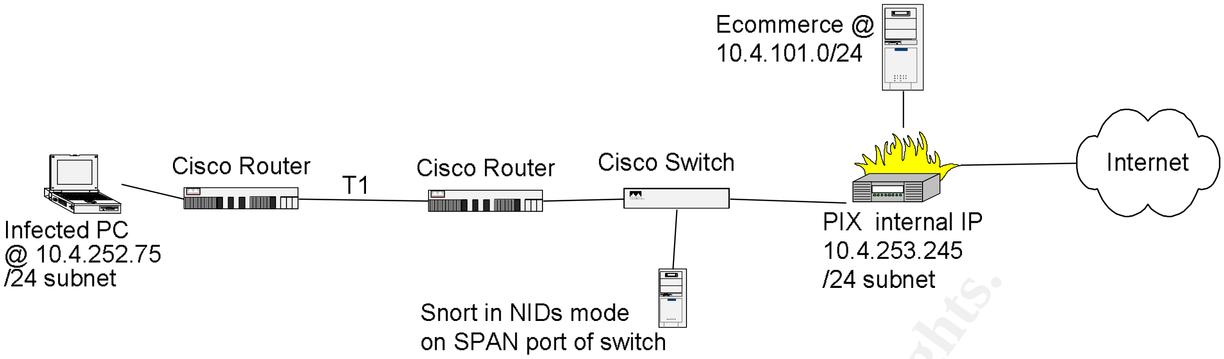
10. *Multiple choice test question:*
    What would be indicated by scans on port 1080, 8080 and 3128 directed at one of your servers?
    a. Scan for buffer overflows on vulnerable systems
    b. Wingate Proxy server scan
    c. Squid server scan
    d. Socks proxy scan
    e. All of the above

    Answer: e, All three of the ports are used in web proxies and are configurable, there are known buffer overflows in each of the services named.

Detect #3: Welchia MS03-026/MS03-039

1. *Source of the trace*: For the purposes of the detect discussion, the trace in question was taken from a binary capture of Snort on 20 November 2003 found in my Employer's internal network.

   Network Diagram:



The Snort sensor which detected this alert is a Linux RedHat 9.0 system running Snort Version 2.0.0 (Build 72) utilizing a custom rule set based on default rules.

Below are the logs taken from the PIX firewall which connects to the ecommerce webserver. These logs indicate an infected system (10.4.252.75) attempting to ping scan the ecommerce subnet (10.4.101.0/24) across the PIX firewall (10.4.253.245). The Welchia worm standard operational method is to attempt to find new hosts, initially by pinging the host, and then by attempting to connect on TCP port 135.[23]

```
2003-11-20 09:27:08      Local4.Warning 10.4.253.245    Nov 20 2003 10:14:02: %PIX-4-106023:
Deny icmp src inside:10.4.252.75 dst dmz:10.4.101.0 (type 8, code 0) by access-group "inside_out"
2003-11-20 09:27:08      Local4.Warning 10.4.253.245    Nov 20 2003 10:14:02: %PIX-4-106023:
Deny icmp src inside:10.4.252.75 dst dmz:10.4.101.1 (type 8, code 0) by access-group "inside_out"
2003-11-20 09:27:08      Local4.Warning 10.4.253.245    Nov 20 2003 10:14:02: %PIX-4-106023:
Deny icmp src inside:10.4.252.75 dst dmz:10.4.101.2 (type 8, code 0) by access-group "inside_out"
2003-11-20 09:27:08      Local4.Warning 10.4.253.245    Nov 20 2003 10:14:02: %PIX-4-106023:
.
.
.
2003-11-20 09:27:10      Local4.Warning 10.4.253.245    Nov 20 2003 10:14:04: %PIX-4-106023:
Deny icmp src inside:10.4.252.75 dst dmz:10.4.101.253 (type 8, code 0) by access-group
"inside_out"
2003-11-20 09:27:10      Local4.Warning 10.4.253.245    Nov 20 2003 10:14:04: %PIX-4-106023:
Deny icmp src inside:10.4.252.75 dst dmz:10.4.101.254 (type 8, code 0) by access-group
"inside_out"
2003-11-20 09:27:10      Local4.Warning 10.4.253.245    Nov 20 2003 10:14:04: %PIX-4-106023:
Deny icmp src inside:10.4.252.75 dst dmz:10.4.101.255 (type 8, code 0) by access-group
"inside_out"
```

2. *Detect was generated by*: This detect of the Welchia malicious worm was originally captured on an internal system running Snort in NIDs mode. The detect was then reanalyzed by Snort Version 2.0.4 (Build 96) running on a Linux Redhat 8 using the Snortrules-stable dated 19 November 2003. Please note that the time on the logs are out of sync, and that has been discussed previously. I used a modified snort.conf with all the rules turned on.

   The command used to detect the anomaly was: snort –r snort.log.1069348346 –c ./snort.conf –l output_log –k none –A full –edNUyX

The following snort options were used:
-r which file to read.
-c which configuration file to use
-l where to log the output
-k none ignore Checksums
-A full write the alert file with full decoded header as well as the alert message.
-e displays the second layer header info
-d dumps the application layer
-N turn off logging
-U uses UTC for timestamps
-y puts the year in
-X dump the raw packet date starting at the link layer

The snort rule that triggered the alerts was:

alert tcp any any -> any 135 (msg:"RPC Vulnerability 10-13-2003- bind initiation"; content:"|05 00 0B 03 10 00 00 00 48 00 00 00 7F 00 00 00 D0 16 D0 16 00 00 00 00 01 00 00 00 01 00 01 00 a0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 00 00 00 00 04 5D 88 8A EB 1C C9 11 9F E8 08 00 2B10 48 60 02 00 00 00|"; flow:to_server,established; classtype:attempted-admin; sid:1; rev:1;)

This Snort rule is designed to issue an alert relating to TCP activity from any source IP address to any port going to any destination IP address and port 135. The alerts will include the message: ‘RPC Vulnerability 10-13-2003- bind initiation’. Additionally, the rule is searches for a content of ‘05 00 0B 03 10 00 00 00 48 00 00 00 7F 00 00 00 D0 16 D0 16 00 00 00 00 01 00 00 00 01 00 01 00 a0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 00 00 00 00 04 5D 88 8A EB 1C C9 11 9F E8 08 00 2B10 48 60 02 00 00 00’ within the payload of the packet. The alert is set to trigger on established client requests to the server with ‘flow:to_server,established’. There is an attack classification assigned, specifically, ‘attempted-admin’ = attempted administrator privilege gain. Snort rules unique id and revision are 1 ‘sid:1; rev:1’. The sid does not correspond to an active sid on the Snort web site

Below are samples of the alerts generated by the Snort rule. Only three alerts from each of the beginning and end of the attack are included for brevity’s sake.

```
[**] [1:1:1] RPC Vulnerability 10-13-2003- bind initiation [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
11/20/03-18:03:27.839317 0:50:E2:E5:B0:0 -> 0:B0:4A:5A:C4:0 type:0x800 len:0x7E
10.4.252.75:3370 -> 10.4.3.232:135 TCP TTL:126 TOS:0x0 ID:39952 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x4937712F  Ack: 0x797B161  Win: 0x4470  TcpLen: 20

[**] [1:1:1] RPC Vulnerability 10-13-2003- bind initiation [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
11/20/03-18:03:28.136312 0:50:E2:E5:B0:0 -> 0:B0:4A:5A:C4:0 type:0x800 len:0x7E
10.4.252.75:3379 -> 10.4.4.35:135 TCP TTL:126 TOS:0x0 ID:40062 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x493F0377  Ack: 0x375B2823  Win: 0x4470  TcpLen: 20

[**] [1:1:1] RPC Vulnerability 10-13-2003- bind initiation [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
11/20/03-18:03:28.261997 0:50:E2:E5:B0:0 -> 0:B0:4A:5A:C4:0 type:0x800 len:0x7E
```

```
10.4.252.75:3382 -> 10.4.4.42:135 TCP TTL:126 TOS:0x0 ID:40065 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x4941BCE9  Ack: 0xFF64F1  Win: 0x4470  TcpLen: 20
.
.
.
[**] [1:1:1] RPC Vulnerability 10-13-2003- bind initiation [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
11/20/03-21:01:24.153799 0:50:E2:E5:B0:0 -> 0:6:29:A8:A2:FB type:0x800 len:0x7E
10.4.244.36:2929 -> 10.4.253.234:135 TCP TTL:127 TOS:0x0 ID:45651 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x46B79168  Ack: 0x79644359  Win: 0xFFFF  TcpLen: 20

[**] [1:1:1] RPC Vulnerability 10-13-2003- bind initiation [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
11/20/03-21:01:24.192625 0:50:E2:E5:B0:0 -> 0:6:29:A8:8F:CD type:0x800 len:0x7E
10.4.244.36:2930 -> 10.4.253.238:135 TCP TTL:127 TOS:0x0 ID:45667 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x46B8AFA7  Ack: 0x3C78BA10  Win: 0xFFFF  TcpLen: 20

[**] [1:1:1] RPC Vulnerability 10-13-2003- bind initiation [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
11/20/03-21:01:24.242915 0:50:E2:E5:B0:0 -> 0:10:B5:DB:98:49 type:0x800 len:0x7E
10.4.244.36:2934 -> 10.4.253.243:135 TCP TTL:127 TOS:0x0 ID:45688 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x46BC046B  Ack: 0xD8344FCD  Win: 0xFFFF  TcpLen: 20
```

Descriptions of the fields in the Snort Alert:
11/20/03-18:03:27.839317: Date and time of the alert
0:50:E2:E5:B0:0 -> 0:2:55:54:E6:70: Source and destination MAC address
type:0x800: Encapsulated Protocol is IP
len:0x7E: Length of the frame is 126 bytes
10.4.252.75:3370 -> 10.4.3.232:135: Source IP of 10.4.252.75 with a source port of 3370 for this packet
10.4.3.232 and destination port of 135 for all alerts
TCP: TCP packet
TTL:104: Time To Live of 15
TOS:0x0: Type of Services bits are 0
ID:39952: IP ID is 39952 (always different)
IpLen:20: IP Packet header Length is 20 bytes
DgmLen:112: Total Datagram length is 112 bytes
DF: Don’t Fragment bit is set
AP ACK and PUSH bit are set
Seq: 0x4937712F: Sequence numbers (always different)
Ack: 0x797B161: Acknowledgement ID (always different)
Win: 0x4470: TCP Window size = 17520
TcpLen: 20: TCP Header Length is 20 bytes

The alert was triggered by traffic coming to the destination IP address on TCP port 135 with the following contents: 05 00 0B 03 10 00 00 00 48 00 00 00 7F 00 00 00 D0 16 D0 16 00 00 00 00 01 00 00 00 01 00 01 00 a0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 00 00 00 00 04 5D 88 8A EB 1C C9 11 9F E8 08 00 2B10 48 60 02 00 00 00 in the packet, with an established connection, towards the server.

A tcpdump of the binary file for a packet that caused the alert follows for further packet analysis):

```
# tcpdump -r snort.log.1069348346 port 135 -Xnnvvec 5
```

```
10:03:27.839317 0:50:e2:e5:b0:0 0:b0:4a:5a:c4:0 0800 126:
10.4.252.75.3370 > 10.4.3.232.135: P [tcp sum ok]
1228370223:1228370295(72) ack 127381857 win 17520 (DF) (ttl 126, id
39952, len 112)
0x0000   4500 0070 9c10 4000 7e06 4c3c 0a04 fc4b        E..p..@.~.L<...K
0x0010   0a04 03e8 0d2a 0087 4937 712f 0797 b161        .....*..I7q/...a
0x0020   5018 4470 8cdf 0000 0500 0b03 1000 0000        P.Dp............
0x0030   4800 0000 7f00 0000 d016 d016 0000 0000        H...............
0x0040   0100 0000 0100 0100 a001 0000 0000 0000        ................
0x0050   c000 0000 0000 0046 0000 0000 045d 888a        .......F.....]..
0x0060   eb1c c911 9fe8 0800 2b10 4860 0200 0000        ........+.H`....
10:03:28.136312 0:50:e2:e5:b0:0 0:b0:4a:5a:c4:0 0800 126:
10.4.252.75.3379 > 10.4.4.35.135: P [tcp sum ok]
1228866423:1228866495(72) ack 928720931 win 17520 (DF) (ttl 126, id
40062, len 112)
0x0000   4500 0070 9c7e 4000 7e06 4b93 0a04 fc4b        E..p.~@.~.K....K
0x0010   0a04 0423 0d33 0087 493f 0377 375b 2823        ...#.3..I?.w7[(#
0x0020   5018 4470 53c6 0000 0500 0b03 1000 0000        P.DpS...........
0x0030   4800 0000 7f00 0000 d016 d016 0000 0000        H...............
0x0040   0100 0000 0100 0100 a001 0000 0000 0000        ................
0x0050   c000 0000 0000 0046 0000 0000 045d 888a        .......F.....]..
0x0060   eb1c c911 9fe8 0800 2b10 4860 0200 0000        ........+.H`....
10:03:28.261997 0:50:e2:e5:b0:0 0:b0:4a:5a:c4:0 0800 126:
10.4.252.75.3382 > 10.4.4.42.135: P [tcp sum ok]
1229044969:1229045041(72) ack 16737521 win 17520 (DF) (ttl 126, id 40065,
len 112)
0x0000   4500 0070 9c81 4000 7e06 4b89 0a04 fc4b        E..p..@.~.K....K
0x0010   0a04 042a 0d36 0087 4941 bce9 00ff 64f1        ...*.6..IA....d.
0x0020   5018 4470 93d5 0000 0500 0b03 1000 0000        P.Dp............
0x0030   4800 0000 7f00 0000 d016 d016 0000 0000        H...............
0x0040   0100 0000 0100 0100 a001 0000 0000 0000        ................
0x0050   c000 0000 0000 0046 0000 0000 045d 888a        .......F.....]..
0x0060   eb1c c911 9fe8 0800 2b10 4860 0200 0000        ........+.H`....
10:03:29.837022 0:50:e2:e5:b0:0 0:b0:4a:5a:c4:0 0800 126:
10.4.252.75.3387 > 10.4.4.232.135: P [tcp sum ok]
1229770232:1229770304(72) ack 181904456 win 17520 (DF) (ttl 126, id
40196, len 112)
0x0000   4500 0070 9d04 4000 7e06 4a48 0a04 fc4b        E..p..@.~.JH...K
0x0010   0a04 04e8 0d3b 0087 494c cdf8 0ad7 a448        .....;..IL.....H
0x0020   5018 4470 38c9 0000 0500 0b03 1000 0000        P.Dp8...........
0x0030   4800 0000 7f00 0000 d016 d016 0000 0000        H...............
0x0040   0100 0000 0100 0100 a001 0000 0000 0000        ................
0x0050   c000 0000 0000 0046 0000 0000 045d 888a        .......F.....]..
0x0060   eb1c c911 9fe8 0800 2b10 4860 0200 0000        ........+.H`....
10:03:31.620825 0:50:e2:e5:b0:0 0:b0:4a:5a:c4:0 0800 126:
10.4.252.75.3398 > 10.4.5.232.135: P [tcp sum ok]
1230729718:1230729790(72) ack 3244084669 win 17520 (DF) (ttl 126, id
40468, len 112)
0x0000   4500 0070 9e14 4000 7e06 4838 0a04 fc4b        E..p..@.~.H8...K
0x0010   0a04 05e8 0d46 0087 495b 71f6 c15c cdbd        .....F..I[q..\..
0x0020   5018 4470 b3b6 0000 0500 0b03 1000 0000        P.Dp............
0x0030   4800 0000 7f00 0000 d016 d016 0000 0000        H...............
```

```
0x0040   0100 0000 0100 0100 a001 0000 0000 0000        ................
0x0050   c000 0000 0000 0046 0000 0000 045d 888a        .......F.....]..
0x0060   eb1c c911 9fe8 0800 2b10 4860 0200 0000        ........+.H`....
```

tcpdump -r snort.log.1069348346 port 135 -Xnnvvec 5

Tcpdump Options:
-r snort.log.1069348346 -Read data from this file snort.log.1069348346
-port 135 – look for traffic on port 135
-X -Full ASCII translation of HEX
nn -No resolution of host name or port numbers
vv –very verbose output
e -Dump MAC Layer information
c 5 –only want the first 5 packets

```
10:03:27.839317 0:50:e2:e5:b0:0 0:b0:4a:5a:c4:0 0800 126:
10.4.252.75.3370 > 10.4.3.232.135: P [tcp sum ok]
1228370223:1228370295(72) ack 127381857 win 17520 (DF) (ttl 126, id
39952, len 112)
0x0000   4500 0070 9c10 4000 7e06 4c3c 0a04 fc4b        E..p..@.~.L<...K
0x0010   0a04 03e8 0d2a 0087 4937 712f 0797 b161        .....*..I7q/...a
0x0020   5018 4470 8cdf 0000 0500 0b03 1000 0000        P.Dp............
0x0030   4800 0000 7f00 0000 d016 d016 0000 0000        H...............
0x0040   0100 0000 0100 0100 a001 0000 0000 0000        ................
0x0050   c000 0000 0000 0046 0000 0000 045d 888a        .......F.....]..
0x0060   eb1c c911 9fe8 0800 2b10 4860 0200 0000        ........+.H`....
```

Tcpdump output description:
10:03:27.839317: Date and Time
0:50:e2:e5:b0:0 0:b0:4a:5a:c4:0: Source and Destination MAC addresses
0800: Encapsulated Protocol is IP
126: Length of the frame is 126 bytes
10.4.252.75.3370: Source IP and Port pair
> Going to
10.4.3.232.135: Destination IP and Port pair
P: Push TCP flag set:
[tcp sum ok]: Checksum is ok:
1228370223:1228370295(72): TCP Starting Sequence #: TCP Ending Sequence # (bytes of data)
ack 127381857: Acknowledgement ID
win 17520: TCP Window size is 17520
(DF): Don't Fragment bit set
ttl 126: Time To Live is 126:
id 39952: IP ID is 39952
len 112: Datagram length is 112 bytes
0500 0b03…: TCP Packet contents start
0200 0000…: TCP Packet contents stop.

3. *Probability the source address was spoofed*: The probability that the source address was spoofed is quite low considering the fact that the attack is a malicious worm propagation attempt. It's purpose is to infect additional systems with the worm payload and move on. The Snort rule requires that the communication be established prior to the issuance of an alert, therefore, there must be a 3-way handshake and connectivity. It is known from the behavior of the worm that there is an ICMP echo request followed by a connection to TCP port 135 and then the exploit of the buffer overflow (discussed below in attack mechanism). In this case, I regrettably do not have a full trace of the 3-way handshake, and cannot therefore confirm that there was such a process. However, despite this, knowledge that the system did possess the worm files, it was at the IP address 10.4.252.75 and was subsequently and successfully cleaned of the worm, leads to the fairly safe assertion that the address was, in all probability, not spoofed.

4. *Description of the attack*: This purpose of this attack is the replication of a malicious worm. This version of the Welchia worm initially sends an ICMP echo request for potential victims, then connects to TCP 135 and implements a buffer overflow on the victim system.

   The infected system was that of a consultant's Windows 2000 SP2 laptop, whose antivirus software and system patches had been allowed to lapse. The initial infection occurred off site. Once the consultant had connected to the network on premises, the malicious worm program began the attempt to infiltrate its way toward the IDS and firewalls. The infected pc began the process by issuing ICMP echo requests, and when the worm program found a receptive system on TCP 135 it connected and attempted to trigger a buffer overflow on vulnerable systems. The Tcpdump trace illustrates the effects of an infected system attempting to infect others.

   CVE: Description: Buffer overflow in a certain DCOM interface for RPC in Microsoft Windows NT 4.0, 2000, XP, and Server 2003 allows remote attackers to execute arbitrary code via a malformed message, as exploited by the Blaster/MSblast/LovSAN and Nachi/Welchia worms.[24]

5. *Attack Mechanism*. An infected system with the Welchia worm begins probing the local network with ICMP echo requests to the local network. Once the worm receives a reply it attempts to connect utilizing a 3-way handshake to TCP 135 (RPC-DCOM). If successful, the worm then attempts to exploit a buffer overflow in the RPC-DCOM system. The faulty buffer exists in un-patched Windows 2000, XP and 2003 systems. The patch that repairs this buffer overflow is referenced by MS03-026 and MS03-039.[25&26] The specific buffer overflow is caused by an unchecked parameter within the DCOM function. Once the buffer overflow has been exploited, a command shell is listening on TCP 4444, 666 or 765. The worm then has the victim tftp the worm code to the victim system and the cycle continues. Code for this attack is readily available on the web in various formats.[27]

6. ***Correlations:*** There have been numerous accounts of the RPC DCOM worm MS Blast in various forms, of which Welchia is one, reported by GCIH students Linda Bourbeau and Brian Porter.[28&29] Note that both Porter and Bourbeau experienced infestations, and discovered similar traces on their networks. Additionally, David Markel found MS Blast type packets in the http://www.incidents.org/logs/Raw/log_22_aug.raw log file.[30] Joanne Schell wrote about the worm in GCIA paper.[31] The PIX firewall logs showing 10.4.252.75 attempting to ping scan other networks. In August 2003, my Employer experienced an MSBlast/Welchia outbreak. Cert identified the vulnerability in Vulnerability Note 568148 and Cert Advisory CA-2003-19[32&33]

7. *Evidence of active targeting*: The Welchia worm does not actively seek out a specific IP address. Rather, it scans for receptive, unpatched systems listening on TCP port 135 across entire networks. Once the target is located, it is actively attacked and exploited.

8. *Severity*: Severity is calculated with the following formula:
   Severity = (criticality + lethality) - (system counter measures + network countermeasures)

   Criticality=5: Since this attack will infect any Windows system that is not patched. DNS, Mail, Database, etc.

   Lethality=5: Root compromise and potential DoS.

   System countermeasures=1: Infected system had not been patched and the anti-virus was out of date.

   Network countermeasures=3: The attack was stopped by a well configured firewall to other networks. The worm was detected and tracked down to the user by active monitoring by the Intrusion Analyst. However, the system was allowed on the network without proper patches or up-to-date anti-virus.

   Severity = (5+5) – (1+2) = 7

9. *Defensive recommendations*: Straightforward recommendations would be to ensure that company's employing consultants would require consultants to sign an agreement to the effect that their systems are patched and have current antivirus signatures. TCP Port 135 cannot be blocked because Windows Active Directory is used across the WAN for AD replication. ICMP blocking could be implemented, but creates network engineering difficulties.

10. *Multiple choice test question*: What worm came out in August of 2003 that exploited the MS RPC/DCOM service?
    a. bugbear
    b. swen

c. blast/welchia
d. slammer
e. sobig

Answer: c.

Section 3: Analyze This

Executive Summary:

The analysis discussion relates to the network security events taking place over the course of a consecutive five-day period,19 December 2003 through 23 December 2003, as seen by the 'host' network's intrusion detection system. Accompanying information included the fact that this system utilizes an "older" version of Snort Intrusion Detection system, and it became apparent, from perusing the alerts, that the system had been customized to meet the requirements of the University of Maryland at Baltimore. The intent, however, is to analyze the data presented from the perspective of identifying the areas of greatest vulnerability, and to discuss whether the current intrusion detection rules apply, as well as the effectiveness of these same rules. Additionally, recommendations will be made, which, if implemented, would maximize the intrusion detection system to the University's benefit.

There were 89,778 alerts, 17,627,342 scans and 1,177 Out of Specification alerts during the course of the five day data sampling. Given the dates that the security events took place; it is readily apparent that the University is less active during their winter break than during regular session.

It becomes obvious from an examination of the files presented, that the University places an emphasis on their network security. However, there are some immediate areas of concern. The primary issue found was that there appears to be several systems within the University's network which are infected with variants of the MS Blast worm (Welchia/Nachi). Additionally, there is a potentially compromised system with a Trojan backdoor, Game Server, and a couple of Peer-to-Peer (P2P) systems. Complete recommendations for further protective customization can be found in the section titled Recommendations and Discussion, below.

The analysis conforms to the design hereafter. Beginning with a list of files, the next step is to examine the alert files, focus on the top 10 most frequently occurring alerts, and to analyze each. Following this outlined procedure with Alerts, the next section discusses the Scans data, then followed by the Out of Spec alerts. A discussion of relevant findings and recommendations for each will follow the analyses..

List of files: The following files are those that were analyzed:

Alert.031219.gz	scans.031219.gz	oos_report_031219
Alert.031220.gz	scans.031220.gz	oos_report_031220
Alert.031221.gz	scans.031221.gz	oos_report_031221
Alert.031222.gz	scans.031222.gz	oos_report_031222

Alert.031223.gz scans.031223.gz oos_report_031223

Examples of raw files contents:

Alert.031219

```
12/23-00:01:36.914157  [**] ICMP SRC and DST outside network [**] 172.158.185.230 ->
172.157.112.213
12/23-00:01:43.007736  [**] High port 65535 udp - possible Red Worm - traffic [**] 207.5.180.138:65535 -
> 130.85.1.4:53
12/23-00:01:52.499472  [**] EXPLOIT x86 NOOP [**] 130.67.227.123:3226 -> 130.85.190.95:135
```

Scans.031219

```
Dec 19 00:00:00 130.85.163.107:3541 -> 131.88.40.38:135 SYN ******S*
Dec 19 00:00:00 130.85.163.107:3542 -> 131.88.40.39:135 SYN ******S*
Dec 19 00:00:00 130.85.163.107:3543 -> 131.88.40.40:135 SYN ******S*
```

Oos_report_031219

```
12/23-00:05:06.113439 63.194.83.210:2960 -> 130.85.34.11:80
TCP TTL:49 TOS:0x0 ID:15287 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x28DAE4C1  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3530614 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/23-00:05:47.654922 147.232.24.11:50894 -> 130.85.34.14:113
TCP TTL:51 TOS:0x0 ID:34000 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x253A11EB  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 1144952123 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/23-00:05:57.507763 147.232.24.11:50896 -> 130.85.34.14:113
TCP TTL:51 TOS:0x0 ID:61642 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x264469DE  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 1144953108 0 NOP WS: 0

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

Definition of Terms:

The following terms may have achieved an accepted standard definition for usage as jargon; however, for the purposes of the analyses herein, they are defined as follows:

Severity: Relative scale; specifically measurements being high, medium, and low severity based on alert type, ports examined, and potential for system compromise.

False positive: Relative scale; high, medium, and low possibility of false positive. Given that I do not have access to the rules, it is assumed that these alerts are triggered on real events. The false positive statement gives weight to whether there is potential sophistication in the rule writing. A rule that appears to trigger on a source or destination port alone would lend itself to a higher probability of “false positive” than one examining the packet contents, the former leads to lots of Noise.

Noise: Scale 1-5 with 5 being very noisy. Alerts that send out an alarm that is really non-threatening, but was correct for the rule. I.e., the rule is telling you exactly what you ask for but is relatively useless for security purposes.[34]

Assumptions:

All data being analyzed for the purposes of this documentation and discussion are assumed to have been sourced from the same network. Data is prepared in accordance with the steps set forth in the Methods of Analysis Section, hereafter. All MY.NET were changed to 130.85 for the sake of consistency. It was possible to disregard the port scans contained in the alert files due to the fact that there was direct access to the scan log files. Please see the Methods of Analysis Section for data manipulation and analysis processes.

List of Alerts:

Table 1 Total number of Alerts per day:

15212	Dec 23
22833	Dec 22
14826	Dec 21
18695	Dec 20
18173	Dec 19

Total number of Alerts 89778 ± some anomalous data points.

Table 2 List of all unique alerts:

23810	130.85.30.3 activity
21855	130.85.30.4 activity
13669	Incomplete Packet Fragments Discarded
7869	TFTP - Internal TCP connection to external tftp server
4713	EXPLOIT x86 NOOP
4343	SMB Name Wildcard
3557	Connect to 515 from inside
3242	High port 65535 udp - possible Red Worm - traffic
1713	ICMP SRC and DST outside network
1696	NMAP TCP ping!
1086	High port 65535 tcp - possible Red Worm - traffic
662	Null scan!
327	Possible trojan server activity
270	TCP SRC and DST outside network
172	[UMBC NIDS IRC Alert] IRC user /kill detected
171	SUNRPC highport access!

118	FTP passwd attempt
107	SMB C access
79	[UMBC NIDS] External MiMail alert
45	EXPLOIT x86 setuid 0
38	scan (Externally-based)
33	EXPLOIT x86 setgid 0
30	RFB - Possible WinVNC - 010708-1
29	FTP DoS ftpd globbing
16	TFTP - External TCP connection to internal tftp server
11	EXPLOIT NTPDX buffer overflow
10	Tiny Fragments - Possible Hostile Activity
9	EXPLOIT x86 NOPS
8	EXPLOIT x86 stealth noop
8	IRC evil - running XDCC
8	Probable NMAP fingerprint attempt
8	Attempted Sun RPC high port access
7	TFTP - Internal UDP connection to external tftp server
7	TFTP - External UDP connection to internal tftp server
5	External FTP to HelpDesk 130.85.53.29
5	External FTP to HelpDesk 130.85.70.49
5	DDOS mstream client to handler
5	DDOS shaft client to handler
5	External FTP to HelpDesk 130.85.70.50
5	[UMBC NIDS IRC Alert] K\:line'd user detected
4	NIMDA - Attempt to execute cmd from campus host
4	[UMBC NIDS IRC Alert] User joining Warez channel detect...
3	[UMBC NIDS IRC Alert] User joining XDCC channel detecte...
2	EXPLOIT identd overflow
1	Bugbear@MM virus in SMTP
1	Happy 99 Virus
1	Possible wu-ftpd exploit - GIAC000623
1	[UMBC NIDS IRC Alert] Possible sdbot floodnet detected ...
1	TCP SMTP Source Port traffic
1	Traffic from port 53 to port 123
1	PHF attempt
Total 52	Total Alerts 89778

Table 3 Top 10 Alerts, ranked by number of occurrence:

7869	TFTP - Internal TCP connection to external tftp server
4713	EXPLOIT x86 NOOP
4343	SMB Name Wildcard
3557	connect to 515 from inside
3242	High port 65535 udp - possible Red Worm - traffic
1713	ICMP SRC and DST outside network

1696	NMAP TCP ping!
1086	High port 65535 tcp - possible Red Worm - traffic
662	Null scan!
327	Possible trojan server activity

Detailed analysis of each of the top 10 alerts:

It is important to note that the top three alerts (130.85.30.3 activity; 130.85.30.4 activity, and Incomplete Packet Fragments Discarded) are analyzed separately.

1. **TFTP - Internal TCP connection to external tftp server**	Severity: Low	Reported: 7869
False Positive: High. General rule to a port.	Snort signature ID: None	Noise: 3

Example Alerts:

```
12/19-08:31:06.864717  [**] TFTP - Internal TCP connection to external tftp server [**]
130.85.70.225:1736 -> 68.61.18.36:69
12/19-08:31:06.912595  [**] TFTP - Internal TCP connection to external tftp server [**] 68.61.18.36:69 ->
130.85.70.225:1736
```

Sample Rule: alert tcp $HOME_NET any <> $EXTERNAL_NET 69 (msg: “TFTP - Internal TCP connection to external tftp server;)

These alerts serve as notification of TFTP activity from within the network connected to an external source. This is indicating that there is activity on TCP port 69 from inside to the outside of the network. The alerts could indicate something as simple as a router performing updates or other hardware systems receiving updates.

The most active IP address for this alert is, as follows:

```
$ grep "TFTP - Internal TCP connection to external tftp server" alert6.mrg | awk -F :
'{print $5}' | sort | uniq -c | sort -rn | head
   3024  69.10.132.121
   2295  130.85.42.1
   1990  130.85.42.3
    304  130.85.70.225
    236  68.61.18.36
     16  130.85.82.109
      2  130.85.60.16
      1  69.10.132.121
      1  66.57.196.184
      1  217.59.186.109
```

There appears to be only one primary external host involved with the majority of the alerts, in fact, almost all the TFTP transactions; six of the top ten involve 69.10.132.121.

The most active pairs of IP addresses are:

```
$ grep "TFTP - Internal TCP connection to external tftp server" alert6.mrg | awk -F : '{print $5,$7}' | sort |
uniq -c | sort -rn | head
   2295  130.85.42.1  69.10.132.121
   1990  130.85.42.3  69.10.132.121
   1520  69.10.132.121  130.85.42.1
   1493  69.10.132.121  130.85.42.3
    304  130.85.70.225  68.61.18.36
    236  68.61.18.36  130.85.70.225
     16  130.85.82.109  69.10.132.121
     11  69.10.132.121  130.85.82.109
      2  130.85.60.16  66.160.63.18
```

(last line in output had bad data and was removed)

When: The TFTP alerts occurred in a chronological manner commencing 19 December 2003 and finalizing 23 December 2003. Specifically, on 19 December 2003, the alerts started at 0831 and continued until 1515, for a total of 543 alerts. On 20 December 2003 the alerts started at 1433 and continued until 2327, approximately 3-5 per minute, for a total of 3215 alerts. On 21 December 2003, there were no alerts. On 22 December 2003 the alerts started at 1650 and continued until 2144, for a total of 1263 alerts. On 23 December 2003 the alerts started at 0714 and continued until 2347, for a total of 2849 alerts.

Dshield[18] Results: There is no indication of previous malicious behavior from these IP addresses.

MyNetWatchman[35]: Incident ID: 57196252 Source IP: 69.10.132.121 on ports: 6881, 6882, 6883, 6884, 6886 and 6889.

Additionally, there were some UDP TFTP alerts, however there were far fewer than the TCP connections. There were only 7 Internal to External and External to Internal UDP TFTP connections.

Examples:

```
12/19-13:24:56.035427  [**] TFTP - Internal UDP connection to external tftp server [**] 66.203.121.99:69 -
> 130.85.97.180:49452
12/21-05:59:10.245959  [**] TFTP - External UDP connection to internal tftp server [**]
198.64.140.205:33865 -> 130.85.97.87:69
```

Correlations: Mario Ricci noted this event in his GCIA paper as did Michael Hotaling. Ricci did not find any anomalous activity and saw about 4000 alerts. Hotaling saw about 800,000 alerts and was between only 2 hosts.[9&36]

Scans on port 69: 130.85.82.109 did one port scan of 69.10.132.121.

```
$ grep ":69:" scansall | awk -F : '{print $6,$7,$8,$9}' | sort | uniq -c | sort -rn | head
      1 67.20.173.236 4970 130.85.5.92 69
      1 67.20.173.236 34112 130.85.5.92 69
      1 67.20.173.236 34086 130.85.5.92 69
      1 62.118.129.10 41863 130.85.24.44 69
      1 62.118.129.10 37427 130.85.24.44 69
```

```
   1 204.1.226.228 62414 130.85.97.31 69
   1 130.85.82.109 1284 69.10.132.121 69
   1 130.85.60.16 43902 66.160.63.18 69
   1 130.85.60.16 40554 66.160.63.18 69
   1 130.85.6.49 256 12.4.221.77 69
```

2. **EXPLOIT x86 NOOP**	Severity: Medium	Reported: 4713
False Positive: Medium, need to see the whole rule.	Snort Signature ID: None (close matches: 1390, 1394)	Noise: 2

Example Alerts:

```
12/19-00:00:30.684350  [**] EXPLOIT x86 NOOP [**] 217.81.206.31:3115 -> 130.85.190.102:135
12/19-00:16:44.911999  [**] EXPLOIT x86 NOOP [**] 207.46.131.229:80 -> 130.85.150.85:1281
12/19-01:46:06.155532  [**] EXPLOIT x86 NOOP [**] 67.66.105.188:2252 -> 130.85.190.102:135
12/19-01:46:08.260763  [**] EXPLOIT x86 NOOP [**] 67.66.105.188:2932 -> 130.85.190.102:135
12/19-01:46:08.765881  [**] EXPLOIT x86 NOOP [**] 67.66.105.188:2933 -> 130.85.190.102:135
```

Sample Rule SID 1394: alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content:"|61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61|"; classtype:shellcode-detect; sid:1394; rev:4;)

This specific type of event is customarily generated when a possible attempt is made to overflow a buffer in a precursor to compromising a system. The NOOP warning occurs when a series of NOOP (no operation) are found in a data stream. Most buffer overflow exploits typically use NOOPs sleds to pad the code. However, as I do not have the exact signature being utilized for this specific alert which means that we cannot be completely certain of what this rule seeks.

IP addresses indicating the most activity with this group of alerts are:

```
$ grep "EXPLOIT x86 NOOP" alert6.mrg | awk -F : '{print $5,$8}' | sort | uniq -c | sort -rn | head
   1696  210.183.217.72 80
   1354  81.86.86.87 80
    161  131.118.254.130 119
    156  68.17.190.66 80
     96  218.148.120.180 80
     71  65.203.33.194 135
     52  62.234.24.4 80
     45  66.40.9.130 2290
     43  218.238.196.58 80
     42  81.249.167.184 80
```

Most of the activity appears to be related to web servers on port 80.

When: The NOOP were tracked for a period of five consecutive days, 19 December through 23 Decmeber 2003. On 19 December 2003 the alerts started at 0000 and continued until 2221 for a total of 126 alerts. On 20 December 2003, the alerts started at 0012 and continued until 2346 for a total of 119 alerts. On 21 December 2003 the alerts started at 0032 and continued until 2347 for a total of 425 alerts. On 22 December 2003 the alerts started at 0008 and continued until 2333, for a total of 2203 alerts. There was unusual activity between 0845-0846 when 210.183.217.72 had 267 alerts in one minute to 130.85.31.7 to port 80. Once again, there was an unusual amount of activity at 1044-1045 for 94 alerts from the same address to another address 130.85.162.175. This pattern continued with other IP addresses in the 130.85.0.0 subnet range. The address 210.183.217.72 had a total of 1696 NOOP alerts between 0205 22 December and 0024 23 December 2003, which lead into the logging of alerts tracked for 23 December. On 23 December 2003 the alerts started at 0001 and continued until 2345. There is a different, and very active, IP address, 81.86.86.87, on this date. The periods of peak activity varied throughout the course of the day, for a total of 1354 NOOP alerts between the hours of 0210 and 2345.

Dshield results: There was no previous malicious behavior tracked from these IP addresses.

MyNetWatchman: No signs of previous malicious behavior from these IP addresses had been noted.

Correlations: Marshall Heilman and Mark Embrich both found these alerts in their papers.[37&38]

3. **SMB Name Wildcard**	Severity: Low	Reported: 4343
False Positive: High. Very general rule to a port.	Snort SID: None	Noise: 5

```
12/19-00:19:01.939892  [**] SMB Name Wildcard [**] 130.85.75.13:137 -> 218.88.131.75:137
12/19-00:19:04.928774  [**] SMB Name Wildcard [**] 130.85.75.13:137 -> 218.88.131.75:137
```

Sample rule: alert udp $HOME_NET any -> $EXTERNAL_NET 137 (msg: “SMB Name Wildcard”;)

SMB is used during normal Name Resolution by NetBIOS. NetBIOS over TCP/IP (NBT) uses UDP 137 and 138 and TCP 139, and SMB (Server Message Block) is the protocol that runs over NBT. The protocol is used to connect to servers and workstations for drive mappings and sending commands.

When: The SMB alerts occurred in a chronological manner commencing 19 December 2003 and finalizing 23 December 2003. Specifically, on 19 December 2003 alerts started at 0008 and continued until 2348, for a total of 1060 alerts. On 20 December 2003 alerts started at 0003 and continued until 2347, for a total of 764 alerts. On 21 December 2003 alerts started at 0003 and continued until 2336, for a total of 766 alerts. On 22 December 2003 alerts started at 0000 and continued until 2346, for a total of 949 alerts. On 23 December 2003 alerts started at 0003 and continued until 2335, for a total of 805 alerts. There was no real discernable pattern for each day’s alerts which occurred sporadically throughout the day.

The most active IP addresses for this alert are as follows:

$ grep "SMB Name Wildcard" alert6.mrg | awk -F : '{print $5}' | sort | uniq -c | sort -rn | head 1845 130.85.11.6	$ grep "SMB Name Wildcard" alert6.mrg | awk -F : '{print $5,$7}' | sort | uniq -c | sort -rn | head 1845 130.85.11.6 169.254.0.0

414 130.85.75.13	215 130.85.11.7 169.254.0.0
284 130.85.150.198	59 130.85.84.155 218.145.28.100
268 130.85.150.44	25 130.85.190.102 66.98.154.21
227 130.85.11.7	24 130.85.162.108 169.254.0.0
112 130.85.190.102	22 130.85.190.102 66.98.212.28
59 130.85.84.155	22 130.85.112.153 169.254.45.176
24 130.85.162.108	21 130.85.189.17 169.254.45.176
22 130.85.190.95	20 130.85.75.13 216.74.144.15
22 130.85.112.153	20 130.85.112.179 169.254.45.176

All traffic relating to this alert originated internally on the 130.85.0.0 subnet and was destined externally.

Dshield results: There was no previous malicious behavior from these IP addresses.

MyNetWatchman results:
Incident ID: 69041693 Source IP: 66.98.212.28 on ports: 5666, 6050 and 6668.
Incident ID: 70344065 Source IP: 216.74.144.15 on ports: 25 and 53.

Correlations: In the scans data, there were many scans on port 137 across the 130.85.0.0 sub net range. Ian Martin had these alerts in his paper.[39]

Scans on port 137:

```
$ grep ":137:" scansall | awk -F : '{print $6,$7,$8,$9}' | sort | uniq -c | sort -rn | head
      3 61.241.226.74 137 130.85.190.95 137
      2 61.241.226.74 137 130.85.190.102 137
      2 130.85.84.194 137 62.251.222.210 137
      1 82.65.45.231 1076 130.85.190.254 137
      1 82.65.45.231 1076 130.85.190.253 137
      1 82.65.45.231 1076 130.85.190.252 137
      1 82.65.45.231 1076 130.85.190.251 137
      1 82.65.45.231 1076 130.85.190.250 137
      1 82.65.45.231 1076 130.85.190.249 137
      1 82.65.45.231 1076 130.85.190.248 137
```

4. **Connect to 515 from inside**	Severity: Low	Reported: 3557
False positive: High. General rule to a port.	Snort SID: None	Noise: 3

```
12/19-00:00:03.325006  [**] connect to 515 from inside [**] 130.85.162.41:721 -> 128.183.110.242:515
12/19-00:03:22.625038  [**] connect to 515 from inside [**] 130.85.162.41:721 -> 128.183.110.242:515
```

Sample Rule: alert tcp $HOME_NET any -> any 515 (msg: “Connect to 515 from inside”;)

Most active pairs of IP addresses are:

```
$ grep "connect to 515 from inside" alert6.mrg | awk -F : '{print $5,$6,$7,$8}' | sort | uniq -c |
sort -rn | head
   3545  130.85.162.41 721   128.183.110.242 515
      2  130.85.97.66 3163   192.168.0.14 515
      2  130.85.97.66 3162   192.168.0.14 515
      1  130.85.97.66 3165   192.168.0.14 515
      1  130.85.97.66 3164   192.168.0.14 515
```

```
     1  130.85.97.66 3160   192.168.0.14 515
     1  130.85.97.206 3098   192.168.2.1 515
     1  130.85.60.16 43627   66.160.63.18 515
     1  130.85.60.16 43612   66.160.63.18 515
     1  130.85.60.16 40844   66.160.63.18 515
```

The port involved with this alert refers to the lpd service running on TCP port 515. There are many vulnerabilities on that service listed within the CVE database.[5] These alerts do not seem to indicate a buffer overflow, merely a connection to port 515. All but eleven alerts came from one IP address, 130.85.162.41. These appear as legitimate connectivity from a source port of 721 to a destination port of 515. The others, however, are attempting to connect to a private IP range (192.168.0.0). The most likely scenario for this activity is that these are obfuscated IP addresses; however, if that is not the case, then the most probably assumption is that there is either a leaky firewall or a misconfigured client.

When: Alerts were tracked over the chronological course of five days, 19 December 2003 through 23 December 2003. Specifically, on 19 December 2003 alerts started at 0000 and continued until 2048 for a total of 3556 alerts. On the following dates, 20 December and 21 December 2003, there were no alerts logged. On 22 December 2003 there was a single alert at 1906. And, again, there were no alerts on 23 December 2003. With the sole exception of the alert on 22 December, all the logged alerts occurred on 19 December 2003

Dshield results: There was no previous malicious behavior from these IP addresses.

MyNetWatchman: No previous malicious behavior from these IP addresses was found.

Correlations: Tod Beardsley and Jasmir Beciragic discuss this alert in their papers. Beardsley conjectured that this port was now blocked at the border routers, apparently not from my results. Beciragic did not discuss this alert but it was referred to in his paper.[40&41]

Scans on port 515:

```
$ grep ":515:" scansall | awk -F : '{print $6,$7,$8,$9}' | sort | uniq -c | sort -rn | head
     1 130.85.60.16 43627 66.160.63.18 515
     1 130.85.60.16 40844 66.160.63.18 515
```

5. **High port 65535 udp - possible Red Worm – traffic**	Severity: Low	Reported: 3242
False Positive: High. General rule to a port.	Snort SID: None	Noise: 3

```
12/19-00:18:32.372769  [**] High port 65535 udp - possible Red Worm - traffic [**] 130.85.163.76:6257 ->
218.102.85.203:65535
12/19-00:45:08.307840  [**] High port 65535 udp - possible Red Worm - traffic [**] 210.153.125.57:65535
-> 130.85.163.76:6257
```

Sample Rule: alert udp any any <> any 65535 (msg: “High port 65535 udp - possible Red Worm – traffic”;)

Most active pairs of IP addresses:

```
$ grep "High port 65535 udp - possible Red Worm - traffic" alert6.mrg | awk -F : '{print $5,$6,$7,$
8}' | sort | uniq -c | sort -rn | head
    658  219.48.176.27 65535   130.85.163.76 6257
    227  204.116.162.109 65535   130.85.163.76 6257
    227  130.85.163.76 6257   204.116.162.109 65535
    194  219.213.15.15 65535   130.85.163.76 6257
    170  130.85.163.76 6257   219.213.15.15 65535
    151  130.85.163.76 6257   219.48.176.27 65535
    113  221.188.74.200 65535   130.85.163.76 6257
    111  130.85.163.76 6257   221.188.74.200 65535
    101  219.39.246.40 65535   130.85.163.76 6257
    101  130.85.163.76 6257   219.39.246.40 65535
```

“Adore is a worm that spreads in Linux systems using four different, known vulnerabilities already used by Ramen and Lion worms. These vulnerabilities concern BIND named, wu-ftpd, rpc.statd and lpd services… The backdoor activates when it receives a ping packet with correct size, and opens a shell in the port 65535.” The Adore worm should be using TCP not UDP. Red Worm is also known as Adore.[42]

130.85.163.76 was a source 1346 times, a destination 1744 times, and was reported a total of 3090 times as involved with this alert. Also of interest is the fact that the ports, 65535 and 6257 are repeated as either source or destination ports. Port 6257/udp is WinMX file sharing.[43] It would seem that 130.85.163.76 is involved with some type of file sharing.

When: The alerts occurred in a chronological manner commencing 19 December 2003 and finalizing on 23 December 2003. On 19 December 2003 alerts started at 0018 and continued until 2347 for a total of 1029 alerts. On 20 December 2003 alerts started at 0001 and continued until 2101 for a total of 139 alerts. On 21 December 2003 alerts started at 0002 and continued until 2341 for a total of 562 alerts. On 22 December 2003 the alerts started at 0006 and continued until 2339 for a total of 132 alerts. On 23 December 2003 the alerts started at 0001 and continued until 2336 for a total of 1380 alerts.

Dshield results: There is no previous malicious behavior from these IP addresses.

MyNetWatchman: No previous malicious behavior from these IP addresses has been found.

Correlations: Both Les Gordon and Scott Shinberg refer to these alerts in their papers. Gordon does not think this is Adore while others associated this behavior with AFS servers. Shinberg believed that University systems were potentially compromised.[11&44]

Scans for/from port 65535:

```
$ grep ":65535:" scansall | awk -F : '{print $6,$7,$8,$9,$10}' | sort | uniq -c | sort -rn | head
    156 130.85.163.76 6257 204.116.162.109 65535 UDP
     69 130.85.163.76 6257 219.39.246.40 65535 UDP
     48 130.85.163.76 6257 221.188.74.200 65535 UDP
     25 130.85.163.76 6257 219.29.250.37 65535 UDP
     23 130.85.163.76 6257 219.58.10.114 65535 UDP
      9 130.85.163.76 6257 219.164.150.138 65535 UDP
      6 130.85.25.73 65535 213.216.138.129 25 SYN
      5 130.85.82.104 3185 212.149.203.85 65535 UDP
      5 130.85.34.14 65535 209.88.229.110 25 SYN
      5 130.85.163.76 6257 218.219.102.24 65535 UDP
```

6. **ICMP SRC and DST outside network**	Severity: Low	Reported: 1713
False Positive: High. Very general rule.	Snort SID: None	Noise: 5

```
12/19-00:04:42.158743  [**] ICMP SRC and DST outside network [**] 172.166.97.125  -> 172.168.67.165
12/19-00:05:34.294971  [**] ICMP SRC and DST outside network [**] 172.166.97.125  -> 172.165.167.7
```

Sample Alert: alert icmp $EXTERNAL_NET any -> $EXTERNAL_NET any (msg: “ICMP SRC and DST outside network”;)

Most active pairs of IP addresses:

```
$ grep "ICMP SRC and DST outside network" alert6.mrg | awk -F : '{print $5 $6}' | sort | uniq -c | sort -rn |
head
     39  192.168.0.25  211.150.211.6
      7  172.174.114.142  67.105.78.198
      7  172.169.253.52  208.60.8.140
      6  172.169.253.52  67.105.78.198
      3  68.85.214.43  68.85.207.181
      3  68.85.214.43  68.85.201.53
      3  172.169.253.52  172.172.128.241
      3  172.169.246.212  67.105.78.198
      3  172.169.246.212  208.60.8.140
      3  172.169.246.212  172.170.254.237
```

This alert is simply telling that there is ICMP activity only on the external network. Other than a potentially spoofed IP (192.168.0.25) address this alert is un-eventful. None of the top 10 ip’s was from the Universities class B network.

When: The alerts were tracked in a chronological manner commencing 19 December 2003 and finalizing 23 December 2003. On 19 December the alerts started at 0004 and continued until 2251 for a total of 431 alerts. On 20 December 2003 the alerts started at 0014 and continued until 2343 for a total of 399 alerts. On 21 December 2003 the alerts started at 0028 and continued until 2328 for a total of 93 alerts. On 22 December 2003 the alerts started at 0003 and continued until 2346 for a total of 226 alerts. On 23 December 2003 the started start at 0001 and continued until 2314 for a total of 564 alerts.

Dshield results: There was no previous malicious behavior from these IP addresses.

MyNetWatchman: Incident ID: 41288952 for IP: 67.105.78.198. Incident ID: 58979661 for IP: 208.60.8.140. Incident ID: 41288952 for IP 68.85.207.181. Incident

ID: 65602670 for IP: 68.85.201.53. All target acquisition type behavior or ICMP Echo Request Possible Nachi/Welchia Infection.

Correlations: Scott Shinberg discussed this type of activity in his paper. However, Shinberg could not draw any conclusions from this behavior.[44] It appears likely that the alerts from the 192.168.0.0/16 subnet are from a misconfigured firewall. MyNetWatchman found 4 of the source IP addesses to be involved with some type of malicious ICMP behavior.

7. **NMAP TCP ping!**	Severity: Low	Reported: 1696
False positive: High. General rule.	Snort SID: Closest would be 628, Scan nmap TCP	Noise: 5

```
12/19-00:14:57.676803  [**] NMAP TCP ping! [**] 61.30.119.193:80 -> 130.85.1.4:53
12/19-01:03:48.573460  [**] NMAP TCP ping! [**] 209.109.246.253:80 -> 130.85.12.6:25
```

Sample Rule: alert tcp any any -> $HOME_NET any (flags: A; ack: 0; msg: "NMAP TCP ping!";)

Most active pairs of IP addresses:

```
$ grep "NMAP TCP ping" alert6.mrg | awk -F : '{print $5,$7}' | sort | uniq -c | sort -rn | head
   1081  67.20.173.236  130.85.5.92
     56  205.244.232.133  130.85.12.4
     51  216.5.176.162  130.85.12.4
     28  63.211.17.228  130.85.1.3
     24  64.152.70.68  130.85.1.3
     13  81.255.44.14  130.85.100.165
     10  194.98.100.14  130.85.100.165
      8  213.31.226.162  130.85.100.165
      8  213.223.49.226  130.85.100.165
      8  211.21.74.30  130.85.185.13
```

Nmap is a well known port scanning utility. Snort has many built in signatures to recognize Nmap scanning. There is not an exact match for this particular alert; however, the signature of the alert I did find was Scan Nmap TCP.[45] This particular alert looks for an ACK number of 0. This alert occurred 1081 times for the IP address 67.20.173.236 scanning 130.85.5.92. Upon closer examination, 67.20.173.236 was only active on 19 December 2003 between the hours of 22:18:39.470989 and 22:44:03.927979 with the majority of the traffic occurring between 22:43:48.540373 and 22:44:03.927979. During that time there were 1079 packets sent on port 25 to 130.85.5.92 and then there were no more packets during the 5 day period. It appears as though the system at 67.20.173.236 was trying some type of attack on port 25. I cannot determine whether the attack was successful from the alert traffic.

```
$ grep "67.20.173.236" alert6.mrg | awk -F : '{print $4,$5,$7}' | sort | uniq -c | sort -rn | head
   1081  NMAP TCP ping!   67.20.173.236  130.85.5.92
      4  TFTP - External TCP connection to internal tftp server   67.20.173.236  130.85.5.92
      4  TFTP - External TCP connection to internal tftp server   130.85.5.92  67.20.173.236
      4  Null scan!   67.20.173.236  130.85.5.92
      3  Attempted Sun RPC high port access   67.20.173.236  130.85.5.92
```

```
    2  Incomplete Packet Fragments Discarded   67.20.173.236
    1  TCP SMTP Source Port traffic   67.20.173.236  130.85.5.92
    1  Probable NMAP fingerprint attempt   67.20.173.236  130.85.5.92
    1  Possible wu-ftpd exploit - GIAC000623   67.20.173.236  130.85.5.92
    1  PHF attempt   67.20.173.236  130.85.5.92
```

Because of the unusually high Nmap activity I decided to see who the 67.20.173.236 address was registered to.

Hostname: md-wmnsmd-cuda1-c6c-236.chvlva.adelphia.net

CustName: Adelphia Address: 1 North Main Street City: Coudersport StateProv: PA PostalCode: 16915 Country: US RegDate: 2003-06-23 Updated: 2003-06-23	NetRange: 67.20.160.0 - 67.20.191.255 CIDR: 67.20.160.0/19 NetName: 67201600-Z5 NetHandle: NET-67-20-160-0-1 Parent: NET-67-20-0-0-1 NetType: Reassigned Comment: RegDate: 2003-06-23 Updated: 2003-06-23

When: Nmap alerts occurred in a chronological manner commencing 19 December 2003 and continuing through 23 December 2003. On 19 December 2003 the alerts started at 0014 and continued until 2334 for a total of 1235 alerts. Additionally, at 2243 on 19 December, 67.20.173.236 began its port scan of 130.85.5.92. At 2244 it had finished the port scan of port 25. In approximately one minute it did 1081 Nmap tcp pings. Following this, on 20 December 2003 the alerts started at 0005 and continued until 2324 for a total of 76 alerts. On 21 December 2003 the alerts started at 0005 and continued until 2345 for a total of 96 alerts. On 22 December 2003 the alerts started at 0009 and continued until 2316 for a total of 155 alerts. On 23 December 2003 the alerts started at 0014 and continued until 2126 for a total of 135 alerts.

Dshield results:
Showed activity for 205.244.232.133 on ports, 53, 80, 35647 and 37852 (Linkproof, Radware);[43]
Showed activity for 216.5.176.162 on ports: 53, 35647 and 37852.
Showed activity for 63.211.17.228 on ports: 37852,53,0, 1088, 34433, 1174, 1069, 3702, 4314 and 9772.
Showed activity for 64.152.70.68 on ports: 37852, 53, 0, 4950, 4535, 34432, 47375, 45658, 47798, 41997.

MyNetWatchman:
Incident ID: 70333364 for IP: 205.244.232.133 on ports: 25, 2094, 6346, 6348 and 37852.
Incident ID: 58186150 for IP: 213.31.226.162 on ports: 25, 53, 80 3789, 37852 and 46497

Correlations:Tod Beardsley found this exact alert.[40] DShield and MyNetWatchman both indicated a few of the IP addresses were involved with suspicious activity. Much of the suspicious activity has something to do with Radware's product LinkProof, a product used to manage multihomed internet connections.[46]

8. **High port 65535 tcp - possible Red Worm – traffic**	Severity: Low	Reported: 1086
False Positive: High. General rule to a port.	Snort SID: None	Noise: 2

```
12/19-00:03:12.286880  [**] High port 65535 tcp - possible Red Worm - traffic [**] 67.9.68.185:65535 ->
130.85.97.139:2626
12/19-00:16:38.980405  [**] High port 65535 tcp - possible Red Worm - traffic [**] 130.85.97.139:3179 ->
67.9.68.185:65535
```

Sample Rule: alert tcp any any <> any 65535 (msg: "High port 65535 tcp - possible Red Worm – traffic";)

Please see 5. above. There does not seem to be any correlation between the TCP and UDP traffic. This alert appears to trigger on the type of traffic and the source port (TCP and port 65535).

Most active pairs of IP addresses:

```
 $ grep "High port 65535 tcp - possible Red Worm - traffic" alert6.mrg | awk -F : '{print $5,$6,$7,$
8}'| sort | uniq -c | sort -rn | head
     15  130.85.25.70 65535   64.157.4.79 25
     14  65.54.252.230 25   130.85.25.70 65535
     14  130.85.6.7 80   151.203.195.219 65535
     14  130.85.12.6 25   128.220.2.67 65535
     13  130.85.25.70 65535   194.26.184.18 25
     13  130.85.12.4 143   68.55.66.114 65535
     12  68.109.149.53 65535   130.85.97.26 1297
     12  204.127.202.61 65535   130.85.24.20 25
     12  194.26.184.18 25   130.85.25.70 65535
     11  68.55.176.240 65535   130.85.24.74 443
```

It is possible with heavy traffic to have many naturally occurring hits to TCP port 65535. The host 130.85.25.70 appears more frequently than any other host for this alert. All alerts involving 130.85.25.70 revolve around ports 65535 and 25. The conclusion that 130.85.25.70 is some type of mail server is possible.

68.109.149.53 seemed a bit out of character for most of the alerts, so I looked at that IP separately. It is evident that 68.109.149.53 connects to 130.85.97.26 from a high port, and there is some traffic communicated on these ports. While this is unusual, the information is inconclusive.

```
$ grep "68.109.149.53" alert6.mrg | awk -F : '{print $5,$6,$7,$8}' | sort | uniq -c | sort -rn
     12  68.109.149.53 65535   130.85.97.26 1297
      7  130.85.97.26 1297   68.109.149.53 65535
      5  68.109.149.53 65535   130.85.97.26 1201
      4  130.85.97.26 1201   68.109.149.53 65535
```

When: The alerts were tracked for the five days commencing 19 December 2003 through and including 23 December 2003. On 19 December 2003 the alerts started at 0003 and continued until 2156 for a total of 260 alerts. On 20 December 2003 the alerts started at 0129 and continued until 2125 for a total of 160 alerts. On 21 December 2003 the alerts started at 0000 and continued until 2304 for a total of 256 alerts. On 22 December 2003 the alerts started at 0017 and continued until 2345 for a total of 239 alerts. On 23 December 2003 the alerts started at 0159 and continued until 2330 for a total of 171 alerts.

Dshield results:
There was no previous malicious behavior from these IP addresses.

MyNetWatchman:
Incident ID: 68912504 Source IP: 64.157.4.79 for many destination ports all from port 25.
Incident ID: 70963022 Source IP: 68.109.149.53 from port 2270 and 45069
Incident ID: 65144514 Source IP: 204.127.202.61 from port 25 to 58213

Correlations: Les Gordon found one host in his analysis that warranted further examination.[11]

Scan for port 65535: See 5. above.

9. **Null scan!**	Severity: Low	Reported: 662
False Positive: High. General rule.	Snort SID: None, closest match 623 (Scan Null)	Noise: 4

```
12/19-00:37:43.286008  [**] Null scan! [**] 68.122.128.111:40452 -> 130.85.12.4:110
12/19-00:39:30.622747  [**] Null scan! [**] 24.236.113.185:0 -> 130.85.97.62:0
```

Sample Rule: alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN NULL"; stateless; flags:0; seq:0; ack:0; reference:arachnids,4; classtype:attempted-recon; sid:623; rev:2;)

Most active pairs of IP addresses:

```
$ grep "Null scan" alert6.mrg | awk -F : '{print $5,$6,$7,$8}'| sort | uniq -c | sort -rn | head
    106  63.251.52.75 0   130.85.53.196 0
     95  195.208.34.220 0   130.85.12.6 0
     38  218.189.230.43 0   130.85.12.6 0
     15  63.251.52.75 0   130.85.82.112 0
     12  61.194.13.120 0   130.85.97.89 0
     12  212.85.224.66 33970   130.85.185.13 4662
     10  63.251.52.75 0   130.85.97.10 0
      9  63.251.52.75 0   130.85.70.197 0
      7  63.251.52.75 0   130.85.97.215 0
      6  81.218.84.118 21137   130.85.185.13 4662
```

From the Snort website documentation: "Summary A tcp packet with none of it's control bits set was detected. Information regarding firewall rule sets, open/closed ports, ACLs, and possibly even OS type is possible. This technique can also be used to bypass certain firewalls or traffic filtering/shaping devices.

Detailed Information A tcp packet with none of it's control bits (URG, ACK, PSH, RST, SYN, FIN) was detected. Additionally, both the sequence number and acknowledgement number were set to 0. An open port will generally not respond at all, whereas a closed port will generally respond with an ACK RST. The particular response varies between operating systems, and is also governed by any filtering that may be done between the two hosts. Attack Scenarios, as part of information gathering leading up to another (more directed) attack, an attacker may attempt to figure out what ports are open/closed on a remote machine.”[47]

When: Alerts were tracked on five chronologically consecutive days, from 19 December 2003 through and including 23 December 2003. The alerts started on 19 December 2003 at 0037 and continued until 2338 for a total of 205 alerts. On 20 December 2003 the alerts started at 0000 and continued until 2238 for a total of 68 alerts. On 21 December 2003 the alerts started at 0028 and continued until 2244 for a total of 92 alerts. On 22 December 2003 the alerts started at 0034 and continued until 2311 for a total of 105 alerts. On 23 December 2003 the alerts started at 0145 and continued until 2317 for a total of 192 alerts.

Dshield results: There was no previous malicious behavior from these IP addresses.

MyNetWatchman:
Incident ID: 12534621 Source IP: 63.251.52.75 for many different ports all from port 80.
Incident ID: 71136818 Source IP: 212.85.224.66 for port: 6348.

Correlations: Robert Sorensen found, in his paper, that these were indicative of crafted packets.[48] I tend to agree with this statement.

Scans for port 0: There were 397 Null Scans reported in the top 10 scanning results below. Of the 397, 382 were from a single IP source/destination pair and were not from our alert results above.

```
$ grep ":0:" scansall | awk -F : '{print $6,$7,$8,$9}' | sort | uniq -c | sort -rn | head
    382 130.85.6.49 0 12.4.221.77 0
```

10. **Possible trojan server activity**	Severity: Medium	Reported: 327
False Positive: Medium. General rule to a port.	Snort SID: None	Noise: 3

```
12/19-07:43:30.743570  [**] Possible trojan server activity [**] 130.85.60.38:25 -> 211.157.252.69:27374
12/19-07:43:31.528434  [**] Possible trojan server activity [**] 211.157.252.69:27374 -> 130.85.60.38:25
```

The rule appears to be set to trigger an alert when there is any traffic to or from port 27374. Port 27374 is a well established port used by many malicious Trojans.

Sample Rule: Alert tcp any any <> any 27374 (msg; “Possible Trojan server activity”;)

Most active pairs of IP addresses:

```
$ grep "Possible trojan server activity" alert6.mrg | awk -F : '{print $5,$6,$7,$8}'| sort| uniq –c | sort -rn |
head
     33  64.68.82.28 27374   130.85.24.34 80
     26  130.85.24.34 80   64.68.82.28 27374
     24  130.85.24.34 80   12.5.169.91 27374
     15  66.236.191.164 27374   130.85.97.92 6413
     12  24.35.0.138 27374   130.85.29.3 80
     11  130.85.29.3 80   24.35.0.138 27374
     10  209.165.168.2 27374   130.85.5.20 80
     10  12.5.169.91 27374   130.85.24.34 80
      9  146.82.220.34 27374   130.85.12.6 25
      9  130.85.29.3 80   140.185.28.43 27374
```

When you look at the IP addresses of the potential ‘trojaned’ hosts you see only one host that potentially might be infected, 130.85.97.92:

It is possible with heavy traffic to have this many naturally occurring hits to TCP port 27374. There were a few portscan records directed toward 130.85.97.92. None, however, from 66.236.191.164 to port 27374

```
$ grep "130.85.97.92" scansall | awk -F : '{print $6,$8,$9}' | sort | uniq -c | sort -rn | head
      4 164.64.80.23 130.85.97.92 21
      3 136.165.63.200 130.85.97.92 6129
      2 80.55.127.226 130.85.97.92 80
      2 67.68.147.94 130.85.97.92 6129
      2 62.251.222.210 130.85.97.92 21
      2 213.208.112.174 130.85.97.92 6129
      2 213.130.59.14 130.85.97.92 2277
      2 212.12.161.94 130.85.97.92 6129
      2 209.208.0.15 130.85.97.92 57123
      2 209.208.0.15 130.85.97.92 53311
```

6129 is Dameware Remote Admin service.[43]

Upon further examination, into the alerts it becomes apparent that there is indeed some suspicious activity.

```
$ grep "130.85.97.92" alert6.mrg | awk -F : '{print $4,$5,$6,$7,$8}'| sort| uniq -c | sort -rn | head
     15  Possible trojan server activity   66.236.191.164 27374   130.85.97.92 6413
      5  Possible trojan server activity   130.85.97.92 6413   66.236.191.164 27374
      1  [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.   66.205.212.39 6667
130.85.97.92 1366
```

There is symmetrical communication on port 27374 and 6413. Additionally, there is an alert on 6667 for IRC. IRC is often used by hackers to communicate that a system has been compromised.

When: Alerts were tracked on five consecutive dates in December 2003, specifically the 19th through the 23rd. On 29 December 2003 the alerts started at 0743 and continued until 1607 for a total of 64 alerts. On 20 December 2003 the alerts started at 0220 and continued until 1919 for a total of 29 alerts. On 21 December 2003 the alerts started at 0108 and continued until 2330 for a total of 51 alerts. On 22 December 2003 the alerts started at 0737 and continued until 2201 for a total of 148 alerts. Also on 22 December, the activity for 66.236.191.164 and 130.85.97.92 occurred between 0737 and 0746. On 23 December 2003 the alerts started at 0004 and continued until 1605 for a total of 35 alerts.

Dshield results: Showed results for 66.236.191.164 on port 53.

MyNetWatchman: There was no previous malicious behavior from these IP addresses.

Correlations: Scott Shinberg found this activity in his GCIA research. It seems reasonable to agree with his statement that a high port connecting to 27374 is far more suspect than a low server type port (< 1024). Tod Beardsley, in his work, found that a few hosts were compromised. Additionally, Les Gordon found this rule did not produce too many false positives.[44&40]

Scans for port 27374:

```
$ grep ":27374:" scansall | awk -F : '{print $6,$7,$8,$9,$10}' | sort | uniq -c | sort -rn | head
     1 213.212.142.252 27374 130.85.7.237 80 SYN
     1 130.85.60.16 42227 66.160.63.18 27374 SYN
```

11. 130.85.30.3 and 130.85.30.4 and Incomplete Packet Fragments Discarded

Table 4: Highest number of alerts

Alert Count	Alert Message:
23810	130.85.30.3 activity
21855	130.85.30.4 activity
13669	Incomplete Packet Fragments Discarded

130.85.30.3 and 130.85.30.4 activity

These alerts were among the most prevalent found during the five day period. They are analyzed separately because the sheer volume of alerts appeared to indicate significant interest on the part of the University, and, it was, in fact their volume that leads me to speculate the University was watching these IP's for a reason.

The alerts from 130.85.30.3 and 130.85.30.4 notify that there is activity from these IP addresses, however, there is no way of telling what they are from the description. These 3 alert messages were in the top 3 of total alert counts, but were removed since they do not tell us anything of great importance.

Sample Rule: alert ip any -> 130.85.30.3 any (msg: “130.85.30.3 activity”;)

False Positive: Unknown. It depends on what they are looking for. It seems as if the University wants to know whether there is activity to 130.85.30.3. This rule allows for that information. It is, however, extremely general, and will give notification regarding all activity to the IP address in question

Noise: 5.

We see that from the top 10 list of most active connections to 130.85.30.3 that all the connections are to port 524 (NCP).

```
$ grep "130.85.30.3 activity" alert6.mrg | awk -F : '{print $5,$7,$8}'| sort | uniq -c | sort -rn | head
  11059  68.50.114.89  130.85.30.3 524
   2399  68.57.90.146  130.85.30.3 524
   1610  68.55.113.194  130.85.30.3 524
   1488  68.55.62.79  130.85.30.3 524
    757  66.168.239.240  130.85.30.3 524
    718  68.32.122.89  130.85.30.3 524
    534  131.92.177.18  130.85.30.3 524
    531  68.55.27.157  130.85.30.3 524
    380  151.196.10.167  130.85.30.3 524
    302  68.81.2.19  130.85.30.3 524
```

When we look at 130.85.30.4 we see that port 51443, 524 and 80 are involved.

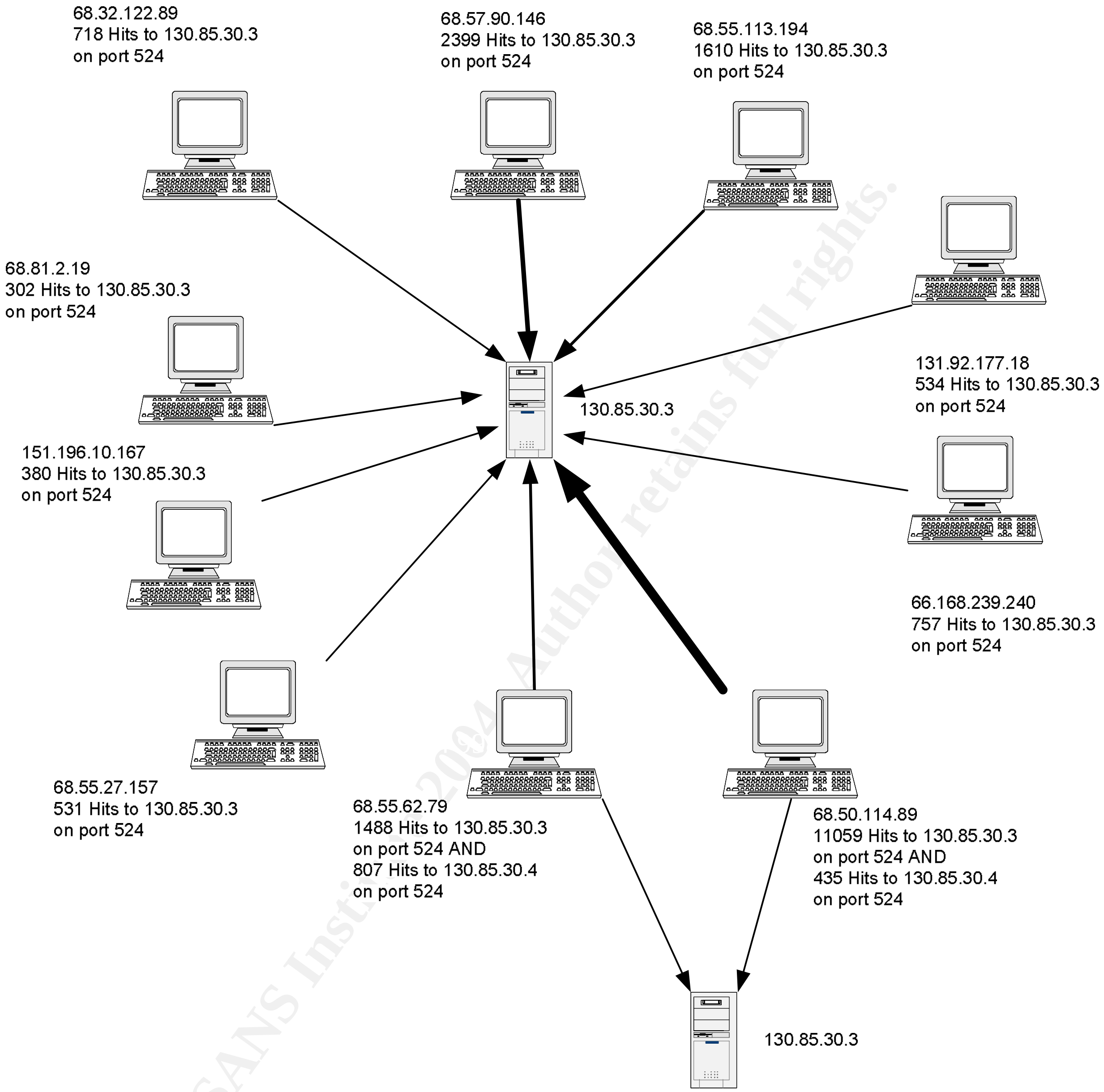
```
grep "130.85.30.4 activity" alert6.mrg | awk -F : '{print $5,$7,$8}'| sort | uniq -c | sort -rn | head
  4848  68.55.241.230  130.85.30.4 51443
  3183  66.68.62.250  130.85.30.4 51443
  2259  151.196.239.212  130.85.30.4 51443
   807  68.55.62.79  130.85.30.4 524
   552  67.20.160.15  130.85.30.4 51443
   490  151.196.116.233  130.85.30.4 51443
   435  68.50.114.89  130.85.30.4 524
   330  66.196.72.58  130.85.30.4 80
   320  66.196.65.37  130.85.30.4 80
   306  66.196.72.46  130.85.30.4 80
```

Table 5 Registration information for 4 of the external sources causing alerts to 130.85.30.3. 68.32.0.0 - 68.63.255.255 is registered to Comcast and was therefore only examined once since 5 of the top 10 were from Comcast:

68.50.114.89 = pcp04615078pcs.gambrl01.md.comcast.net 68.55.241.230, 68.57.90.146, 68.55.113.194 68.55.62.79 (all registered to Comcast) CustName: Comcast Cable Communications, Inc. Address: 3 Executive Campus Address: 5th Floor City: Cherry Hill StateProv: NJ PostalCode: 08002 Country: US RegDate: 2003-03-19 Updated: 2003-03-19	NetRange: 68.50.0.0 - 68.50.255.255 CIDR: 68.50.0.0/16 NetName: DC-4 NetHandle: NET-68-50-0-0-1 Parent: NET-68-32-0-0-1 NetType: Reassigned Comment: NONE RegDate: 2003-03-19 Updated: 2003-03-19
66.168.239.240 = 66.168.239.240	NetRange: 66.168.224.0 - 66.168.239.255

OrgName: Charter Communications OrgID: CC04 Address: 12405 Powerscourt Dr. City: St. Louis StateProv: MO PostalCode: 63131 Country: US	CIDR: 66.168.224.0/20 NetName: HOVR-AL-66-168-224 NetHandle: NET-66-168-224-0-1 Parent: NET-66-168-0-0-1 NetType: Reallocated Comment: RegDate: 2001-11-20 Updated: 2003-08-27
131.92.177.18 = aeclt-cfdoa4.apgea.army.mil OrgName: Army Information Systems Command - Aberdeen (EA) OrgID: AISCAE Address: AMSSB-SCI-N/BLDG E5234 City: ABERDEEN PROVING GROUND StateProv: MD PostalCode: Country: US	NetRange: 131.92.0.0 - 131.92.255.255 CIDR: 131.92.0.0/16 NetName: APGEA-NET1 NetHandle: NET-131-92-0-0-1 Parent: NET-131-0-0-0-0 NetType: Direct Assignment NameServer: NS01.ARMY.MIL NameServer: NS02.ARMY.MIL NameServer: NS03.ARMY.MIL Comment: RegDate: 1988-11-01 Updated: 2001-08-09
151.196.10.167 = pool-151-196-10-167.balt.east.verizon.net CustName: Verizon Internet Services Address: 1880 Campus Commons Drive City: Reston StateProv: VA PostalCode: 20191 Country: US RegDate: 2002-03-21 Updated: 2002-03-21	NetRange: 151.196.5.0 - 151.196.29.255 CIDR: 151.196.5.0/24, 151.196.6.0/23, 151.196.8.0/21, 151.196.16.0/21, 151.196.24.0/22, 151.196.28.0/23 NetName: VZ-DSLDIAL-CYVLMD-1 NetHandle: NET-151-196-5-0-1 Parent: NET-151-196-0-0-1 NetType: Reassigned Comment: RegDate: 2002-03-21 Updated: 2002-03-21

Figure 1 Link Diagram of top 10 source IP address connections to 130.85.30.3:



When: Again the alerts were tracked for the five consecutive days, 19 December 2003 through 23 December 2003. The alerts for 130.85.30.3 started on19 December 2003 at 0003 and continued until 2343 for a total of 6098 alerts. On 20 December 2003 the alerts started at 0000 and continued until 2347 for a total of 8816 alerts. On 21 December 2003 the alerts started at 0003 and continued until 2320 for a total of 3172 alerts. On 22 December 2003 the alerts started at 0003 and continued until 2332 for a total of 4230 alerts. On 23 December 2003 the alerts started at 0004 and continued until 2333 for a total of 1495 alerts.

The alerts for 130.85.30.4 started on 19 December 2003 at 0001 and continued until 2346 for a total of 2836 alerts. On 20 December 2003 the alerts started at 0000 and continued until 2346 for a total of 2426 alerts. On 21 December 2003 the alerts started at 0000 and continued until 2346 for a total of 6858 alerts. On 22 December 2003 the alerts started at 0000 and continued until 2337 for a total of 7629 alerts. On 23 December 2003 the alerts started at 0005 and continued until 2347 for a total of 2107 alerts.

<u>*Dshield Results*</u>: There was no previous malicious behavior from these IP addresses

<u>*MyNetWatchman*</u>: No previous malicious behavior from these IP addresses has been found.

<u>*Correlation*</u>: Ian Martin thought that these servers were perhaps honeypots, which explanation is quite plausible. Covert communications were also discussed by Ian Martin.[39]

<u>*Scans*</u>: None of the hosts involved (i.e., not on the 130.85.0.0/16 network) were involved as sources of scans.

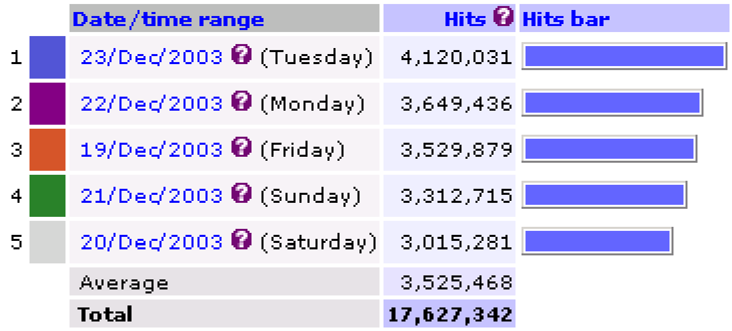
Incomplete Packet Fragments Discarded Analysis:

Apparently these are caused by the old defrag processor on older snort systems. According to M. Roesch there were some issues with the older versions. [49]

<u>*Correlations*</u>: Ian Martin found these to be related to broken TCP or transmission errors., Les Gordon found this alert and determined that it related to internet gaming. [39&11]

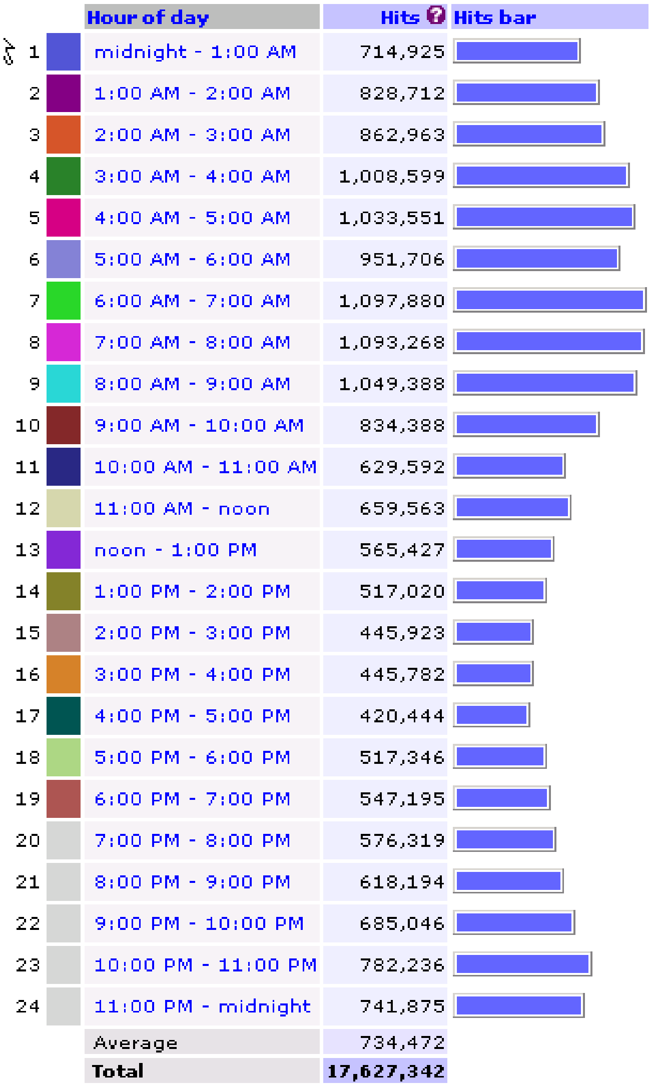
Frequent Scan Analysis:

Figure 2 Number of Scans per day: There were 17,627,342 scans with an average of 3,525,281 scans per day:



		Date/time range	Hits	Hits bar
1	🟦	23/Dec/2003 (Tuesday)	4,120,031	
2	🟪	22/Dec/2003 (Monday)	3,649,436	
3	🟧	19/Dec/2003 (Friday)	3,529,879	
4	🟩	21/Dec/2003 (Sunday)	3,312,715	
5	⬜	20/Dec/2003 (Saturday)	3,015,281	
		Average	3,525,468	
		Total	**17,627,342**	

Figure 3 Number of Scans per hour over the 5 day period: There was an average of 734, 472 scans per hour over the 5 day period:



		Hour of day	Hits	Hits bar
1	■	midnight - 1:00 AM	714,925	
2	■	1:00 AM - 2:00 AM	828,712	
3	■	2:00 AM - 3:00 AM	862,963	
4	■	3:00 AM - 4:00 AM	1,008,599	
5	■	4:00 AM - 5:00 AM	1,033,551	
6	■	5:00 AM - 6:00 AM	951,706	
7	■	6:00 AM - 7:00 AM	1,097,880	
8	■	7:00 AM - 8:00 AM	1,093,268	
9	■	8:00 AM - 9:00 AM	1,049,388	
10	■	9:00 AM - 10:00 AM	834,388	
11	■	10:00 AM - 11:00 AM	629,592	
12	■	11:00 AM - noon	659,563	
13	■	noon - 1:00 PM	565,427	
14	■	1:00 PM - 2:00 PM	517,020	
15	■	2:00 PM - 3:00 PM	445,923	
16	■	3:00 PM - 4:00 PM	445,782	
17	■	4:00 PM - 5:00 PM	420,444	
18	■	5:00 PM - 6:00 PM	517,346	
19	■	6:00 PM - 7:00 PM	547,195	
20	■	7:00 PM - 8:00 PM	576,319	
21	■	8:00 PM - 9:00 PM	618,194	
22	■	9:00 PM - 10:00 PM	685,046	
23	■	10:00 PM - 11:00 PM	782,236	
24	■	11:00 PM - midnight	741,875	
		Average	734,472	
		Total	**17,627,342**	

Top 10 Scanning Host analysis:

None of the Top 10 Scanning hosts were among the Top 10 Alerters.

Table 6:Top 2 of the Top 10 Destination Ports are 53 and 135.

```
$ awk -F : '{print $9,$10}' scansall | sort | uniq -c | sort -rn | head
10391598 135 SYN
3823611 53 UDP
 731246 6129 SYN
 298448 25 SYN
 232376 80 SYN
 166554 6257 UDP
 105324 21 SYN
  83751 4899 SYN
  49237 41170 UDP
  43872 113 SYN
```

Port 135/tcp, is discussed below and elsewhere. Port 53/udp discussed below. Port 6129/tcp is Dameware Remote Admin service. Port 25/tcp is SMTP. Port 80/tcp is HTTP. Port 6257/udp is WinMX file sharing. Port 21/tcp is FTP. Port 4899/tcp is Radmin. Port 41170/udp is Blubster file sharing utility port. Port 113/tcp is Ident, Auth service and some Trojans use this too.[43]

Table 7 lists the systems that were the top scanning hosts.

Table 7: Top scanning hosts (Bandwidth Hogs):

```
$ awk -F : '{print $6}' scansall| sort | uniq -c | sort -rn | head
1.   3299342 130.85.1.3
2.   2925492 130.85.111.72
3.   2394279 130.85.84.194
4.   2384899 130.85.162.92
5.   2379640 130.85.163.107
6.   547966 130.85.1.4
7.   385337 130.85.110.72
8.   324437 130.85.185.13
9.   305964 130.85.84.164
10.  278893 130.85.80.243
```

Analysis of top scanning hosts:

#1,130.85.1.3 and #6,130.85.1.4: Appear to be DNS Servers (Table 8). Port 53 is used for DNS communication. It is possible that there is some type of attack going on, on port 53, but it is more likely to be DNS traffic.

Table 8: 130.85.1.3 DNS port 53 scans.

```
$ grep "130.85.1.3" scansall | awk -F : '{print $6,$8,$9}' | sort |
uniq -c | sort -rn | head
  81006 130.85.1.3 69.6.61.11 53
  56298 130.85.1.3 192.26.92.30 53
  49027 130.85.1.3 203.20.52.5 53
  41952 130.85.1.3 204.29.185.132 53
  41797 130.85.1.3 69.6.61.10 53
  36177 130.85.1.3 128.194.254.4 53
  36126 130.85.1.3 128.194.254.5 53
  35415 130.85.1.3 192.5.6.30 53
  33893 130.85.1.3 69.6.25.84 53
  33865 130.85.1.3 69.6.25.125 53
```

#2, 3, 4, 5 and 10:130.85.111.72, 130.85.84.194, 130.85.162.92, 130.85.163.107 and 130.85.80.243 all appear to be infected with the Welchia/Nachi worm and are actively scanning the internet for more hosts to infect on port 135 (Table 9). Port 135 was shown to be cumulatively the most actively scanned port from the port scanning logs with 10,391,667 scans (Table 6).

Table 9: Port 135 analysis of Scan data.

```
$ grep ":135:" scansall | awk -F : '{print $6,$9}' | sort | uniq -c |
sort -rn | head
2925477 130.85.111.72 135
2392175 130.85.84.194 135
2384855 130.85.162.92 135
2379626 130.85.163.107 135
 278743 130.85.80.243 135
    450 217.84.138.60 135
    409 208.17.225.200 135
    375 220.197.192.39 135
    357 64.121.64.236 135
    320 68.174.23.102 135
```

#7, 130.85.110.72 appears to be an internet gaming server (Table 10). Specifically, it is a Medal of Honor game server. UDP ports 8767 = Team Speak (215311 UDP scans), ports 12203 and 12300 MOH (113519 and 55064 scans).[43]

Table 10: 130.85.110.72 port analysis, game server.

```
$ grep "130.85.110.72" scansall | awk -F : '{print $6,$7}' | sort |
uniq -c | sort -rn | head
 215311 130.85.110.72 8767
 113519 130.85.110.72 12203
  55064 130.85.110.72 12300
    589 130.85.110.72 32774
    515 130.85.110.72 32782
    128 130.85.110.72 32771
     94 130.85.110.72 0
      4 130.85.110.72 13
      3 130.85.110.72 10794
      2 80.55.127.226 3955
```

#8 130.85.185.13 appears to have some type of P2Psystem installed, eDonkey perhaps (Table 11). 130.85.185.13 had 321211 scans from source port 12404. 130.85.185.13 scanned 4662, 4663, 4665 and 4672, all associated with eDonkey and P2P software.[43]

Table 11: 130.85.185.13 port analysis, P2P.

```
$ grep "130.85.185.13" scansall | awk -F : '{print
$7}' | sort | uniq -c | sort -rn | head
 321211 12404
     12 1524
     11 6129
      7 4782
      7 4367
      7 4347
      7 4068
      7 3440
      7 3388
      7 3378
```

```
$ grep "130.85.185.13" scansall | awk -F : '{print
$7,$9}' | sort | uniq -c | sort -rn | head
   3239 12404 4665
   1598 12404 80
   1395 12404 4672
   1187 12404 4662
   1100 12404 1025
    906 12404 3393
    702 12404 3813
    603 12404 5468
    550 12404 4663
    533 12404 7987
```

#9, 130.85.84.164 had 299073 hits on udp port 1304 (Table 12) (boomerang, cisco dns boomerang).[50&51]

Table 12: 130.85.84.164 port analysis port 1304 Cisco Boomerang:

```
$ grep "130.85.84.164" scansall | awk -F : '{print $7,$9,$10}' |
sort | uniq -c | sort -rn | hea
   4919 1304 1214 UDP
   1386 1304 2951 UDP
    563 1304 1239 UDP
    550 1304 2175 UDP
    545 1304 1644 UDP
    491 1304 1745 UDP
    353 1304 1124 UDP
    347 1304 2213 UDP
    340 1304 2540 UDP
    335 1304 3320 UDP
```

Out of Spec Analysis:

The dates appear to be out of sync with actual time. These OOS packets were taken from the file named: oos_report_031219.txt but have a 12/23 date.
The raw output is as follows:

```
12/23-00:05:06.113439 63.194.83.210:2960 -> 130.85.34.11:80
TCP TTL:49 TOS:0x0 ID:15287 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x28DAE4C1  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3530614 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

12/23-00:05:47.654922 147.232.24.11:50894 -> 130.85.34.14:113
TCP TTL:51 TOS:0x0 ID:34000 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x253A11EB  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 1144952123 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

12/23-00:05:57.507763 147.232.24.11:50896 -> 130.85.34.14:113
TCP TTL:51 TOS:0x0 ID:61642 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x264469DE  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 1144953108 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

12/23-00:06:53.433550 129.13.162.95:48859 -> 130.85.185.13:4662
TCP TTL:50 TOS:0x0 ID:54046 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x1661A695  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 550073739 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

12/23-00:07:06.677409 66.48.78.14:54671 -> 130.85.12.6:25
TCP TTL:48 TOS:0x0 ID:32297 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x4B9F6430  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1380 SackOK TS: 944367310 0 NOP WS: 0

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

Total OOS packets: 1177

```
$ grep '\-' oos | wc -l
   1177
```

Table 13: Out of Spec Source and Destination hosts

Top 10 Out of Spec Source Hosts	Top 10 Out of Spec Destination IP's
$ awk -F : '{print $4}' oos1.clean| sort | uniq -c | sort -rn | head	$ awk -F : '{print $6}' oosclean | sort | uniq -c | sort -rn | head
139 194.67.70.10	404 130.85.12.6
116 66.225.198.20	194 130.85.24.44
89 67.114.19.185	139 130.85.66.42
85 209.218.69.253	69 130.85.60.16
62 68.122.128.111	63 130.85.12.4
43 207.228.236.26	48 130.85.185.13
41 64.202.97.130	35 130.85.24.34
29 66.30.247.121	27 130.85.97.11
26 212.36.16.66	23 130.85.84.164
17 64.165.71.94	22 130.85.29.66

How the OOS destination hosts related to the top 10 Alerts:

130.85.12.6 was found in Alert #8 above, High port 65535 tcp - possible Red Worm – traffic, #9, Null scan! and #10 Possible Trojan server activity. The OOS packets are for the most part related to the Null Scan.

130.85.60.16 was found in Alert #1, TFTP - Internal TCP connection to external tftp server, and #4, Connect to 515 from inside

130.85.12.4 was found in Alert #7, NMAP TCP ping!, and #8 High port 65535 tcp - possible Red Worm – traffic.

130.85.185.13 was found in Alert #7, NMAP TCP ping!, and #9 Null scan!, and was found to be a bandwidth hog that is using eDonkey.

130.85.24.34 was found in Alert #10 Possible trojan server activity.

130.85.84.164 was not in the top 10, but was a bandwidth hog.

Below is an example of the data mining effort on the OOS data to examine what the particular IP address was involved with showing Null scan most prevalent with some potential Nmap activity:

```
$ grep "130.85.12.6" alert6.mrg | awk -F : '{print $4}' | sort | uniq -c | sort -rn | head
    162  Null scan!
     78  [UMBC NIDS] External MiMail alert
     36  NMAP TCP ping!
     34  High port 65535 tcp - possible Red Worm - traffic
     20  Possible trojan server activity
```

```
4  Probable NMAP fingerprint attempt
1 25
1  Incomplete Packet Fragments Discarded
1  Happy 99 Virus
1  Bugbear@MM virus in SMTP
```

Recommendations and Discussion:

The 130.85.0.0/16 network, which is registered to the University of Maryland in Baltimore, appears to be heavily trafficked, and has a multitude of machines with direct access to the internet. It is unclear from the logs whether a perimeter firewall exists separating any of these individual systems from the internet. As is common practice for universities, the University of Maryland appears to adhere to the "open" policy of allowing all traffic to and from their networks. This philosophy may be partially responsible for the reason that certain alerts appear more prevalently than others. This deduction seems to be supported from the analyses of the logs. It also seems apparent from, review of other GIAC papers, that the University has less activity during the winter break.[19, 39&40] This is reflected in the amount of activity observed on the 130.85.0.0/16 network. Recommendations garnered from the analyses performed on the data derived from the top 10 alerts are discussed individually hereafter.

Recommendations for Top 10 Alerts:

1. <u>TFTP - Internal TCP connection to external tftp server</u>

Considering the fairly high number of alerts that were logged during the course of a relatively quiet winter break, specifically, 7,869 over the course of a five day period, it would seem the most prudent recommendation would be to block TFTP where its use is unnecessary, or to limit access to only those specific IP's that require TFTP outbound, specifically 130.85.42.1 and 130.85.42.3. TFTP is well known to be insecure and should not be used for external communications. However, dependent upon the University's Security Policy, the decision could be made as to the need for external TFTP connections. If TFTP is to be allowed, then the University may want to consider disabling this rule or modifying it in a similar manner to the following:

```
var HOME_NET 130.85.0.0/16
var EXTERNAL_NET !$HOME_NET
var TFTP_SERVERS [130.85.42.1, 130.85.42.3]

alert tcp ![$EXTERNAL_NET,$TFTP_SERVERS] any -> $EXTERNAL_NET 69 (msg:
"TFTP – Internal TCP connection to external tftp server";)
```

2. <u>EXPLOIT x86 NOOP</u>

The most appropriate recommendation in this instance would simply be to verify that all exposed web servers are patched on an ongoing basis. During the course of the five day tracking, port 80 was a consistent target. One system (210.183.217.72) made attempts 1,696 times on various servers, which is considerable in light of the fact that the entire number of NOOP alerts during the five day period was 4,713. It is not inconceivable that the determined attempts could have compromised an unpatched system. Without seeing the exact rule that this alert is related to, it is impossible to give a solid recommendation regarding whether the rule should be included or excluded from the rule base.

3. SMB Name Wildcard

NetBIOS traffic is, in simplest terms, not a secure protocol. In the course of five days, there were 4,343 alerts generated in relation to potential incursions. NetBIOS traffic is something that should not be allowed to traverse the internet, and never allowed inbound to the network. Depending on the University's Security stance on this type of traffic, the ideal and most effective remedy would be to block NetBIOS traffic from entering or leaving the internal network at a border router. If the University decides to allow NetBIOS traffic, then the rule may be disabled because of the high false positive and noise factors.

4. Connect to 515 from Inside

There were 3,557 alerts with 'connect to 515 from inside' as the alert message. Allowing lpr to go over the internet is highly discouraged, even though there does seem to be some legitimate printer traffic occurring. Again, dependant upon the University's Security policy, an effective recommendation would be to block all connectivity to this port, both inbound and outbound, or, at the least, there should either be specific access lists or firewall rules to govern the traffic. Alternatively, the rule should be fine tuned to monitor specific activity inbound or outbound, as exemplified hereafter:

```
var HOME_NET 130.85.0.0/16
var EXTERNAL_NET !$HOME_NET
var LPR_SERVERS [130.85.162.41]

alert tcp ![$EXTERNAL_NET,$LPR_SERVERS] any -> $EXTERNAL_NET 515 (msg: "
connect to 515 from inside ";)
```

5. High port 65535 udp - Possible Red Worm - Traffic

In this case, with 3,242 alerts posted during the five day tracking period, I would recommend an integrity check of 130.85.163.76, to ascertain whether it is being used for file sharing. If P2P file sharing is against University policy, then appropriate action should be taken. This is an extremely general rule and may be considered for complete removal due to the high false positive possibilities. Personally, I would refine this alert as I believe it is too broad in scope. As of now, the alert seems to be "alert udp any any **<>** any 65535". If 65535 traffic is considered a required monitored port, then it might be more prudent to be more specific and tied directionally, with "alert udp any 1024: **->** any 65535" if the destination is being required to be monitored, otherwise, "alert udp any 65535 **->** any 1024:" if it is the source port that needs to be monitored. By using 1024: only the higher port numbers (• 1024) are included for source or destination, respectively.

6. ICMP SRC and DST Outside Network

This rule seems to generate nothing but noise, and with a total of 1713 alerts during a low volume period of time, it should not be in the rule set. The appropriate recommendation is simply to disable the rule.

7. *NMAP TCP Ping*!
The controlling factor in determining whether this rule is of value is the University's Security Policy. It seems overreaching and unnecessary to know when a user is attempting or utilizing an Nmap tcp ping. In this instance, even with the evidence of 1696 alerts, I concur with T. Beardsley's recommendation to disable the rule.[40]

8. High port 65535 tcp - Possible Red Worm - Traffic
Without additional packet traces it cannot be determined what specifically is being attempted in these alerts, of which, 1086 were logged in the five day tracking. In general, 68.109.149.53 appears to be suspicious, and the connection from 65535 to 1297 could easily imply malicious intent. Although 68.109.149.53 did show in the MyNetWatchman reports, the reports did not specifically involve tcp 65535. A more in depth appraisal of the host, 130.85.97.26, seems the appropriate course of action in this case. The rule recommendation in 5, above, is most applicable, but in this case for TCP.

9. *Null Scan*!
The Null Scan alerts, 662 in total, indicate the probability of a potential malicious port scan which involve no ports or TCP control bit being set. The recommendation is to disable the rule as it only seems to be generating noise, and garners nothing of any real value.

10. Possible Trojan Server Activity
The first priority in this circumstance would be to confirm whether the host 130.85.97.92 has a current antivirus, and to check for Sub7 or other Trojans. The analysis supports the appearance of a compromised system. There were also some scans for Dameware, Remote Admin service to this IP address. This might indicate that the IP address is known to attackers as having some sort of backdoor Trojan. The rule in its present form indicates many false positives, which obviously renders the rule inadequate. There were a total of 327 alerts logged over the course of the five day period. It would be far more effective if the rule was more precise, and related to higher ports; e.g., 1024 and above. "alert tcp any 27374 <> any 1024:". By following this recommendation, the number of false positive counts should be reduced.

11. *130.85.30.3 and 130.85.30.4 and Incomplete Packet Fragments Discarded*:
There were a significantly high number of alerts in relation to 130.85.30.3 activity, 130.85.30.4 activity, and Incomplete Packet Fragments Discarded in relation to the previous 10 analyses. These three types of alerts were responsible for the highest number reported (Table 2 and Table 4). It is interesting to note that 130.85.30.3 and 130.85.30.4 are, or were, Novell Netware servers at one time. Port 524 is NCP (Netware Core Protocol) used by clients to connect to Netware 5 servers and server-to-server communications, and Port 51443 is used by Netware for secure iFolder

communication.[52]

Figure 1 illustrates the relationship between clients that connect to both 130.85.30.3 and 130.85.30.4. Only two systems connect to both servers (68.50.114.89 and 68.55.62.79) each is registered to Comcast Cable Communications (Table 5), and, noteworthy, is the fact that none of these systems was involved with any other alerts or scans. That is, there appears to be nothing malicious occurring.

In reviewing the analyses, it is apparent that the University has an undisclosed reason to monitor these activities; otherwise there is no logical reason to include this rule. The tremendous number of alerts generated does not bear a relationship to anything in particular, i.e., significant amount of noise. Ian Martin thought that these IP addresses may have been honeypots.[39] However, if that were the case, I think there would be more hits on common ports, 80, 443, 445 or other common service ports. The only plausible reason I could think of to include such a general rule would be to notify the University of users that continue to connect to a Novell Netware server that the University is planning to phase out; and they want specific information as to who continues to use them. With the University's requirements being disguised; however, only the most general of recommendations can be made as there is inadequate information for more specificity. With that being said, there does not appear to be a real need for this monitoring, and the most cursory recommendation would be to remove or redefine the rule to suit the University's purposes.

Incomplete Packet Fragments Discarded alerts can be cleaned up by switching to the newer Frag2 preproccessor.[49]

Scans:

There were a total of 17,627,342 scans breaking down into an average of 3,525,281 per day (Figure 2), and 734, 472 per hour (Figure 3). Additionally, scans were used to determine bandwidth utilization, and it was discovered that the highest users of bandwidth were those systems infected with the Welchia/Nachi virus. These infested systems were actively scanning the internet on tcp port 135 (Table 7), and they were involved with thousands of scans (Table 9).

The Welchia worm first ping scans for active hosts, and then follows that up with scans to locate open TCP port 135. There is active scanning for vulnerable systems through the use of internal systems (130.85.0.0/16) seeking the RPC/DCOM vulnerability described by MS03-026 and MS03-039. (See detect #3). According to the analyses, the following systems require patching against the Welchia/Nachi worm MS03-026 and MS03039: 130.85.111.72, 130.85.84.194, 130.85.162.92, 130.85.163.107 and 130.85.80.243 (Table 9).

In light of these facts, I strongly recommend the inclusion of a custom rule, which pertains specifically to the Welchia/Nachi worm. Thus the malicious worm activity can be identified and isolated, and should be observed over the course of one to two months in order to allow time to completely remove the worm. An example would be, SID 2251:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg:"NETBIOS DCERPC
Remote Activation bind attempt"; flow:to_server,established; content:"|05|"; distance:0;
within:1; content:"|0b|"; distance:1; within:1; byte_test:1,&,1,0,relative; content:"|B8 4A
9F 4D 1C 7D CF 11 86 1E 00 20 AF 6E 7C 57|"; distance:29; within:16;
tag:session,5,packets; reference:cve,CAN-2003-0715; reference:cve,CAN-2003-0528;
reference:cve,CAN-2003-0605; classtype:attempted-admin;
reference:url,www.microsoft.com/technet/security/bulletin/MS03-039.asp; sid:2251;
rev:4;)
```

Another system deserving more customized attention is 130.85.110.72. This system seems to be an internet gaming server hosting Medal of Honor internet games (Table 10). University policy will dictate the appropriate response to this server.

P2P can lead to legal issues in most corporations. In this case, 130.85.185.13 appears to be a P2P server using eDonkey (Table 11). P2P can easily lead to viruses and illegal software trading. If the University is concerned with P2P file sharing, then this system along with 130.85.163.76 from the #5 alert (High port 65535 udp – possible Red Worm – traffic) should be removed/examined forthwith.

The last system in the top 10 scanning is 130.85.84.164 (Table 12). It is entirely probable that this system is a content delivery system using a Cisco product called Boomerang for Director Response Protocol (DRP). Boomerang is used to select a content delivery server with the fastest response time from a group of redundant content servers. The boomerang server provides a way to select a content server with the fastest response time from a group of redundant content servers.[51]

Overall Scan Recommendation:

It appears that the University's port scanning pre-processor is set at the default limits of 4 ports in 3 seconds. It would be beneficial to increase the number of ports to 10 while maintaining the time at 3 seconds. There were a significantly large number of DNS scans on port 53. Thus it would be prudent to perhaps exclude the DNS servers with the line:

```
preprocessor portscan-ignorehosts: $DNS_SERVERS
```

An additional, appropriate line could be added for the DRP server if this system proves to be a content delivery system performing a necessary service.

OOS:

Out of Specification packets are those that are corrupt, mangled, or crafted in a manner that appears anomalous. They may or may not be indications of malicious behavior. In the case of a crafted packet, they are malicious; however, in the case of a mangled packet there is no malintent. An example of a crafted packet would be one that is used to determine an operating system. An example of a mangled packet then would be one from a broken router.

Examination of the top ten OOS destination IP's find that most were involved with some type of scan (Null or NMap). This makes sense when you recognize that OOS packets are malformed in one way or another, design or happenstance. Null and NMap scans typically involve sending unusual or malformed packets to hosts during the scanning process. Malformed or unusual packets are used by Nmap during the detection of operating system phase of its scan.

Overall Summary:

The University's IDS system requires fine tuning in order to reduce the large number of false positive alerts. Of the top ten alerts, only four actually produced potentially viable or "interesting" alerts. The others were unnecessary, and should either be disabled or removed completely. There was active scanning which indicated the probability that there are some systems contaminated with worm infestations. It also appears from looking at other reports similar to this, that the University's network traffic was lighter during the instant five day tracking period coincidental with the winter break than during regular session. As a final note, many of the recommendations herein are based on a fairly tight security philosophy. The University's security policy may be different than that presented here. More explicit recommendations could only be determined after a review of the University's Security policy.

Methods of Analysis:

The analysis process was assessed from a number of perspectives. Initially, as was strongly recommended in both the Administrivia and the GCIA Study Guide, and I also found it beneficial to review the various papers available on the SANS/GIAC website. Many of the papers presented analyses that utilized cat, grep, sed and awk, while some others presented information derived from database analysis. Additionally, there were papers presenting a variety of customized scripts.

I made several decisions based on my strengths and weaknesses before settling into a specific method of analysis, and my first choice for analysis was to investigate the scripts. I am neither an SQL nor a MySQL database analyst. Many of the scripts were quite good and customized; however, they lacked the granularity I was looking for. I was limited to running the exact scripts (scriptkiddie style) as I have no proficiency as a perl programmer. Thereafter, I made a feeble attempt to import the data into MS Excel and Access, but those programs were simply not designed for this type of use, and I discontinued the effort after approximately 65,000 lines of data. The scans data was somewhat larger than 1 GB of data, and exceeded the limitations of MS Excel and Access well before the first day of scan data had been read in. Thus, a failure, of sorts.

Next in line was an attempt to use Snortsnarf[53], which ran for approximately 4 days before crashing as a result of a disk error. Again, I was attempting to utilize an analysis flexible enough to be repeated numerous of times. Following this disappointment, I attempted to use Sawmill[54], and was pleased with the resultant graphing for the scan data. The scan analysis for Sawmill took approximately 2 hours to run on a P4 2.6 GHz with 1GB RAM. Regrettably, it did not easily reproduce. I then used Minitab,[55] which is a statistical program, and thus, on the surface, ideally useful. However, it is not designed to look at such large amounts of data, and the results were very poor.

In frustration, I turned to SAS[56], but when I started working with Linux using cat, grep, sed and awk I achieved some degree of success, except that I was unable to get repeatable results using the Chris Calabrese[21] paper. Finally, it all came together after I read Ian Martin's[39] paper which guided me on a more productive path. Not having a fast Linux system readily available, I settled on Cygwin[57] and primarily used the default installation with cat, grep, sed and awk for the bulk of the analysis. I then relied on SawMill for the graphic output on the scan analysis.

Files were cleaned and processed with the following commands:

Alerts:

Combine all the files into one

```
cat alert.* >> alert1.mrg
```

This can be done in dos with a copy command also:

```
copy alert.* alert1.mrg
```

Remove MY.NET and replace with 130.85

```
sed ‘s/MY.NET/130.85/g’ alert1.mrg > alert2.mrg
```

Move the spp_portscan's out of the alert file into a separate file. It was assumed that the individual scan data was a better source for scan analysis.

```
grep ‘spp_portscan’ alert2.mrg >> alert3.mrg
grep –v ‘spp_portscan’ alert2.mrg >> alert4.mrg
```

Remove [**] and replace with : for field manipulation.

```
sed ‘s/\[\*\*\]/:/g’ alert4.mrg >> alert5.mrg
```

Remove -> and replace with :

```
sed ‘s/\->/:/g’ alert5.mrg >> alert6.mrg
```

Perl could also be used with this with the following commands:

```
perl -i.bk -p -e "s/ -> /:/g" alert4.mrg
```

Sample Cleaned Data:

```
12/23-00:01:36.914157  : ICMP SRC and DST outside network : 172.158.185.230 : 172.157.112.213
12/23-00:01:43.007736  : High port 65535 udp - possible Red Worm - traffic : 207.5.180.138:65535 :
130.85.1.4:53
12/23-00:01:52.499472  : EXPLOIT x86 NOOP : 130.67.227.123:3226 : 130.85.190.95:135
12/23-00:01:54.276263  : EXPLOIT x86 NOOP : 130.67.227.123:3233 : 130.85.190.102:135
12/23-00:03:52.579790  : SMB Name Wildcard : 130.85.11.6:137 : 169.254.0.0:137
```

Scans:

The process of Scan data manipulation follows a similar technique as the Alert manipulation with out the need to remove the [**].

There are excess spaces separating the data in the scan data. It makes data mining easier and faster if you have the fields clearly defined with :'s. The file will be slightly smaller too. You can use sed or perl to do this:

```
perl -i.bk -p -e "s/ /:/g" scandata
```

Sample data:

```
Dec:19:00:00:21:217.81.206.31:2943:130.85.190.40:135:SYN:******S*:
Dec:19:00:00:21:217.81.206.31:2944:130.85.190.41:135:SYN:******S*:
Dec:19:00:00:21:217.81.206.31:2946:130.85.190.43:135:SYN:******S*:
Dec:19:00:00:21:217.81.206.31:2947:130.85.190.44:135:SYN:******S*:
Dec:19:00:00:21:217.81.206.31:2948:130.85.190.45:135:SYN:******S*:
Dec:19:00:00:21:217.81.206.31:2949:130.85.190.46:135:SYN:******S*:
Dec:19:00:00:21:217.81.206.31:2951:130.85.190.48:135:SYN:******S*:
Dec:19:00:00:22:130.85.1.3:41446:69.6.43.102:53:UDP::
```

OOS:

OOS data was different too. The following is an example of an OOS alert:

```
12/23-00:05:06.113439 63.194.83.210:2960 -> MY.NET.34.11:80
TCP TTL:49 TOS:0x0 ID:15287 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x28DAE4C1  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3530614 0 NOP WS: 0
```

With this command you can put all the oos_report files into one file with one command. We only need the line with ->.

```
$ cat oos_* | grep '\->' > oos1
```

Cleaned output:

```
12/23-00:05:06.113439 63.194.83.210:2960 -> MY.NET.34.11:80
12/23-00:05:47.654922 147.232.24.11:50894 -> MY.NET.34.14:113
12/23-00:05:57.507763 147.232.24.11:50896 -> MY.NET.34.14:113
12/23-00:06:53.433550 129.13.162.95:48859 -> MY.NET.185.13:4662
12/23-00:07:06.677409 66.48.78.14:54671 -> MY.NET.12.6:25
```

Remove MY.NET and replace with 130.85

```
$ sed 's/MY.NET/130.85/g' oos1 > oos2
```

Remove -> and replace with : for field delimiter

```
$ sed 's/ \-> /:/g' oos2 > oos3
```

Remove blank spaces and replace with : field delimiter

```
$ sed 's/ /:/g' oos3 > oosclean
```

```
12/23-00:05:06.113439:63.194.83.210:2960:130.85.34.11:80
12/23-00:05:47.654922:147.232.24.11:50894:130.85.34.14:113
12/23-00:05:57.507763:147.232.24.11:50896:130.85.34.14:113
```

```
12/23-00:06:53.433550:129.13.162.95:48859:130.85.185.13:4662
12/23-00:07:06.677409:66.48.78.14:54671:130.85.12.6:25
12/23-00:08:30.301644:195.5.130.67:34535:130.85.12.6:25
```

***Data mining**:*
For retrieving important results from the data I used cat, grep, awk, sort, uniq, and head to pull relevant information from the collated and cleaned data. Please see the relevant sections above for the commands used to retrieve the data.

References:

State of Intrusion Detection: Please see Section One page 9 for references.

1. Internet Protocol Tutorial, Classes of IP Addresses, IP Broadcast and IP Multicast, http://compnetworking.about.com/library/weekly/aa042400b.htm
2. Packetstorm Security. Mixter's area. http://packetstormsecurity.nl/groups/mixter/
3. Common Vulnerabilities and Exposures "A hacker utility, back door, or Trojan Horse is installed on a system, e.g. NetBus, Back Orifice, Rootkit, etc." 1999. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660
4. Aleph One. Phrack online magazine. Issue 49. "Smashing the Stack for Fun and Profit" http://www.phrack.org/show.php?p=49&a=14
5. Computer Emergency Response Team, Software Engineering Institute, Carnegie Mellon University, "CERT® Multiple Vulnerabilities in lpd" November 05, 2001. http://www.cert.org/advisories/CA-2001-30.html
6. Mixter Security. http://mixter.void.ru/ Q backdoor
7. Backdoor Q access? May 04 2001. http://lists.jammed.com/incidents/2001/05/0039.html
8. Security Incidents: Re: Deny IP spoof from 255.255.255.255. Jul 06, 2001 http://seclists.org/lists/incidents/2001/Jul/0023.htm
9. Mario Ricci. GIAC, GCIA Practical. June 18, 2003. http://www.giac.org/practical/GCIA/Mario_Ricci_GCIA.pdf
10. Al Maslowski-Yerges. GIAC, GCIA Practical. January 5, 2003. http://www.giac.org/practical/GCIA/Al_Maslowski-Yerges_GCIA.pdf
11. Les M Gordon. GIAC, GCIA Practical. November 22, 2002. http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc
12. American Registry for Internet Numbers www.arin.org
13. Migrating Microsoft Proxy Server 2.0 configuration. Microsoft Technet. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/isa/proddocs/isadocs/cmt_upgradprx2indep.asp
14. Squid Configuration Manual. http://squid.visolve.com/squid24s1/network.htm
15. Socks Proxy. http://www.gnetlibrary.org/docs/gnet-socks.html
16. Common Vulnerabilities and Exposures. Many used. Will list the numbers for brevity. CVE-1999-0710, CVE-1999-1481, CVE-2001-0142, CVE-2002-0068, CAN-2002-1001, CVE-1999-0291, CVE-2001-0239, CVE-2001-0658. http://www.cve.mitre.org
17. Computer Emergency Response Team, Software Engineering Institute, Carnegie Mellon University, "CERT® Vulnerability Note VN-98.03" http://www.cert.org/vul_notes/VN-98.03.WinGate.html

18. Dshield.org IP lookup. http://www.dshield.org/ipinfo.php
19. Don Murdoch. GIAC, GCIA Practical. June 11, 2003. http://www.giac.org/practical/GCIA/Don_Murdoch_GCIA.pdf
20. Mark Embrich. GIAC, GCIA Practical. February 14, 2002. http://www.giac.org/practical/Mark_Embrich_GCIA.htm
21. Chris Calabrese. GIAC, GCIA Practical. December 2001. http://www.giac.org/practical/Chris_Calabrese_GCIA.html
22. Road Runner Scanning. January 21, 2003. http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00192.html
23. Symantec. W32.Welchia.Worm. August 18, 2003. http://www.sarc.com/avcenter/venc/data/w32.welchia.worm.html
24. Common Vulnerabilities and Exposures (under review) http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352+
25. Microsoft. Microsoft Security Bulletin MS03-026. July 16, 2003. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp
26. Microsoft. Microsoft Security Bulletin MS03-039. September 10. 2003. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-039.asp
27. 07.26.2003 : Windows RPC DCOM Remote Exploit W2k+XP Targets. http://www.k-otik.com
28. Linda Bourbeau. GIAC, GCIH Practical. October 5, 2003. http://www.giac.org/practical/GCIH/Linda_Bourbeau_GCIH.pdf
29. Brian Porter. GIAC, GCIH Practical. November 2, 2003. http://www.giac.org/practical/GCIH/Brian_Porter_GCIH.pdf
30. David Markle, LOGS: GIAC GCIA Version 3.4 Practical Detect (Markle). December 5, 2003. http://cert.uni-stuttgart.de/archive/intrusions/2003/12/msg00026.html
31. Joanne Schell. GIAC, GCIH Practical. August 25, 2003. http://www.giac.org/practical/GCIA/Joanne_Schell_GCIA.pdf
32. Computer Emergency Response Team, Software Engineering Institute, Carnegie Mellon University. Vulnerability Note VU#568148. Microsoft Windows RPC vulnerable to buffer overflow. http://www.kb.cert.org/vuls/id/568148
33. Computer Emergency Response Team, Software Engineering Institute, Carnegie Mellon University, “CERT® Advisory CA-2003-19 Exploitation of Vulnerabilities in Microsoft RPC Interface”. http://www.cert.org/advisories/CA-2003-19.html
34. False Positives: A users guide to making sense of IDS Alarms. February 2003. http://www.icsalabs.com/html/communities/ids/whitepaper/FalsePositives.pdf
35. My | NetWatchman. IP lookup. http://www.mynetwatchman.com
36. Michael Hotaling. GIAC, GCIA Practical. September 15, 2002. http://www.giac.org/practical/GCIA/Michael_Hotaling_GCIA.pdf
37. Marshall Heilman. GIAC, GCIA Practical. December 5, 2003. http://www.giac.org/practical/GCIA/Marshall_Heilman_GCIA.pdf
38. Mark Embrich. GIAC, GCIA Practical. February 14, 2002. http://www.giac.org/practical/Mark_Embrich_GCIA.htm
39. Ian Martin. GIAC, GCIA Practical. July 17, 2003. http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf

40. Tod Beardsley. GIAC, GCIA Practical. May 8, 2002. http://www.giac.org/practical/Tod_Beardsley_GCIA.doc
41. Jasmir Beciragic. GIAC, GCIA Practical. February 2, 2001. http://www.giac.org/practical/Jasmir_Beciragic_GCIA.doc
42. F-Secure Virus Descriptions: Adore. http://www.f-secure.com/v-descs/adore.shtml.
43. Dshield.org. Distributed Intrusion Detection System. Port lookup. http://www.dshield.org/port_report.php
44. Scott Shinberg. GIAC, GCIA Practical. July 11, 2001. http://www.giac.org/practical/Scott_Shinberg_GCIA.doc
45. Snort. Signature ID 628 Scan Nmap TCP. http://www.snort.org/snort-db/sid.html?sid=628
46. Radware. Linkproof. http://www.radware.com/content/products/lp/default.asp
47. Snort. Snort Signature lookup. http://www.snort.org/snort-db/sid.html?sid=623
48. Robert Sorenson. GIAC, GCIA Practical. February 2, 2001. http://www.giac.org/practical/Robert_Sorensen_GCIA.htm
49. Martin Roesch. November 26, 2001. Incomplete Packet Fragments Discarded discussion. http://www.mcabee.org/lists/snort-users/Nov-01/msg00820.html
50. Cisco Systems. Cisco Boomerang service. http://www.cisco.com/en/US/products/sw/iosswrel/ps1839/products_feature_guide09186a0080087d3f.html
51. Cisco Systems. Release Notes for Cisco Content Routing Software. UDP port 1304 Boomerang.http://www.cisco.com/univercd/cc/td/doc/product/webscale/cr/cr4430/cr_11rn.htm
52. Tek-Tips Forums. Novell Netware 6 Faqs. How do I get NetWare 6 Web Services to work? http://www.tek-tips.com/gfaqs.cfm/pid/871/fid/3352 secure iFolder port 51443
53. Silicon Defense. Snortsnarf. http://www.silicondefense.com/software/snortsnarf/
54. Sawmill. http://www.sawmill.net
55. Minitab. Statistical Software. http://www.minitab.com
56. SAS. Statistical Software. http://www.sas.com/
57. Cygwin. Linux in Windows. http://www.cygwin.com