



## **SANS GCIA Practical ver 3.3**

**Al Williams**

**SANS Parliament Hill  
Ottawa Aug 2002**

# Table of Contents

<b>Part 1 – Describe the State of Intrusion Detection .....</b>	<b>4</b>
<b>Trinux: A Pocket Operating System .....</b>	<b>4</b>
1.1 What is Trinux? .....	4
1.2 Why Trinux? .....	4
1.3 Building and Using Trinux .....	5
1.4 Making Packages .....	6
1.5 ...and This Little Piggie Went to Market! .....	7
1.6 The Down Side .....	8
1.7 Securing Against Unauthorized Use .....	9
1.8 Conclusion .....	9
1.9 References .....	9
<b>Part 2 Network Detects .....</b>	<b>10</b>
2.1 Detect 1 Synscan .....	10
2.2 Detect 2 DDoS Attack on IRC Server .....	18
2.3 Detect 3 HTTP Connect Attempt .....	24
<b>Part 3 Analyze This! .....</b>	<b>33</b>
3.1 Log Files .....	34
3.2 Methodology .....	34
3.3 Scan Logs .....	35
3.4 Alert Logs .....	41
3.5 OOS Logs .....	44
3.6 Detects .....	46
3.6.1 SYN-FIN Scan! .....	46
3.6.1.1 Description .....	46
3.6.1.2 Recommendations .....	48
3.6.1.3 References .....	49
3.6.2 SMB Name Wildcard .....	49
3.6.2.1 Description .....	49
3.6.2.2 Recommendations .....	50
3.6.2.3 References .....	51
3.6.3 IRC Evil – Running XDCC .....	51
3.6.3.1 Description .....	51
3.6.3.2 Recommendations .....	52
3.6.3.3 References .....	52
3.6.4 External RPC call .....	53
3.6.4.1 Description .....	53
3.6.4.2 Recommendations .....	54
3.6.4.3 References .....	55
3.7 Internal Systems of Interest .....	55
3.7.1 Nimda Infection .....	55
3.7.2 Possible Trojan Horses .....	56
3.7.3 IRC XDCC .....	57
3.7.4 TFTP – Internal/External Connections .....	57
3.7.5 Port Scanning .....	58
3.8 External Addresses of Interest .....	59
3.8.1 80.11.207.237 .....	59
3.8.2 217.148.2.61 .....	60
3.8.3 195.199.74.91 .....	60
3.8.4 195.101.94.208 .....	61
3.8.5 136.145.82.46 .....	61

3.9 Top Talkers .....	62
3.10 Link Graph of Events of Interest.....	63
3.11 Defensive Recommendations .....	64
3.12 Conclusion .....	66
<b>References.....</b>	<b>67</b>
The State of Intrusion Detection.....	67
Synscan.....	67
DdoS Attack on IRC Server.....	67
HTTP Connect Attempt .....	67
Analyze This .....	68
<b>Appendix A .....</b>	<b>69</b>
<b>Appendix B .....</b>	<b>70</b>
<b>Appendix C .....</b>	<b>71</b>

## **Summary**

This report is a culmination of the Intrusion Detection skills taught at the SANS Parliament Hill, Ottawa Ontario conference and those learned on the job. Over the course of this report several topics and issues are covered, including a brief look at the Trinux Operating System and its potential uses in the Intrusion Detection field. Three detailed network detects are included as well as an analysis of five days worth of Snort generated logs from a large University.

## **Part 1 – Describe the State of Intrusion Detection**

### **1. Trinux: A Pocket Operating System**

#### **1.1 What is Trinux?**

Trinux is a Linux based ramdisk operating system developed by Mathew Franz and released in April 1998. It is an open source operating system that conveniently fits on either two or three floppy disks, or a CDROM. Since this handy little program runs entirely in RAM it can be booted on virtually any x86 PC without over writing the current hard disk, or the file system contained within. It can be booted from a floppy or CDROM and can load additional packages from floppy disks, VFAT / NTFS file system, or from a network server. The current release, 0.80cr2, is based on Linux Slackware 7.1 and supports 2.4.x kernels using [Busybox](#) to replace many of the common UNIX commands with stripped down, more compact versions.

Trinux will run on an i486 or better computer with at least 12 -16 Megs of RAM and a few Mbytes of hard disk space for saving logs or files to, though not required. It comes with built in support for many popular network cards as well as DHCP.

#### **1.2 Why Trinux?**

The system administrator or information security specialist can transform an ordinary workstation into a powerful security management tool by just booting Trinux. Having done this, Trinux can be used to monitor and map TCP/IP networks, and to conduct security research, analyze network traffic, as well as performing vulnerability testing using a wide range of available packages. These packages can be found on the SourceForge [ftp server](#) and include tools falling into the following groups:

- Sniffers and Network Analyzers
- Network Mapping/Vulnerability Scanning
- Intrusion Detection
- Packet Generators
- Proxies and Tunnelling Tools
- Encryption Packages
- Web Utilities
- Network Utilities
- Text Editors
- Disk and File system Tools

### **1.3 Building and Using Trinux**

The ISO images for Trinux can be downloaded from [SourceForge](#), or from [Trinux.Org](#), along with the various packages that can be run with it. Having an ISO viewer, such as [WinISO](#), to modify and customize the CDROM version of Trinux is helpful, but not a requirement as the ISO comes with most of the packages you will require. You will have to edit the TUX/servers file to specify a different location for network loading if you will be running your own Trinux package ftp server. The latest release worked great, right off the wire with no modification needed.

The floppy version requires the [rawrite](#) software to make it bootable. After the floppy version has been downloaded one uses rewrite to transfer it to a floppy to make it bootable. Once the bootable floppy is created it can be modified through windows by simply using WordPad. Since the packages are quite small several of them can be loaded from a single, or from multiple floppies, depending on what packages are required for the job at hand. I was able to make a fairly comprehensive package list that will fit on a single floppy disk.

Loading Trinux requires the computer to be bootable from either the floppy or CDROM drive, depending on the media type you are using. Rebooting with your chosen media, Trinux will automatically identify and configure most modern network cards, and if the network is using DHCP it will negotiate and configure for that IP. Trinux can also be configured manually if not using DHCP through the ifconfig command.

Using the 'getpkg' command you can download and install from the network any required packages. You can use the 'getpkgall' to download and install all of the packages stored on the specified location(s) in the TUX/server file. In a matter of a few seconds/minutes you are up and running.

Most common UNIX commands are supported, though they are the stripped down BusyBox versions. If you're looking for any cool GUI's you will not find

them in Trinux, everything is done through one of seven command line consoles accessed through alt + F1 through F7.

When you run Trinux and do an 'ls' you will see:

```
bin   dev   floppy linuxrc sbin   usr
boot  etc   home   mnt   tmp   var
cdrom fixed lib   proc  tux
```

The important directories, as quoted from [SourceForge](#) are:

- **bin** - the busybox standard UNIX utilities
- **sbin** - more unix utilities and many of the Trinux administration scripts (fmount, fumount, getpkg, gethome, savecfg, etc.)
- **dev** - device files the OS uses to access terminals, ramdisks, and various types of fixed and removable media
- **etc** - the standard location for Linux configuration files
- **tux** - the location of Trinux configuration files that were copied from the boot floppy during startup. This is actually a link (like a windows shortcut or Mac alias) to /etc/tux.

## **1.4 Making Packages**

In order for a package to be loaded it is required to have a .tgz extension for Trinux to recognize and load upon booting. To test this I downloaded the small [Siphon](#) passive OS tool and packaged it as a siphon.tgz, and put this on a floppy with several other packages from [www.trinux.org](#). Ensuring that the package was then listed in the tux/config/pkglist, the package was loaded and worked with no problems. Simply ensure it is in the path or you are in the appropriate directory.

If different files require different directories the package can be located in separate folders with the appropriate destination locations. An example of this can be seen in the Snort.tgz used to install Snortas seen in figure 1.4.1.

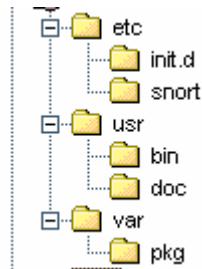


Figure 1.4.1

## **1.5 ...and This Little Piggie Went to Market!**

To use Trinux in an IDS role, a test machine in my home network was booted with the latest version (0.80cr2). Using an obsolete Pentium 200MX with 64megs of RAM was thought to be a good example of how Trinux can be run successfully on older machines that do not require large amounts of resources.

Once the console was up and running the command “getpkg snort” was used to download and install the Snort IDS. Snort comes with a basic set of rules that were used for this testing.

Next, a medium for storing the generated log files was needed. Using the

```
mount -t vfat /dev/hda1 /hda1
```

command, the native hard drive was mounted and a new directory Trinux was created. The Snort required .log was then created in the new Trinux directory.

After the initial setup Snort was run using the following command,

```
C:>snort -c /etc/snort/snort.conf -l /hda1/Trinux/snort.log -v
```

and then left to run to gather it's log file.

Another system, a Pentium III with 256 megs of RAM was set up using the same Trinux bootable CDROM removed from the Snort system. Since the OS runs only in memory the disc can be removed after installation.

On the Pentium III system the scanner Nmap was started with the intention of generating some standard type alerts. With the command

```
C:>nmap -sF -O -p 1-1024 192.168.2.5
```

Nmap returned:

```
All 1024 scanned port on (192.168.2.5) are: closed
Remote OS guesses: Linux Kernel 2.4.0 - 2.5.20, Linux 2.4.19-pre4
on Alpha, Linux Kernel 2.4.0 - 2.5.20 w/o tcp_timestamps
```

The first item of note is that all 1024 registered ports are closed by default on the Trinix system; ports are only opened as required. Another item of interest is that Nmap was able to identify the OS as most likely being the Linux Kernel 2.4.0 – 2.5.20, which covers the 2.4.X kernels that Trinix supports.

After the scan was completed the alert log was examined and what was found was very much the same, as one would expect from Snort. For all intents and purposes the IDS worked the same as its big brother found at Snort.org. Listed below are examples of what was found in the alert file.

```
[**] [111:8:1] spp_stream4: STEALTH ACTIVITY (FIN scan) detection [**]  
12/27-10:47:24.085643 192.168.2.7:51067 -> 192.168.2.5:14  
TCP TTL:48 TOS:0x0 ID:27679 IpLen:20 DgmLen:40  
*****F Seq: 0x0 Ack: 0x0 Win: 0x400 TcpLen: 20
```

```
[**] [100:1:1] spp_portscan: PORTSCAN DETECTED to port 14 from  
192.168.2.7 (STEALTH) [**]  
01/02-13:08:05.430000
```

```
[**] [111:8:1] spp_stream4: STEALTH ACTIVITY (FIN scan) detection [**]  
12/27-10:47:24.085669 192.168.2.7:51067 -> 192.168.2.5:111  
TCP TTL:48 TOS:0x0 ID:40072 IpLen:20 DgmLen:40  
*****F Seq: 0x0 Ack: 0x0 Win: 0x400 TcpLen: 20
```

```
[**] [111:8:1] spp_stream4: STEALTH ACTIVITY (FIN scan) detection [**]  
12/27-10:47:24.085692 192.168.2.7:51067 -> 192.168.2.5:189  
TCP TTL:48 TOS:0x0 ID:29613 IpLen:20 DgmLen:40  
*****F Seq: 0x0 Ack: 0x0 Win: 0x400 TcpLen: 20
```

The Trinix version of Snort comes with a good set of pared down rules, however, the newer rule sets and customized rules can be used.

## **1.6 The Down Side**

The down side to an Operating System such as Trinix is the complete portability and ease of installation. In a very short period of time a system can be converted into a platform to launch any number of illicit activities on an internal network. Tools such as Nmap and Hping make the mapping of targets from internal sources possible, but very hard to detect and even harder to catch.

After having installed Trinix and using it to perform various jobs, the system was examined for any sign of Trinix being left after system shut down. Other than the specifically generated log files no evidence of Trinix was found. Since Trinix runs completely in RAM any intruder could therefore use Trinix and have a reasonable expectation of never being found out through the examination of the computer used. A simple re-boot with the Trinix bootable media removed will wipe any evidence of use.

## **1.7 Securing Against Unauthorized Use**

The number one method for the prevention of unauthorized use of Trinux is physical security. Using some form of access control, such as swipe or smart cards, will help prevent uninvited guests from using internal resources for their own purposes. One has to have physical access to the computer to boot the Trinux Operating Systems.

Making the floppy and CDROM drives non bootable through the system BIOS will make it next to impossible for users to try installing this OS. A strong password on the BIOS that is known only to the IT and security staff is essential to maintaining system integrity.

Deployment of IDS sensors on the internal segments of networks to detect anomalies such as those that may be generated by the use of these types of Operating Systems would be prudent.

## **1.8 Conclusion**

An Operating System that can be carried around inside a shirt pocket and installed on any PC in minutes is very powerful. Security specialists can do vulnerability testing, analyze network traffic, map networks, and conduct security research by using the Trinux Operating System. With it's power, size, flexibility, and portability Trinux will surely become a valuable part of the system administrator's and security professional's toolbox.

## **1.9 References**

<http://www.busybox.net/>

<http://trinux.sourceforge.net/>

<http://www.bernd-roehling.de/trinux/doc/>

<http://home.purplehaze.ch/~olivier/pub/writings/trinux.html>

<http://www.usenix.org/publications/login/2000-12/pdfs/forte.pdf>

<http://www.landfield.com/isn/mail-archive/1998/Sep/0105.html>

## 2 Network Detects

### 2.1 Detect 1          Synscan

This detect was originally posted to the [intrusions@incidents.org](mailto:intrusions@incidents.org) mail list on October 8<sup>th</sup>, 2002.

```
03:41:06.654488 62.153.209.202.21 > 46.5.59.164.21: SF
624843878:624843878(0) win 1028 (ttl 30, id 39426, bad cksum 90c6!)
0x0000      4500 0028 9a02 0000 1e06 90c6 3e99 dlca E..(.....>...
0x0010      2e05 3ba4 0015 0015 253e 5c66 7f90 16ed ..i.....%>\f....
0x0020      5003 0404 218a 0000 0000 0000 0000      P...!.....
03:46:43.754488 62.153.209.202.21 > 46.5.78.72.21: SF
144268715:144268715(0) win 1028 (ttl 30, id 39426, bad cksum 7d24!)
0x0000      4500 0028 9a02 0000 1e06 7d24 3e99 dlca E..(.....}$>...
0x0010      2e05 4e48 0015 0015 0899 5dab 1a0e cbb2 ..NH.....].....
0x0020      5003 0404 da04 0000 0000 0000 0000      P.....
03:52:43.834488 62.153.209.202.21 > 46.5.139.198.21: SF
1246535396:1246535396(0) win 1028 (ttl 30, id 39426, bad cksum 3ea5!)
0x0000      4500 0028 9a02 0000 1e06 3ea5 3e99 dlca E..(.....>.>...
0x0010      2e05 8bc6 0015 0015 4a4c 9ee4 3436 4681 .....JL..46F.
0x0020      5003 0404 83a2 0000 0000 0000 0000      P.....
04:52:47.014488 62.153.209.202.21 > 46.5.129.12.21: SF
1458208403:1458208403(0) win 1028 (ttl 30, id 39426, bad cksum 4a60!)
0x0000      4500 0028 9a02 0000 1e06 4a60 3e99 dlca E..(.....J`>...
0x0010      2e05 810c 0015 0015 56ea 7e93 10d1 3a7c .....V.~....:|
0x0020      5003 0404 d27a 0000 0000 0000 0000      P....z.....
05:03:53.804488 62.153.209.202.21 > 46.5.113.69.21: SF
1861670173:1861670173(0) win 1028 (ttl 30, id 39426, bad cksum 5a27!)
0x0000      4500 0028 9a02 0000 1e06 5a27 3e99 dlca E..(.....Z'>...
0x0010      2e05 7145 0015 0015 6ef6 d51d 5f94 6b46 ..qE....n..._.kF
0x0020      5003 0404 f41d 0000 0000 0000 0000      P.....
05:13:28.744488 62.153.209.202.21 > 46.5.30.119.21: SF
928358045:928358045(0) win 1028 (ttl 30, id 39426, bad cksum acf5!)
0x0000      4500 0028 9a02 0000 1e06 acf5 3e99 dlca E..(.....>...
0x0010      2e05 1e77 0015 0015 3755 9e9d 619f 67a4 ...w....7U..a.g.
0x0020      5003 0404 b6a4 0000 0000 0000 0000      P.....
05:34:13.984488 62.153.209.202.21 > 46.5.251.75.21: SF
2087625956:2087625956(0) win 1028 (ttl 30, id 39426, bad cksum ce21!)
0x0000      4500 0028 9a02 0000 1e06 ce21 3e99 dlca E..(.....!>...
0x0010      2e05 fb4b 0015 0015 7c6e a4e4 7a20 d2fc ...K....|n..z...
0x0020      5003 0404 0897 0000 0000 0000 0000      P.....
06:16:36.914488 62.153.209.202.21 > 46.5.211.130.21: SF
1778720955:1778720955(0) win 1028 (ttl 30, id 39426, bad cksum f6e8!)
0x0000      4500 0028 9a02 0000 1e06 f6e8 3e99 dlca E..(.....>...
0x0010      2e05 d382 0015 0015 6a05 20bb 338a 5f51 .....j...3._Q
0x0020      5003 0404 8232 0000 0000 0000 0000      P....2.....
07:04:06.614488 62.153.209.202.21 > 46.5.82.232.21: SF
1842273904:1842273904(0) win 1028 (ttl 30, id 39426, bad cksum 7982!)
0x0000      4500 0028 9a02 0000 1e06 7982 3e99 dlca E..(.....y.>...
0x0010      2e05 52e8 0015 0015 6dce de70 211e 32ad ..R.....m..p!..2.
0x0020      5003 0404 825d 0000 0000 0000 0000      P....].....
```

```

07:06:04.684488 62.153.209.202.21 > 46.5.8.237.21: SF
1086370025:1086370025(0) win 1028 (ttl 30, id 39426, bad cksum c37d!)
0x0000      4500 0028 9a02 0000 1e06 c37d 3e99 dlca  E..(.....}>...
0x0010      2e05 08ed 0015 0015 40c0 b0e9 7560 b74d  .....@...u`.M
0x0020      5003 0404 4e0b 0000 0000 0000 0000      P...N.....
07:25:33.424488 62.153.209.202.21 > 46.5.219.61.21: SF
966426997:966426997(0) win 1028 (ttl 30, id 39426, bad cksum ee2f!)
0x0000      4500 0028 9a02 0000 1e06 ee2f 3e99 dlca  E..(...../>...
0x0010      2e05 db3d 0015 0015 399a 8175 6685 18a9  ...=.....9..uf...
0x0020      5003 0404 5cd7 0000 0000 0000 0000      P...\.....
07:45:04.394488 62.153.209.202.21 > 46.5.189.98.21: SF
239963126:239963126(0) win 1028 (ttl 30, id 39426, bad cksum c0b!)
0x0000      4500 0028 9a02 0000 1e06 0c0b 3e99 dlca  E..(.....>...
0x0010      2e05 bd62 0015 0015 0e4d 8bf6 417a 8c20  ...b.....M..Az..
0x0020      5003 0404 4d12 0000 0000 0000 0000      P...M.....

```

## **2.1.2 Source of Trace**

This detect was gathered from file [2002.6.7](#) as per the current assignment ver 3.3 instructions. The bad checksums in the trace are the result of log IP sanitation.

## **2.1.3 Detect Generated By**

This detect was generated by a Snort Win32 IDS ver 1.8.4 with a standard rule set 1.8.1. I ran Snort with the following command and then searched through the alert file.

```
Snort -c /path/snort.conf -r /path/2002.6.7 -l /path/snort.log
```

In the alert file I noted many alerts as follows:

```

[**] [1:624:1] SCAN SYN FIN [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/06-16:12:02.064488 62.153.209.202:21 -> 46.5.134.18:21
TCP TTL:30 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
*****SF Seq: 0x5527CDDC Ack: 0x2E97E104 Win: 0x404 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS198]

```

These alerts were generated by this rule in response to the SYN/FIN flags being set:

```

alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS198/scan_SYN FIN
Scan"; flags: SF; classtype: info-attempt; reference: arachnids,198;)

```

I then ran windump ver 3.5.2 on the raw logs to generate the trace using the command:

```
Windump -r /path/2002.6.7 -nvXs 1514 ip and host 62.153.209.202 and port 21 > Newfile.txt
```

### **2.1.3 Possibility the Address is Spoofed**

Not likely. Since the attacker requires a response from the destination host to complete the connection, spoofing is not suspected. The attackers IP resolves through [Ripe.net](http://Ripe.net) to:

<b><u>inetnum:</u></b>	62.153.209.200 - 62.153.209.207
netname:	BERGKEMPER-NET
descr:	Ursula Bergkemper EDV-Engineering
country:	DE
admin-c:	<a href="#">WB6989-RIPE</a>
tech-c:	<a href="#">WB6989-RIPE</a>
status:	ASSIGNED PA
notify:	auftrag@nic.telekom.de
mnt-by:	<a href="#">DTAG-NIC</a>
changed:	auftrag@nic.telekom.de 20010129
source:	RIPE

### **2.1.4 Description of the Attack**

Over the course four hours the attacker sent packets to port 21 of random IP's in the 46.5.0.0/16 block of addresses with no noticeable patterns in timing. No reply from any host was noted during this scan. There are multiple [CVE's](#) listed against the FTP port.

### **2.1.5 Attack Mechanism**

My initial suspicion was that this trace was the Ramen worm trolling for victims to infect through the scanning of port 21. However, without any further traffic other than the Snort Log to analyze, it is difficult to tell if the destination host answered this scan or not. This traffic is probably the use of a scanning tool called [Synscan](#) by Psychoid, which the Ramen worm is built around.

Synscan version 1.5 and 1.6 are UNIX/Linux based tools that scan using the TCP flags of SYN/FIN set, as an attempt to hide their scanning from older versions of Intrusion Detection Systems and Firewalls. When a SYN/FIN is received the destination host replies with a RESET if it doesn't have that port

open, or a SYN/ACK if the service were running. Synscan would then send a RESET back to close the connection. Now knowing the host is running the service a SYN would be sent to establish a legitimate connection with the host to grab the FTP banner. Since this is a legitimate connection it would not flag any IDS and would not be logged.

The tool can scan either a class a/b/c network, or can use an 'Infile' that contains numerical IP's. Port, ports, or port ranges can also be set.

```
05:34:13.984488 62.153.209.202.21 > 46.5.251.75.21: SF
2087625956:2087625956(0) win 1028 (ttl 30, id 39426, bad cksum ce21!)
0x0000      4500 0028 9a02 0000 1e06 ce21 3e99 d1ca  E..(.....!>...
0x0010      2e05 fb4b 0015 0015 7c6e a4e4 7a20 d2fc  ...K....|n.z...
0x0020      5003 0404 0897 0000 0000 0000 0000  P.....
```

The reflective ports, packet ID of 39426, ttl of 30, window size of 1028, and TCP flags SYN and FIN set are all fingerprints of the Synscan tool and several exploit packages that are built around it, including the Ramen Worm and the T0rn Root kit. FTP connections are started from a client's ephemeral port to a server's port 21. It is not normal behavior to see a connection attempt from port 21 to port 21. Once a connection is established the server/client negotiate an ephemeral port to transfer data to, and data flows from server port 20 to the negotiated ephemeral port of the client.

The Ramen Worm is a self-propagating compilation of tools, including Synscan, which looks for, and exploits, three well-known vulnerabilities in Red Hat 6.2 and 7.0. Unpatched Red Hat version 6.2 has [rpc.statd](#) and [wu-ftpd](#) vulnerabilities, while version 7.0 has an [LPRng](#) vulnerability that the worm takes advantage of to propagate.

The [T0rn Root kit](#) by RippEd is a precompiled set of binaries including, in some versions, the Synscan tool rename to t0rnscan. This root kit also includes a log cleaner, a log parser, and a network sniffer.

In this trace the attacker is trying to grab the FTP banner to gather intelligence for further attempts to exploit known vulnerabilities in the FTP service. Synscan is known as a very fast scanner, capable of scanning entire net blocks in a very short time. With the randomization of the destination IP, it leads to the suspicion that an Input file is being used that could be either a script to generate random IP address over a wide range, or he targets hosts he has previously identified as running Linux through either active or passive OS detection. A passive tool known as [p0f](#), or an active mapper such as [Nmap](#) might have been used. The code could also have been altered to slow the scan down in an attempt to avoid being as noticeable. This might have also been an attempt to divert analysts from a more covert attack/scan.

Since it is not known if any replies were sent from the scanned IP's, there is no evidence that this was, or was not a successful scan. With out a logged

three-way handshake, there is no way to prove this was anything other than Synscan 1.5/1.6.

### 2.1.6 Correlation

This 1.5/1.6 version of Synscan is rather old as far scanners go and it does not seem to be used that much any more with the release of newer versions. Most of the correlation found appears in postings to mail list archives from 1999-2001. Several examples of more recent postings are [Daniel Martin's](#) posting to incidents.org on Oct 18 2001:

The complete class B address space was scanned very rapidly first by a SYN-FIN scan, followed by a TCP-Connect scan to the ports found open. The second connection was almost immediate, suggesting a single tool doing both scans.

The host appears to run linux (from passive OS fingerprint) and the host was both a dns server and mail exchanger for its domain. Log information from a single host is below, Times are in EET (GMT+0200)

```
Oct 16 19:54:25.228427 XXX.XXX.XXX.XXX.22 > YYY.YYY.YYY.YYY.22: SF [tcp sum ok] 415795998:415795998(0) win 1028 (ttl 27, id 39426)
```

and from [James C. Slora](#) to intrusions@incidents.org on 9 Jul 2002:

I've seen quite a few Synscans in the past few days - TCP 21 seems to be a particular favorite. Based on Donald Smith's comments about Synscan identification in previous posts, here's my summary for one small network.

Mostly Synscan 1.9++, (source=target port, ACK set in SYN, random ID and window), **one 1.5/1.6 scan (ID=39426, SYNFIN)**. Sequence numbers remain constant throughout each scan on this small subnet, and all scans took only a few hundredths of a second. One scan targets only the common target host on this subnet, most of the scans sweep the subnet in order, and a few of them scan the subnet randomly but feature the common target host in an interesting position within the scan.

According to [isc.incidents.org](#) the attackers IP is the only one from the 62.153.209/24 net that has been listed as having been reported for actively targeting other hosts. Port 21, FTP is also listed as one of the top ten attacked ports today.

### 2.1.7 Evidence of Active Targeting

This scan is directly targeting these IP's with the expectation of receiving a reply in order to pursue vulnerable systems for exploit.

### **2.1.8 Severity**

Criticality: 3

This scan was directed at the FTP port that has numerous vulnerabilities listed; however it used a SYN/FIN in an attempt to bypass older IDS and was easily detected with this updated / modern ID. What these target hosts are or what they are running for services is unknown.

Lethality: 4

A successful connection to an unpatched [FTP](#) server could lead to exploitation that could include the allowing of read/write to files, FTP Bounce, DoS attacks, root access, and Buffer Overflow attacks to name a few.

System Counter Measures: 3

Nothing is known about the security level of the hosts on the 46.5/16 net. If they were running vulnerable FTP services it cannot be determined in this log, however this would warrant more follow up by the analyst.

Network Counter Measures: 3

Since the 46.5/16 net is running Snort with a standard rule set the scan was caught, and with further investigations a compromise was detected it could have been dealt with in a timely manner. The state of firewalls and router ACL's are not known.

Severity is calculated with the following formula:

$$\begin{aligned} &(\text{Criticality} + \text{Lethality}) - (\text{System Countermeasures} + \text{Network Countermeasures}) = \text{Severity} \\ &(3 + 4) - (3 + 3) = 1 \end{aligned}$$

### **2.1.9 Defensive Recommendations**

The first recommendation is to block this IP at the Firewall. It has been listed on the Dshield.org recommended Block List for previously reported activities.

Since there are so many vulnerabilities inherent to FTP the easiest way to secure the network would be to disable FTP on all hosts not requiring this service. The gateway router's [ACL's](#) could be updated to block FTP attempts to all hosts with an entry similar to this one for Cisco routers.

```
Deny tcp any any eq 21
```

If FTP services are required for a particular subnet, this ACL could be adjusted to reflect this.

```
permit tcp a.b.c.d 0.0.0.255 w.x.y.z eq 21
```

Also one has to ensure that all required security patches for the systems running the service are installed and up to date. Security audits and strong enforcement of security policies should also be employed

### **2.1.10 Multiple Choice Question**

Chose all that apply:

```
05:34:13.984488 62.153.209.202.21 > 46.5.251.75.21: SF
2087625956:2087625956(0) win 1028 (ttl 30, id 39426, bad cksum ce21!)
0x0000      4500 0028 9a02 0000 1e06 ce21 3e99 d1ca  E..(.....!>...
0x0010      2e05 fb4b 0015 0015 7c6e a4e4 7a20 d2fc  ...K....|n..z...
0x0020      5003 0404 0897 0000 0000 0000 0000      P.....
```

This packet is:

- a) [An attempt to by pass older IDS by using a SF.](#)
- b) A nullified code unavailability
- c) [An attempt to map this IP/port](#)
- c) An asynchronous logic-subsystem rejection
- d) The close of a legitimate ftp session

Answer: A,C

### **2.1.11 References**

[http://www.simovits.com/trojans/tr\\_data/y1386.html](http://www.simovits.com/trojans/tr_data/y1386.html)  
<http://www.cve.mitre.org/cve/>  
<http://isc.incidents.org>  
[http://www.whitehats.ca/main/publications/external\\_pubs/scanner\\_fingerprints/scanner\\_fingerprints.html](http://www.whitehats.ca/main/publications/external_pubs/scanner_fingerprints/scanner_fingerprints.html)  
<http://www.brianhouk.com/papers/acl-cisco-bhouk.htm#eacl>  
<http://www.ripe.net/>  
<http://www.psychoid.lam3rz.de/synscan1.6.tar.gz>  
<http://lists.jammed.com/incidents/2001/10/0104.html>  
<http://cert.uni-stuttgart.de/archive/intrusions/2002/07/msg00045.html>  
[rfc 791](#)  
Tcp/IP Illustrated, Vol 1 by W. Richard Stevens

## **2.1.12 Questions and Points Received From Posting**

I received several questions and a few points from my posting to the mailing list. I have included these in the above revised detect with the corrections and points covered.

Paul Bradley wrote on Oct 8<sup>th</sup>:

How do you know that none of the targeted hosts replied? What type of response would you expect from a targeted host listening on port 21?

Oliver Viitamaki wrote on Oct 8<sup>th</sup>:

How does FTP work? Is it normal for FTP communication to have both Source and destination ports as 21? Is there any possibility the sensor was blind to replies? Should there be any thought given to "blacklisting" this host? If yes, why, If no, why?

## 2.2 Detect 2 DDoS Attack on IRC Server

The date is Aug01 (MmmDD) timezone GMT and the filter is -nvX and host 80.68.100.52

```
15:00:28.270228 80.68.100.52.6667 > xxx.xxx.4.106.1024: R [tcp sum ok] 0:0(0) ack 1042425706 win 0 (ttl 109, id 48990, len 40)
0x0000 4500 0028 bf5e 0000 6d06 5206 5044 6434 E..(^..m.R.PDd4
0x0010 XXXX 046a 1a0b 0400 0000 0000 3e22 276a ..j.....>"j
0x0020 5014 0000 efcd 0000 0204 0218 0000 P.....
15:01:36.825486 80.68.100.52.6667 > xxx.xxx.160.10.1024: R [tcp sum ok] 0:0(0) ack 3282650728 win 0 (ttl 109, id 2493, len 40)
0x0000 4500 0028 09bd 0000 6d06 6c07 5044 6434 E..(....m.l.PDd4
0x0010 XXXX a00a 1a0b 0400 0000 0000 c3a9 4668 .....Fh
0x0020 5014 0000 afa7 0000 0101 080a 0001 P.....
15:01:42.672441 80.68.100.52.6667 > xxx.xxx.231.83.1024: R [tcp sum ok] 0:0(0) ack 3054875211 win 0 (ttl 109, id 63162, len 40)
0x0000 4500 0028 f6ba 0000 6d06 37c0 5044 6434 E..(....m.7.PDd4
0x0010 XXXX a00a 1a0b 0400 0000 0000 b615 b24b ...S.....K
0x0020 5014 0000 0a0f 0000 0234 3303 3139 P.....43.19
15:01:59.994405 80.68.100.52.6667 > xxx.xxx.178.53.1024: R [tcp sum ok] 0:0(0) ack 179545205 win 0 (ttl 109, id 25130, len 40)
0x0000 4500 0028 622a 0000 6d06 016f 5044 6434 E..(b*..m.o.PDd4
0x0010 XXXX b253 1a0b 0400 0000 0000 0ab3 a475 ...5.....u
0x0020 5014 0000 f865 0000 0204 05b4 0001 P...e.....
15:02:07.452612 80.68.100.52.6667 > xxx.xxx.1.99.1024: R [tcp sum ok] 0:0(0) ack 989449538 win 0 (ttl 109, id 26174, len 40)
0x0000 4500 0028 663e 0000 6d06 ae2d 5044 6434 E..(f>..m.-PDd4
0x0010 XXXX 0163 1a0b 0400 0000 0000 3af9 cd42 ...c.....B
0x0020 5014 0000 5025 0000 0101 080a 0009 P...P%.....
```

This activity continues, though the middle section of the log has been omitted.

```
15:25:46.499484 80.68.100.52.1024 > xxx.xxx.172.76.3072: R [tcp sum ok] 0:0(0) ack 4104439661 win 0 (ttl 109, id 35824, len 40)
0x0000 4500 0028 8bf0 0000 6d06 dd91 5044 6434 E..(....m..PDd4
0x0010 XXXX ac4c 0400 0c00 0000 0000 f4a4 c76d ...L.....m
0x0020 5014 0000 ff6f 0000 0204 05b4 0101 P...o.....
15:26:02.862695 80.68.100.52.1024 > xxx.xxx.208.58.1024: R [tcp sum ok] 0:0(0) ack 4011319823 win 0 (ttl 109, id 44699, len 40)
0x0000 4500 0028 ae9b 0000 6d06 96f8 5044 6434 E..(....m..PDd4
0x0010 XXXX d03a 0400 0400 0000 0000 ef17 e20f .....
0x0020 5014 0000 ce6c 0000 0204 0200 048d P...l.....
15:26:06.170230 80.68.100.52.1024 > xxx.xxx.195.25.3072: R [tcp sum ok] 0:0(0) ack 1397259577 win 0 (ttl 109, id 12668, len 40)
0x0000 4500 0028 317c 0000 6d06 2139 5044 6434 E..(1|..m.!9PDd4
0x0010 XXXX c319 0400 0c00 0000 0000 5348 7d39 .....SH)9
0x0020 5014 0000 d433 0000 0101 080a 048d P...3.....
15:26:38.099620 80.68.100.52.1024 > xxx.xxx.219.30.3072: R [tcp sum ok] 0:0(0) ack 1194678285 win 0 (ttl 109, id 28360, len 40)
0x0000 4500 0028 6ec8 0000 6d06 cbe7 5044 6434 E..(n...m..PDd4
0x0010 XXXX db1e 0400 0c00 0000 0000 4735 580d .....G5X.
0x0020 5014 0000 ed6d 0000 0204 05b4 0101 P...m.....
15:26:40.654493 80.68.100.52.1024 > xxx.xxx.123.19.3072: R [tcp sum ok] 0:0(0) ack 1193457197 win 0 (ttl 109, id 54712, len 40)
0x0000 4500 0028 d5b8 0000 6d06 c502 5044 6434 E..(....m..PDd4
0x0010 XXXX 7b13 0400 0c00 0000 0000 4722 b62d ..{.....G".-
0x0020 5014 0000 ef6b 0000 0101 080a 048d P...k.....
```

### 2.2.1 Source of Trace

This detect was gathered from my employers class B network. The sensor that gathered the log is located between the firewall and the Border Router. All logs have been sanitized for security reasons.

### **2.2.2 Detect Generated By**

This detect was originally found on our SHADOW IDS sensor, ver 1.6, NSWC version, from the following two entries:

```
15:11:22.256649 80.68.100.52.6667 > xxx.xxx.14.0.1024: R 0:0(0) ack 2403830560 win 0
15:25:05.367832 80.68.100.52.6667 > xxx.xxx.17.0.1024: R 0:0(0) ack 3460519707 win 0
```

TCPdump was used with the filter “-nvX ip and host 80.68.100.52 and port 1024” to generate this log.

### **2.2.3 Possibility the Address is Spoofed**

The likelihood of this source address being spoofed is very low. It appears to be either an inverse mapping attempt, which would likely require a legitimate address, or it is a response to our netblock addresses being spoofed for a DoS attack on the source host. The address resolves through [whois.ripe.net](http://whois.ripe.net) to:

```
inetnum: 80.68.100.0 - 80.68.100.127
netname: INTEGRA-SWEDEN
descr: Enterprise Systems
country: SE
admin-c: LT2404-RIPE
tech-c: TP6125-RIPE
tech-c: TF5334-RIPE
status: ASSIGNED PA
notify: lasse.tegenborg@integra-sweden.com
notify: tech@integra-sweden.com
mnt-by: INTEGRA-SET
changed: tony.fritzell@integra-sweden.com 20010828
source: RIPE
```

### **2.2.4 Description of Attack**

Our network received many packets over a twenty-six minute period with the reset and ack flags set, destined to many seemingly random IP's on our class B network, to destination ports of 1024 and 3072. The default port of 6667 shows the source is probably an IRC server. There is a possibility that an IRC server has been used to map our network, or it could be traffic for a Trojan that communicates or receives instructions over IRC.

There are multiple [CVE's](#) listed with IRC vulnerabilities.

### **2.2.5 Attack Mechanism**

An attack using a set TCP Syn or Ack flag, with randomly generated source IP's is an attempt to consume server/cpu resources and/or network resources of the target host or network, and would likely generate reset / ack responses to the source host(s), in this case our spoofed class B network addresses.

No stimulus for this detect could be found on our network, therefore it would appear that our address block was spoofed in some form of DDoS attack on the source host (80.68.100.52) and the response to that attack is what showed up on our sensors.

Since the packet id numbers from this server are advancing at a very high rate of speed in relation to time one can conclude that the server is very busy. This also leads credence to the theory of some form of DDoS attack with several attackers acting in concert.

Both IRC Server and various trojans including DarkFTP, EGO and Subseven use the tcp port of 6667 according to [treachery.net](http://treachery.net). Several trojans including Jade, Latinus, Netspy and Rat, as well as the K Display Manager also use the other tcp source port of 1024. Treachery Unlimited lists Port 3072 as being the ContinStor Monitor Port. This detect does not seem to follow any behaviour for these Trojans that I can find, however many [Trojans](#) do communicate through IRC. I could not find any available tool or script that would use these the ports of 1024 or 3072, nor does it seem to be widely used.

As for original attack on IP 80.68.100.52, it appears to be a concerted effort to take down an IRC server through a DDoS attack. Since the only correlation I could find was dated Dec 2000 I believe that the tool or script used appears to be an older one and not well used.

### **2.2.6 Correlation**

No correlating data could be found within our network other than the SHADOW log and subsequent tcpdump search. Searching the Internet I did find some correlation in late Dec 2000, Conor McGrath posted a question on [SecurityFocus](#) on scans of ports 1024 and 3072. Following the thread it lead to the [answer](#) that the network IP's in question were being used for a DDoS attack on DALnet IRC servers.

I contacted the ISP in Sweden by e-mail (Appendix A) and received a reply from a Mr. Andreas Odman from [Lunarworks.se](http://Lunarworks.se).

I sent:

.....  
As we can find no stimulus from our network to solicit this traffic I am sure our class B address block was spoofed in a DDoS attack against your IP. I was hoping that you might have some information about this IP that you could share with me. Since the source port of the above log is 6667 (Internet Relay Chat) I was hoping if you could tell me if this was indeed using IRC, or if not, do you have any idea what the attacker(s) might be trying to do? I have found reference to this type of attack using these ports as far back as Dec of 2000 as a DDoS to take down IRC servers. Any logs, correlation, or answers that you could send me would be greatly appreciated,

His Reply:

Yes, you are correct. 80.68.100.52 is, or now was, the ip-adress for our irc-server. It now resides at 80.xxx.xxx.xxx (irc.lunarstorm.se). We are aware of this problem and in our best effort trying to fend this off. Our irc-community is small, and only serves as a small part of our larger community www.lunarstorm.se. We don't have any logs for this incident.

This e-mail reply from the sys admin confirms that this was a DDoS attack against an IRC server using our net block addresses as a spoofed source.

### **2.2.7 Evidence of Active Targeting**

At first glance it appeared that we were targeted with some form of inverse mapping attempt, but on further analysis I now know that our net block addresses were spoofed and this is not an active targeting of our networks.

### **2.2.8 Severity**

Criticality: 1

This was not an attack against my network, just a reflection of an attack on a third party host. If more evidence was found to suggest a Trojan was communicating this would have been much higher.

Lethality: 2

No damage would have been done in this example, as our net block was only spoofed and not the intended victim. This could also have been used as a distraction from more devious and stealthy recon or serious attack. Also the victim could "blame" my employer as the originator of the attack, which may have significant consequences if they were in aggressive in nature.

System Countermeasures: 4

All systems have the latest vendor patches applied. Configuration management of internal host is tight and systems are base lined.

Network Countermeasures: 5

Our hosts sit behind firewalls with restrictive gateways and strong ACL's.

(Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity

( 1 + 2 ) – ( 4 + 5 ) = - 6

### **2.2.9 Defensive Recommendations**

Since this could have been an attempt to inversely map the network, one has to ensure that out bound ICMP [host unreachable messages](#) are blocked by the router if not replying to an ARP. To ensure that Trojans are not installed the system administrators should keep anti-virus software up to date with the most recent signature files. The closing of port 6667 if IRC servers are not running would help tighten down security as well.

### **1.2.10 Multiple Choice Question**

Numerous packets that have the TCP flags RST/ACK set and are not solicited by your networks are most likely: (pick all that apply)

- a) [an attempt to inversely map your network](#)
- b) an attempted buffer overflow attack on your server
- c) [the result of your address being spoofed in an attack on someone else](#)
- d) [an attempt to hide stealthy attacks from overworked IDS personnel](#)
- e) a misconfigured management console

answer: a c d

## **2.2.11 References**

<http://www.ripe.net/>

<http://www.cve.mitre.org/>

<http://www.treachery.net/tools/ports/lookup.cgi>

<http://cert.uni-stuttgart.de/archive/incidents/2000/12/msg00156.html>

<http://cert.uni-stuttgart.de/archive/incidents/2000/12/msg00165.html>

<http://www.lunarstorm.se/>

<http://www.sans.org/newlook/resources/IDFAQ/traffic.htm>

<http://www.ash.udel.edu/ash/teacher/AUP.html>

E-mail to Lunarworks in Sweden (Appendix A)

## 2.3 Detect 3 HTTP Connect Attempt

### Shadow Log

```
61.172.246.78 > xxx.xxx.210.203
00:00:19.628022 61.172.246.78.58416 > xxx.xxx.210.203.3128: S 1908257516:1908257516(0) win 1024
00:00:22.828475 61.172.246.78.1635 > xxx.xxx.210.203.1080: S 548921344:548921344(0) win 1024
00:00:33.993039 61.172.246.78.578 > xxx.xxx.213.57.1080: S 557042138:557042138(0) win 1024
00:00:37.908700 61.172.246.78.5624 > xxx.xxx.213.57.3128: S 1439127734:1439127734(0) win 1024
00:01:50.731521 61.172.246.78.57807 > xxx.xxx.210.179.3128: S 1069243264:1069243264(0) win 1024
00:01:54.143283 61.172.246.78.60759 > xxx.xxx.210.179.1080: S 115436860:115436860(0) win 1024
00:04:10.122517 61.172.246.78.29907 > xxx.xxx.210.78.3128: S 242656534:242656534(0) win 1024
```

### Ngrep Output

```
T 2002/09/01 00:00:19.628022 61.172.246.78:58416 -> xxx.xxx.210.203:3128 [S]
T 2002/09/01 00:00:22.828475 61.172.246.78:1635 -> xxx.xxx.210.203:1080 [S]
T 2002/09/01 00:00:31.858249 61.172.246.78:37366 -> xxx.xxx.213.57:80 [S]
T 2002/09/01 00:00:33.993039 61.172.246.78:578 -> xxx.xxx.213.57:1080 [S]
T 2002/09/01 00:00:37.908700 61.172.246.78:5624 -> xxx.xxx.213.57:3128 [S]
T 2002/09/01 00:00:39.831754 61.172.246.78:64177 -> xxx.xxx.213.57:8080 [S]
T 2002/09/01 00:01:47.572889 61.172.246.78:43877 -> xxx.xxx.210.179:80 [S]
T 2002/09/01 00:01:49.151481 61.172.246.78:57714 -> xxx.xxx.210.179:8080 [S]
T 2002/09/01 00:01:50.731521 61.172.246.78:57807 -> xxx.xxx.210.179:3128 [S]
T 2002/09/01 00:01:54.143283 61.172.246.78:60759 -> xxx.xxx.210.179:1080 [S]
T 2002/09/01 00:04:06.815307 61.172.246.78:12988 -> xxx.xxx.210.78:80 [S]
T 2002/09/01 00:04:08.442125 61.172.246.78:18292 -> xxx.xxx.210.78:8080 [S]
T 2002/09/01 00:04:10.122517 61.172.246.78:29907 -> xxx.xxx.210.78:3128 [S]
T 2002/09/01 00:04:10.623935 61.172.246.78:19367 -> xxx.xxx.172.207:80 [S]
T 2002/09/01 00:04:12.597500 61.172.246.78:29116 -> xxx.xxx.172.207:1080 [S]
.....
T 2002/09/01 18:43:56.430161 61.172.246.78:47790 -> xxx.xxx.16.40:80 [S]
T 2002/09/01 18:43:56.506714 xxx.xxx.16.40:80 -> 61.172.246.78:47790 [AS]
T 2002/09/01 18:43:56.725127 61.172.246.78:47790 -> xxx.xxx.16.40:80 [AP]
CONNECT 194.67.26.89:80 HTTP/1.0....
T 2002/09/01 18:44:26.808851 xxx.xxx.16.40:80 -> 61.172.246.78:47790 [AF]
T 2002/09/01 18:44:27.027277 61.172.246.78:47790 -> xxx.xxx.16.40:80 [AF]
```

### Tcpdump

The date is Sep01 (MmmDD) timezone GMT and the filter is -nvXs 0 ip and host 61.172.246.78

```
00:01:47.572889 61.172.246.78.43877 > xxx.xxx.210.179.80: S [tcp sum
ok] 150626275:150626275(0) win 1024 (ttl 44, id 35526, len 40)
0x0000      4500 0028 8ac6 0000 2c06 79d2 3dac f64e E..(.....,y.=..N
0x0010      xxxx d2b3 ab65 0050 08fa 5fe3 08fa 5fe3 .....e.P._._._.
0x0020      5002 0400 a43a 0000 0000 0000 0000 0000 P.....:.....
00:01:49.151481 61.172.246.78.57714 > xxx.xxx.210.179.8080: S [tcp sum
ok] 1885021016:1885021016(0) win 1024 (ttl 44, id 41178, len 40)
0x0000      4500 0028 a0da 0000 2c06 63be 3dac f64e E..(.....,c.=..N
0x0010      xxxx d2b3 e172 1f90 705b 2358 705b 2358 .....r..p[#Xp[#X
0x0020      5002 0400 f940 0000 0000 0000 0000 0000 P.....@.....
00:01:50.731521 61.172.246.78.57807 > xxx.xxx.210.179.3128: S [tcp sum
ok] 1069243264:1069243264(0) win 1024 (ttl 44, id 468, len 40)
0x0000      4500 0028 01d4 0000 2c06 02c5 3dac f64e E..(.....,...=..N
0x0010      xxxx d2b3 e1cf 0c38 3fbb 5b80 3fbb 5b80 .....8?.[?.[.
0x0020      5002 0400 fd2b 0000 0000 0000 0000 0000 P.....+.....
00:01:54.143283 61.172.246.78.60759 > xxx.xxx.210.179.1080: S [tcp sum
ok] 115436860:115436860(0) win 1024 (ttl 44, id 51312, len 40)
0x0000      4500 0028 c870 0000 2c06 3c28 3dac f64e E..(.p.,.<(=..N
0x0010      xxxx d2b3 ed57 0438 06e1 6d3c 06e1 6d3c .....W.8..m<..m<
0x0020      5002 0400 47e0 0000 0000 0000 0000 0000 P...G.....
```

Logs shortened for brevity.

```
18:43:56.506714 xxx.xxx.16.40.80 > 61.172.246.78.47790: S [tcp sum ok]
4104715709:4104715709(0) ack 3513251763 win 8192 <mss 512,nop,wscale 0>
(ttl 61, id 54306, len 48)
0x0000      4500 0030 d422 0000 3d06 elf9 xxxx 1028 E..0."..=.....(
0x0010      3dac f64e 0050 baae f4a8 fdbd d167 f7b3 =..N.P.....g..
0x0020      7012 2000 2996 0000 0204 0200 0103 0300 p.....).....
18:43:56.725127 61.172.246.78.47790 > xxx.xxx.16.40.80: P [tcp sum ok]
1:37(36) ack 1 win 5840 (DF) (ttl 43, id 9540, len 76)
0x0000      4500 004c 2544 4000 2b06 62bc 3dac f64e E..L%D@.+..b.=..N
0x0010      xxxx 1028 baae 0050 d167 f7b3 f4a8 fdbe ...(...P.g.....
0x0020      5018 16d0 78fe 0000 434f 4e4e 4543 5420 P...x...CONNECT.
0x0030      3139 342e 3637 2e32 362e 3839 3a38 3020 194.67.26.89:80.
0x0040      4854 5450 2f31 2e30 0d0a 0d0a HTTP/1.0....
18:44:26.808851 xxx.xxx.16.40.80 > 61.172.246.78.47790: F [tcp sum ok]
1:1(0) ack 37 win 8192 (ttl 61, id 54312, len 40)
0x0000      4500 0028 d428 0000 3d06 elfb xxxx 1028 E..(.(...=.....(
0x0010      3dac f64e 0050 baae f4a8 fdbe d167 f7d7 =..N.P.....g..
0x0020      5011 2000 5181 0000 0000 0000 0000 0000 P...Q.....
18:44:27.027277 61.172.246.78.47790 > xxx.xxx.16.40.80: F [tcp sum ok]
37:37(0) ack 2 win 5840 (DF) (ttl 43, id 9541, len 40)
0x0000      4500 0028 2545 4000 2b06 62df 3dac f64e E..(%E@.+..b.=..N
0x0010      xxxx 1028 baae 0050 d167 f7d7 f4a8 fdbf ...(...P.g.....
0x0020      5011 16d0 5ab0 0000 0000 0000 0000 0000 P...Z.....
```

### **2.3.1 Source of Trace**

This detect was gathered from my employer's class B network. The sensor that gathered the logs is located between the firewall and the DMZ. All logs have been sanitized for security reasons.

### **2.3.2 Detect Generated By**

This detect was generated from a SHADOW IDS sensor, ver 1.6 , NSWC version. I then ran the raw logs through ngrep ver 1.40.1 revision 1.23 to sort through the logs for Sept 1 through Sept 3 2002. Upon finding a rather large scan of the network I then ran the raw logs through tcpdump ver 3.7.1 for final correlation of the material found previously. Tcpdump was started using the following script:

```
tcpdump -i eth1 -s 120 -w - -F /usr/local/logger/sensor/gmt.filter
```

### **2.3.3 Probability the Source Address was Spoofed**

The source host is trying to establish a connection via a three way hand shake with randomly selected IP's from our class B network, therefore the likelihood that the source address has been spoofed is very low to nil. Our source address resolves through [APNIC](#) to:

```
inetnum: 61.172.244.0 - 61.172.255.255
netname: SHTELE-XINCHAN-IDC
descr: Shanghai Information Industrial Co.
country: CN
admin-c: ZY108-AP
tech-c: JZ5-AP
mnt-by: MAINT-CHINANET-SH
mnt-lower: MAINT-CN-SHTELE-XINCHAN
changed: ip-admin@mail.online.sh.cn 20020619
status: ALLOCATED PORTABLE
source: APNIC
```

while the embedded HTTP Connect IP resolves through [Ripe.net](#) to:

```
inetnum: 194.67.26.0 - 194.67.26.255
netname: STNET26
descr: Teleross
descr: Moscow, Russia
country: RU
admin-c: TELE1-RIPE
tech-c: TELE1-RIPE
```

status: ASSIGNED PA  
notify: noc@sovam.com  
mnt-by: AS3216-MNT  
changed: iga@sovam.com 20020527  
source: RIPE

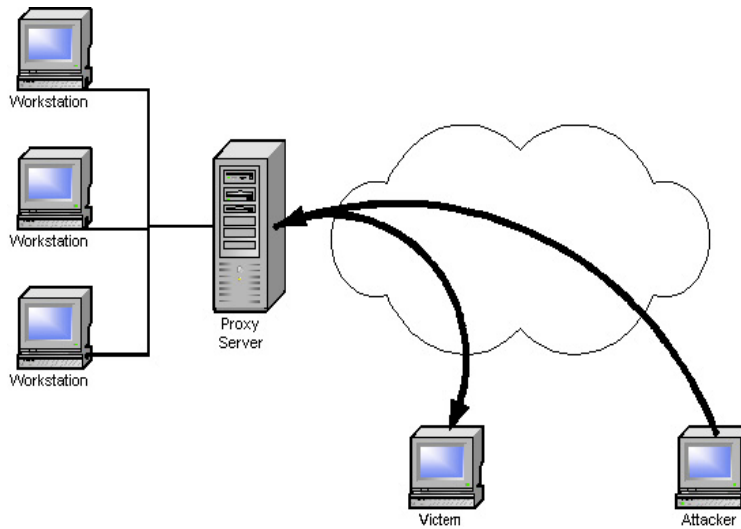
The CERT Vulnerability Note [VU#150227](#) addresses this problem, though no CVE has been issued on this.

### **2.3.4 Description of the Attack**

This is an attempt to connect to a default install of a web proxy server on ports 80, 1080, 3128, and 8080 by IP 61.172.246.78. If the server supports the HTTP CONNECT method then the source can arbitrarily connect to either an internal IP or a third party external IP, by circumventing the normal application layer function of the proxy server. In this case the attacker was able to connect to several proxy servers in our network but did not succeed in CONNECTing to the embedded third party address.

### **2.3.5 Attack Mechanism**

The HTTP request method is an instruction to indicate the purpose of an http request. The CONNECT method is a tunnelling mechanism that resides lower than other http methods and allows the proxy to forward the transaction to either an internal IP address or a third party external address. The initiator of the CONNECT requires the completion of the three handshake with a tcp based proxy server with an insecure default install. In order for this transaction to be completed the proxy server must first initiate and complete a three-way handshake with the specified CONNECT IP and then acts as a tunnel between the two hosts regardless of what data is transferred. Thus, like a regular proxy connection the destination host would only see the proxy address.



HTTP CONNECT to an external host through network Proxy

In this detect the source IP, 61.172.246.78 is attempting to use this CONNECT request method to connect to a remote host through our networks proxy servers. While no redirection takes place, the destination host and port to connect to are listed within the payload of the packet:

```
T 2002/09/01 18:43:56.725127 61.172.246.78:47790 -> xxx.xxx.16.40:80 [AP]
CONNECT 194.67.26.89:80 HTTP/1.0....
```

This IP, [194.67.26.89](http://194.67.26.89), appears to be a legitimate Russian language news site. Why this site has been targeted can only be conjecture as no CONNECT was actually made. This is probably an attempt to either generate revenue through banner ads or click throughs, or an attempt to increase the number of hits to this web site.

### 2.3.6 Correlation

Numerous e-mails posted to the Incidents.Org mailing list noted the same traffic on other networks. Donna MacLeod [dmacleod@fmco.com] posted on 27 Aug 2002:

Anyone else notice an increase in scans for proxy servers? We rarely get them, and last night our network was scanned from a few sources looking for one. Here's a log excerpt:

```
[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
08/27-00:57:49.265539 61.172.246.78:65376 -> xxx.yyy.zzz.25:1080
TCP TTL:44 TOS:0x0 ID:39446 IpLen:20 DgmLen:40
*****S* Seq: 0x7FA4D32E Ack: 0x7FA4D32E Win: 0x400 TcpLen: 20
```

```
[**] [1:620:2] SCAN Proxy (8080) attempt [**]
08/27-00:57:51.823222 61.172.246.78:63234 -> xxx.yyy.zzz.25:8080
TCP TTL:44 TOS:0x0 ID:13388 IpLen:20 DgmLen:40
*****S* Seq: 0x5B4A3B00 Ack: 0x5B4A3B00 Win: 0x400 TcpLen: 20
```

```
[**] [1:618:2] SCAN Squid Proxy attempt [**]
08/27-00:57:54.032623 61.172.246.78:45875 -> xxx.yyy.zzz.25:3128
TCP TTL:44 TOS:0x0 ID:41192 IpLen:20 DgmLen:40
*****S* Seq: 0x32D13C6E Ack: 0x32D13C6E Win: 0x400 TcpLen: 20
```

Rodrigo Goya [lucent@securenet.com.mx] replied also on 27 Aug 2002:

If it's of any use, I've seen that IP (61.172.246.78) on our IDS scanning several IPs, the packets are stopped when they reach the firewall and some of those IPs are unused. The offending IP 61.172.246.78 is sending out probes in pairs (2 seconds apart) for port 1080 and 8080. And jumps from IP to IP on apparently random minutes. Here is a log summary of this activity only for August 2002-08-27 from 00:00 until 12:00.

```
[2002-08-27 00:04:13] 61.172.246.78:26291 -> XXX.YYY.ZZZ.43:1080
[2002-08-27 00:04:15] 61.172.246.78:33386 -> XXX.YYY.ZZZ.43:8080
[2002-08-27 00:08:36] 61.172.246.78:38588 -> XXX.YYY.ZZZ.147:1080
[2002-08-27 00:08:38] 61.172.246.78:7424 -> XXX.YYY.ZZZ.147:8080
[2002-08-27 00:17:30] 61.172.246.78:46623 -> XXX.YYY.ZZZ.227:1080
[2002-08-27 00:17:32] 61.172.246.78:3599 -> XXX.YYY.ZZZ.227:8080
```

There are several more in this thread that all show attempts to connect to proxy servers. A search on [dshield.org](http://dshield.org) returned the following (figure 2.3.1) that shows this IP has been very busy during this time period.

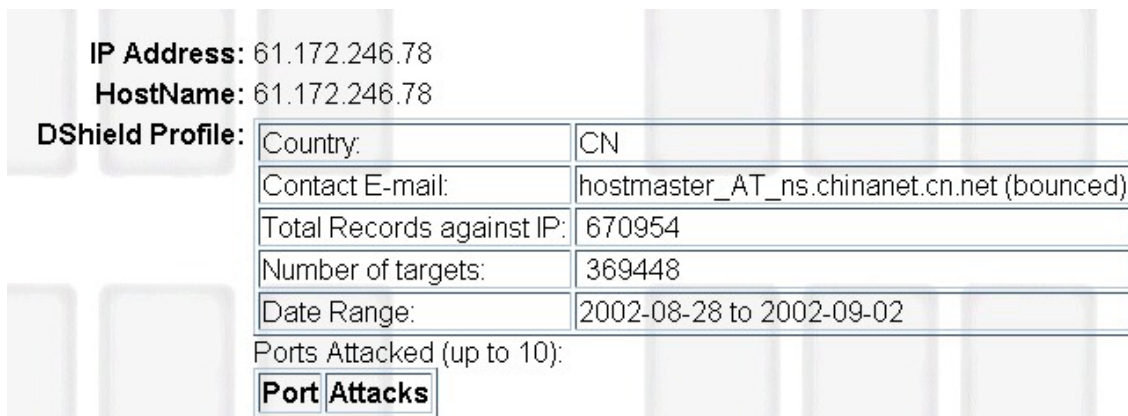


Figure 2.3.1

Dshield also shows that the ports [1080](#), (figure 2.3.2) [3128](#),(figure 2.3.3) and [8080](#) (figure 2.3.4) are targeted quite actively during the same time period.

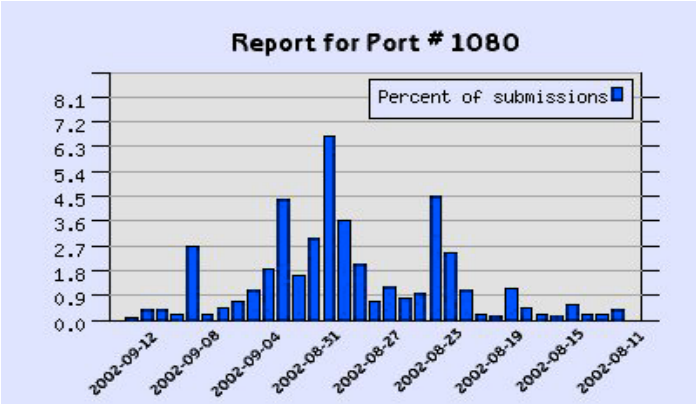


Figure 2.3.2

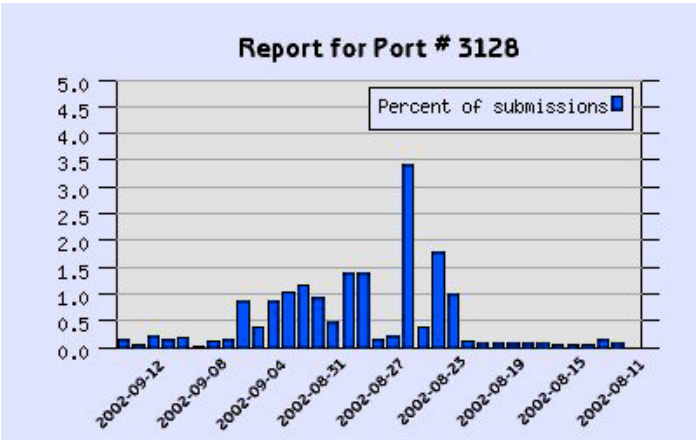


Figure 2.3.3

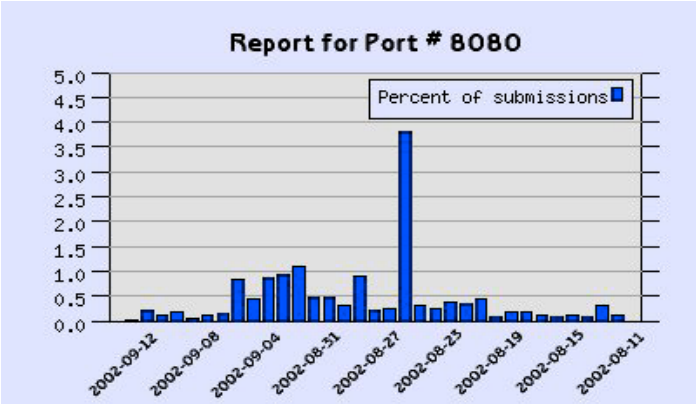


Figure 2.3.4

### **2.3.7 Evidence of Active Targeting**

This is an active attempt to target our entire class B network, however it is also part of a rather wide-ranging scan of many networks across the Internet.

### **2.3.8 Severity**

Criticality: 4

This was an active attempt to use a known vulnerability in HTTP to remotely connect to either an internal or third party host.

Lethality: 4

This exploit could have created a tcp tunnel through the proxy if a successful connection had been established. Once the connection is established there is no control over what type of information is passed. Since the source IP would be listed as our proxy it could be detrimental to my organization if any illegal event or occurrence was made through the connection.

System Countermeasures: 4

All hosts have the latest vendor patches installed to correct the HTTP CONNECT vulnerability. Configuration management of internal hosts is tight and all systems are base lined.

Network Countermeasures: 5

The network sits behind firewalls with restrictive gateways and strong ACL's

(Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity

$$( 4 + 4 ) – ( 4 + 5 ) = -1$$

### **2.3.9 Defensive Recommendations**

Ensuring all systems affected have the appropriate vendor patch installed as listed in the CERT Vulnerability Note [VU#150227](#) would be a first step. Also enforcing existing policy relating to server installation and following good security practice with passwords and [hardening](#) of these servers would reduce the security risk. Scheduled audits of system configurations should be enacted to ensure proper configurations are adhered to.

### **2.3.10 Multiple Choice Question**

```
18:43:56.725127 61.172.246.78.47790 > xxx.xxx.16.40.80: P [tcp sum ok]
1:37(36) ack 1 win 5840 (DF) (ttl 43, id 9540, len 76)
0x0000      4500 004c 2544 4000 2b06 62bc 3dac f64e  E..L%D@.+b.=.N
0x0010      xxxx 1028 baae 0050 d167 f7b3 f4a8 fdbe  ...(...P.g.....
0x0020      5018 16d0 78fe 0000 434f 4e4e 4543 5420  P...x...CONNECT.
0x0030      3139 342e 3637 2e32 362e 3839 3a38 3020  194.67.26.89:80.
0x0040      4854 5450 2f31 2e30 0d0a 0d0a                HTTP/1.0....
```

The above packet is or shows:

- a) the actual IP address of an attacker spoofing the source address
- b) a third party connecting to a three way connection
- c) an attempted exploitation of the HTTP CONNECT vulnerability
- d) an attacker sending a target IP to his controlled systems for a DDoS attack

Answer: c

### **2.3.11 References**

<http://mtip.net/aware/MarkLachnietChecklist.pdf>

<http://www.kb.cert.org/vuls/id/150227>

<http://www.ripe.net/>

<http://www.apnic.net/apnic-bin/whois2.pl>

<http://www.dshield.org>

<http://www.web-cache.com/Writings/Internet-Drafts/draft-luotonen-web-proxy-tunneling-01.txt>

<http://www.ietf.org/rfc/rfc2817.txt>

<http://www.ietf.org/rfc/rfc2616.txt>

<http://www.ietf.org/rfc/rfc2068.txt>

### **3. Analyze This!**

I have been asked to conduct traffic analysis of the University over a five-day period covering the 13<sup>th</sup> to 17<sup>th</sup> of September 2002. The University appears to be set up over the entire MY.NET/16 network, with a possible 65353 IP addresses available. While not every host is noted in the alert or scan logs analyzed, thirty-eight separate class C subnets have been seen in these logs. There are no scan or alert logs that track traffic from MY.NET>MY.NET which leads to the conclusion that the Snort sensor is located outside the firewall and logs only traffic incoming to or outgoing from the University.

There are several areas of concern that are addressed through out this report, including possible Trojans, port scanning, suspected compromised TFTP servers, and some systems that may be susceptible to compromise. The University has, as well, a huge amount of Peer-to-Peer traffic and Internet gaming traffic occurring.

It would be extremely helpful in future to include the following information with the logs for more detailed and accurate analysis:

1. A complete layout of the University net, with both Infrastructure and Residential networks, including IP addresses and the major services running on each.
2. An accurate mapping of locations of IDS sensors.
3. A complete rule set for the IDS, including locally written rules, and some identification of why these local rules were implemented.
4. A list of authorized web, app, and ftp servers with the software load versions installed.
5. Access to the Firewall/Gateway Router logs.
6. Supplying all logs in binary format.

### **3.1 Log Files**

The following is a list of the downloaded files used as a basis for this report. All file sizes shown are in bytes in a zipped format. (Table 3.1.1)

<b>LOG FILES</b>					
<b>SCANS</b>		<b>ALERTS</b>		<b>OOS</b>	
scans.020913	1,654,768	alerts.020913	542,580	oos_Jun.11	691
scans.020914	2,472,222	alerts.020914	666,077	oos_Jun.12	195
scans.020915	2,948,728	alerts.020915	676,285	oos_Jun.13	901
scans.020916	1,940,788	alerts.020916	608,510	oos_Jun.14	726
scans.020917	1,479,115	alerts.020917	385,769	oos_Jun.15	295
total	10,495,621	total	2,879,221	total	2808

Table 3.1.1

The logs analyzed for this report represent 97.1MB of scan logs and 34MB of alert logs over the five-day period of 13th to 17th September 2002. This translates into 1,569,913 scan entries, and 118,607 alert entries to import and analyze. The OOS logs do not correspond exactly with the dates of the other logs but are overlapping for three days worth of logging and are significantly smaller in size.

### **3.2 Methodology**

Using a Windows platform for this project presented a challenge since most of the tools usually used to help parse files are built for UNIX. The first step was to find and download the necessary tools ported to a Win32 format, including [ActivePerl 5.6.1.633](#), and [SED for windows](#). The next step was to run SED on the alert files to change the MY.NET sanitation to 10.10 for ease of handling using the following command:

```
C :> sed s/MY.NET/10.10/g filename > newfile.txt
```

The perl scripts were then run to convert the downloaded Alert and Scan logs into Comma Separated Value format. The results were directed to a text file as shown:

```
C :> perl \perl\script1 filename > newfilealert.txt
```

The newfiles.txt was then imported to an Access Database for future mining of information.

The scan files were treated in a similar fashion using a perl script to convert them into a CSV format as well using:

```
C :>perl \perl\script2 filename > newfilescan.txt
```

This CSV format scan file was then imported into an Access Database.

These scripts were borrowed from Harry Halladay's GCIA practical (0516) and worked as advertised.

### **3.3 Scan Logs**

Starting with the scan logs, which were divided into the categories of those originating from inside MY.NET, and those targeting MY.NET, it was noticed there was a significant difference in the types of scanning logged between the internal and external hosts. The internal traffic is predominantly user oriented, while the external traffic appears to be various scans for vulnerabilities, mapping attempts, or virus based.

There were a total of 1,238,735 scans originating from MY.NET. Of the inside scanners, 683681, or 53%, are Peer to Peer (P2P) applications, as per Figure 3.3.1, including WinMX, Gnutella, KaZaa, Blubster, and Bearshare. As expected, gaming traffic was evident, originating from the inside accounting for 465573 scans logged, or 34% of total traffic. In all 30 hosts from MY.NET were involved in P2P file sharing and 8 hosts are involved in online gaming.

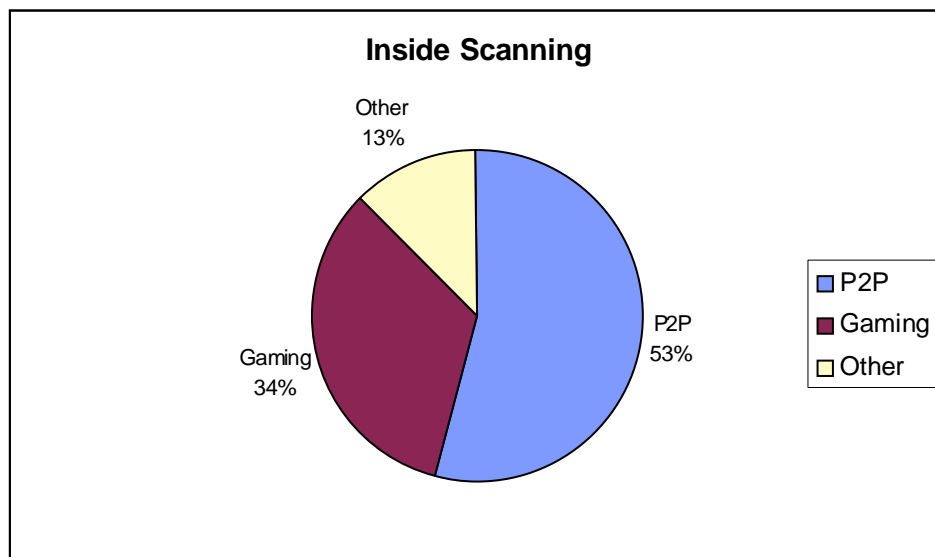


Figure 3.3.1

The peer-to-peer traffic is broken down as per Figure 3.3.2 and table 3.3.1.

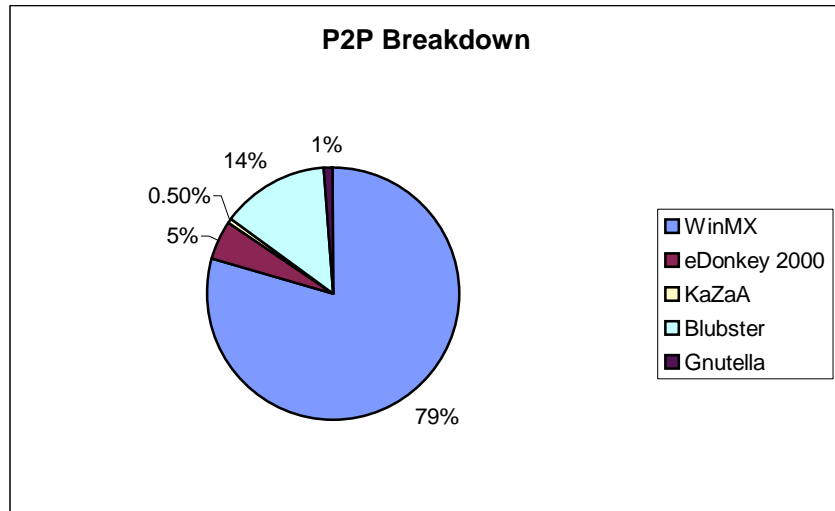


Figure 3.3.2

P2P software	Number of Users	Number of Scans
WinMX	9	545961
eDonkey 2000	1	33928
Blubster	1	96241
Gnutella	15	7030
KaZaA	4	521

Table 3.3.1

The online gaming traffic follows a similar pattern with few users generating a lot of traffic as seen in Figure 3.3.3 and table 3.3.2.

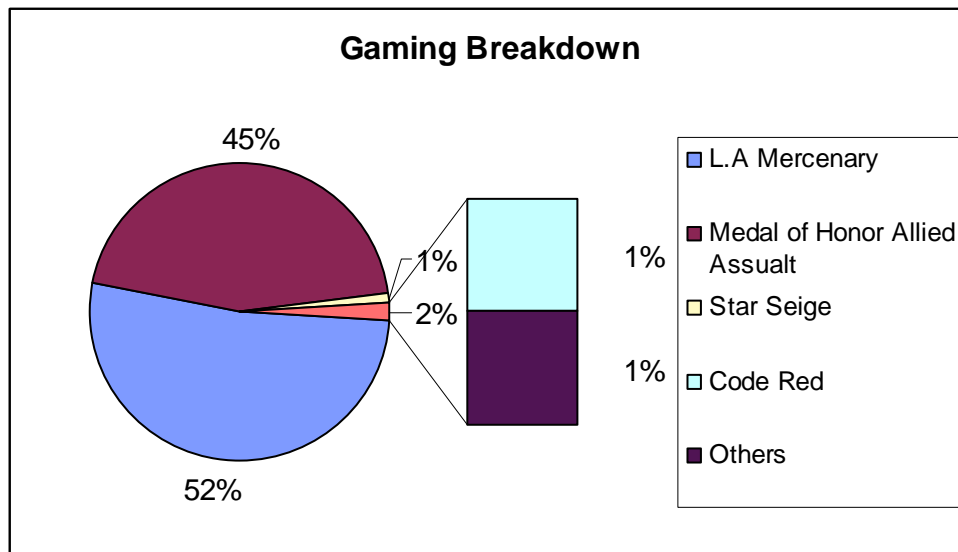


Figure 3.3.3

Game	Number of Users	Number of Scans
L.A Mercenary	1	241238
Medal of Honor Allied Assault	2	219276
Star Siege	2	2696
Code Red	3	1642
Others	4	721

Table 3.3.2

By reviewing this traffic it becomes quite apparent that very few hosts using P2P software or playing online games can generate a huge amount of logged traffic. In total 1149254 scans, or 92% of total internal scans logged, can be attributed to this type of traffic. This list, as per table 3.3.3, of the top ten inside scanners shows how these types of software can generate a large volume of logs.

Internal Scanners	Totals	Port
MY.NET.83.146	495220	6257 WinMX
All Other IP's	376799	
MY.NET.87.44	256482	27021 LA Mercenary
MY.NET.70.207	165935	12300 MOHAA
MY.NET.70.180	96283	41170 Blubster
MY.NET.82.2	53413	12300 MOHAA
MY.NET.111.215	40237	3296 Rib License Manager
MY.NET.114.45	33634	6257 WinMX
MY.NET.137.7	29584	53 DNS
MY.NET.87.50	8135	888 Flex LM
MY.NET.168.152	7212	6257 WinMX

Table 3.3.3

Once the Peer to Peer, gaming, and what appears to be normal web traffic is removed from the logs the 'real' traffic is left behind showing several interesting events and services. There is one host in MY.NET that is almost certainly infected with NIMDA, and another two hosts appears to be performing an Nmap type port scans. This left over traffic breaks down as per Table 3.3.4.

<b>IP</b>	<b>Event</b>
MY.NET.1.2	DNS
MY.NET.100.158	DNS
MY.NET.111.215	Flex LM - a web-based software for managing flexible software licensing over an enterprise.
MY.NET.115.115	NIMDA
MY.NET.139.51	Sun ONE Application Server 7 on Windows and UNIX platforms.
MY.NET.153.153	User Management Service
MY.NET.153.157	DNS 2 Go - Domain Name System allowing computer access through a domain name associated with the assigned IP address.
MY.NET.153.168	Web Cache
MY.NET.154.27	Sometimes RPC 5
MY.NET.163.132	Port scan of host 128.183.252.83
MY.NET.179.78	Port scan of host 66.80.148.137
MY.NET.6.49	Mail Server
MY.NET.84.209	Wnn-6 Taiwanese Input – a server for the conversion of Chinese characters to text.
MY.NET.87.44	Quick Time Server Admin
MY.NET.87.50	Web based MS Access database interface
MY.NET.88.148	ICAP Server - Squid Proxy Server
MY.NET.89.241	Express Pay - allows secure payment by credit card

Table 3.3.4

The scanning of MY.NET by outside sources follows a predictable format with port 80, HTTP, being the most numerous as illustrated in Figure 3.3.4. While some of this traffic is legitimate web traffic, there are numerous worm type scans.

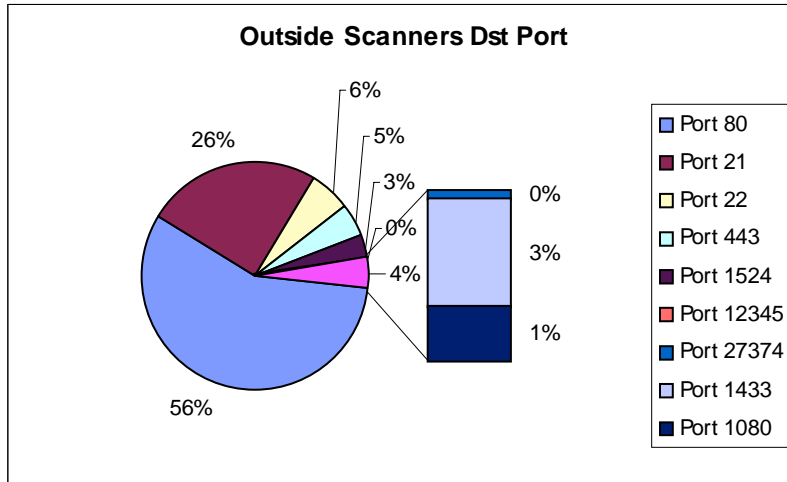


Figure 3.3.4

Several large scans against the FTP port, port 21, also accounted for a very large number of events in the log files. Both SSH and HTTPS are also scanned, as well as some very well known Trojan ports, including SubSeven, Trinoo, and Netbus. Scans targeting SQL Server vulnerabilities through port 1433, and Socks proxy on port 1080 were also conducted against MY.NET hosts through out this time period as seen in Table 3.3.5.

Destination Port	Number of scans
80	177013
21	79578
22	17627
443	14513
1524	10067
1433	8012
1080	4071
27374	837
12345	59

Table 3.3.5

Examination of the top external scanners, shown in Table 3.3.6, reveals that the majority of traffic is probably some form of Code Red type worm.

External Scanners	Total Scans Logged	Port Scanned
All Other IP's	203434	
80.135.93.213	18661	80
211.132.31.163	14540	80
61.18.71.113	13718	21
194.51.206.145	13195	80
164.2.255.244	12659	21
211.161.255.184	12322	21
195.146.82.34	12233	80
213.140.1.78	10382	80
172.183.169.84	10264	80
195.188.248.100	9770	80

Table 3.3.6

Overall scanning broken down into protocol shows that UDP is far more prevalent in the scan logs (Figure 3.3.5). The large number of UDP scans being logged can be directly attributed to the use of P2P and gaming programs by the student body, and by what appears to be normal DNS traffic.

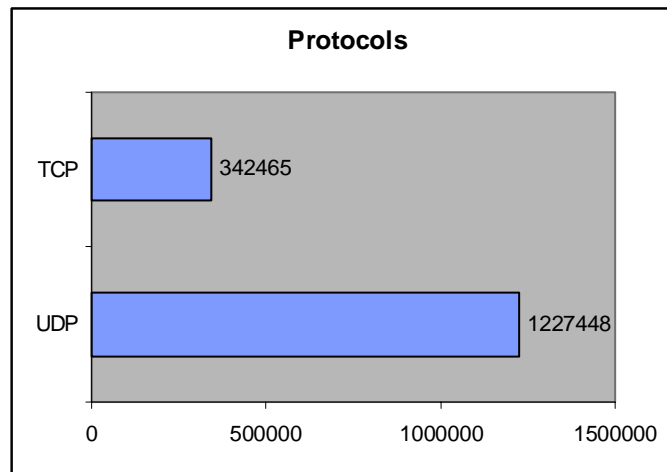


Figure 3.3.5

### 3.4 Alert Logs

The alert logs consisted of 118,607 alerts over the five-day period. This is roughly the same number of alerts generated by the MY.NET hosts as where generated from the outside world (see Figure 3.4.1). However, it was noted right away that the HTTP pre-processor installed on Snort was alerting on many seemingly normal HTTP events, possibly by cookies, the use of Unicode, or other normal browsing events from the MY.NET.

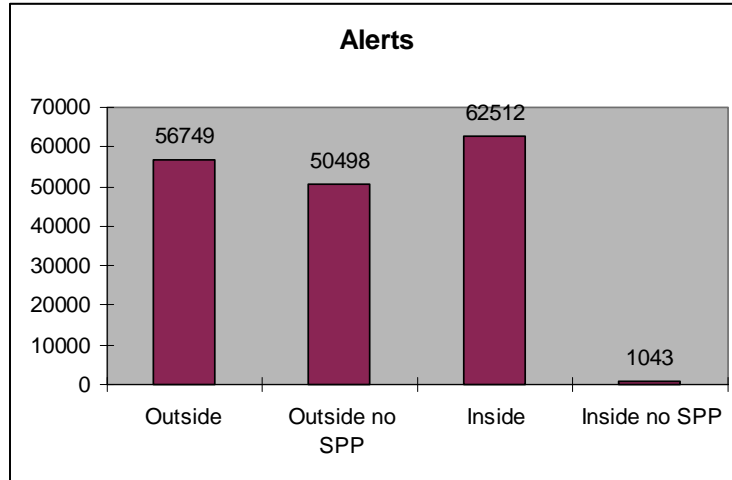


Figure 3.4.1

The huge diversity in internal alerts with and with out the preprocessor generated alarms is quite apparent. A total of 61469 alerts were attributed to HTTP traffic, which, in most cases were false positives. This pre-processor will have to be evaluated in light of the value of data it provides versus the number of false positives that are produced. Potentially hostile activity can very easily be lost in an overload of false positive alerts.

The following Table (3.4.1) contains all of the alerts logged over the reporting period and the number times they were generated.

Alert	Total
spp_http_decode: IIS Unicode attack detected	58122
Watchlist 000220 IL-ISDNNET-990517	11682
spp_http_decode: CGI Null Byte attack detected	9607
Watchlist 000222 NET-NCFC	8286
SYN-FIN scan!	7765
SUNRPC highport access!	6664
IDS552/web-iis_IIS ISAPI Overflow ida nosize	3552
External RPC call	2392
SMB Name Wildcard	2265
Attempted Sun RPC high port access	1797
Queso fingerprint	1552

Possible trojan server activity	1180
Incomplete Packet Fragments Discarded	1027
High port 65535 udp - possible Red Worm - traffic	715
Tiny Fragments - Possible Hostile Activity	650
EXPLOIT x86 NOOP	611
NMAP TCP ping!	319
NIMDA - Attempt to execute cmd from campus host	280
TCP SRC and DST outside network	129
IRC evil - running XDCC	112
Null scan!	110
EXPLOIT x86 setuid 0	68
Port 55850 udp - Possible myserver activity - ref. 010313-1	66
EXPLOIT x86 stealth noop	55
High port 65535 tcp - possible Red Worm - traffic	49
TFTP - Internal UDP connection to external tftp server	45
EXPLOIT x86 setgid 0	43
Port 55850 tcp - Possible myserver activity - ref. 010313-1	36
FTP passwd attempt	17
RFB - Possible WinVNC - 010708-1	10
External FTP to HelpDesk 10.10.70.49	10
EXPLOIT solaris NOOP	6
External FTP to HelpDesk 10.10.70.50	6
HelpDesk 10.10.70.49 to External FTP	6
HelpDesk 10.10.70.50 to External FTP	5
TFTP - External UDP connection to internal tftp server	4
ICMP SRC and DST outside network	4
Probable NMAP fingerprint attempt	4
EXPLOIT NTPDX buffer overflow	4
HelpDesk 10.10.83.197 to External FTP	3
connect to 515 from inside	3
FTP DoS ftpd globbing	3
External FTP to HelpDesk 10.10.83.197	2
TFTP - External TCP connection to internal tftp server	2
TCP SMTP Source Port traffic	2
Back Orifice	2
TFTP - Internal TCP connection to external tftp server	1

Table 3.4.1

Two of the more numerous alarms identified were generated from a locally grown set of rules found as Watchlist 000220 IL-ISDNNET-990517 and Watchlist 000222 NET-NCFC. The ISDN Net is a net block belonging to ISDN Net Ltd, an Israeli Internet Service Provider. The NCFC belongs to [The Computer Network Center Chinese Academy of Sciences](#), which has intellectual agreements with many Universities and organizations around the world, including the United States. The file sharing program Kazaa, communicating through port 1214 caused a total of 14293 alerts, or 71%, from these two sources. Aside from the file sharing, the remainder of this Watchlist traffic appears to be normal web traffic. An

accounting of this Watchlist and why it was implemented should be made accessible to analysts for assessment on future reports.

Further analysis was conducted on the Alert logs, once the worm type traffic, web traffic, and Watchlist logged alerts were removed. The more serious types of remaining alerts are covered in the Detects section (section 3.6) of this report. While the web traffic cannot be ignored, if routine security auditing is done on the network systems coupled with updated virus scanning, it becomes more network background 'noise' and therefore a nuisance. This type of traffic was not looked any further.

It is very concerning that a large number of these remaining Alert logs seem to be generated by the use of Peer to Peer file sharing and online gaming. As an example of those logs please review the following:

NMAP TCP ping! These alerts were reviewed and appear to be the results of on-line gaming and Peer-to-Peer programs running on MY.NET hosts. No further investigation of these alarms is warranted.

Port 55850tcp – Possible myserver activity – ref 010313-1 These alarms were investigated and found to be the results of web traffic and Peer to Peer software. No further investigation is required

High Port 55850 tcp – possible Red Worm – traffic Alarms were generated by one internal host involved in Peer to Peer file sharing, and one listed in the Alert logs for Back Orifice traffic. This host is examined in the Back Orifice alert listed in Table 3.3.2.

TFTP – Internal UDP connection to external tftp server Appears to be normal TFTP traffic, and traffic that is associated with known Peer to Peer ports.

After these alerts were examined, it left several more of lesser concern that should be looked into. The following are presented as examples of what is left over. With few exceptions, they all appear to be benign in nature.

EXPLOIT x86 The vast majority of these type of alerts are being generated by seemingly legitimate web traffic from sites such as <http://home.eatel.net/>, [University of Maryland's Project Glue](#), and [Earth & Space Data Computing Division \(ESDCD\)](#) at NASA.

The main suggestion presented in this case is to update the Snort Rules Set that is being used, since a majority of these types of alerts are firing on rules that appear to be out of date. Alerts that may be of concern are being masked by web traffic that is firing as false positives and may cause a serious exploit attempt to be overlooked.

Back Orifice This alert appears twice in the logs, generated by two different external hosts that are attempting a connection to port 31337 of two separate MY.NET hosts. Host 61.74.67.25 generated on Alert and does not appear in the Scans logs. Host 211.162.191.12 appears three times in the Alert logs (see Table 3.4.2).

Further investigation into these MY.NET hosts should be under taken to ensure that neither system is compromised.

Sep-13	00:10.7	High port 65535 udp - possible Red Worm - traffic	211.162.191.12	65535	MY.NET.153.142	65472
Sep-15	07:53.7	High port 65535 udp - possible Red Worm - traffic	211.162.191.12	60185	MY.NET.153.143	65535
Sep-17	17:00.4	Back Orifice	211.162.191.12	48508	MY.NET.153.142	31337

Table 3.4.2

FTP passwd attempts On the 16<sup>th</sup> of September IP 203.218.251.10 made 17 attempts to log in to FTP services running on the hosts MY.NET 5.92 and MY.NET.5.95. Whether the IP that generated this alert was trying to guess passwords or was legitimately trying to log in is unknown. Both MY.NET address do not appear on the list of FTP enabled hosts that responded to the scan reviewed in section 3.5.1, SYN-FIN Scan!

Further investigation into this should be made by the IT security staff to ensure that any FTP services running are patched securely and that good password selection is reviewed with the owners.

### **3.5 OOS Logs**

Out of Specification logs are the result of invalid or unusual TCP flag combinations. The usual causes of these packets are badly configured routers or crafted packets. Packets crafted by such programs as [Hping2](#), [Nmap](#), and [Queso](#) are specifically designed to evade or to be overlooked by network security applications. Many older firewall and IDS were specifically looking for completed three way handshakes before logging events. Thus as example, a SYN FIN combination might not be logged, however the destination host would reply with a SYN ACK by disregarding the FIN flag. Some operating system detection systems like NMAP use specially crafted packets with numerous flags to help 'guess' the OS by the analyzing of the responses.

Over the five day period 12 external hosts generated 34 packets that where logged, but only three are generating more than one packet. These three IP's with multiple OOS packets breakdown as per Table 3.5.1.

IP	Flags	IP	Flags	IP	Flags
24.112.58.210	**sf***u	64.4.124.151	21**r**u	195.101.94.208	21s*****
	*1sfrpau		21**rp*u		21s*****
	2*sfrpau		21**r***		21s*****
	21sf***u		21**r**u		
	*1sf**au		21**rp*u		
	21*frp*u		*1sf****		
			21**rp*u		

Table 3.5.1

While all three of these hosts look like they are attempting to remotely type the OS of the destination host the actual tool used is unknown. NMap uses combinations of TCP flags to gauge responses as part of its sequence of testing. Other programs such as Queso also do this in a similar manner.

The IP 24.112.58.210 was logged on the 12<sup>th</sup> September and did not coincide with the Scan or Alert logs so no further correlation is available for this address. IP 64.4.124.151 scans on the 13<sup>th</sup> September and coincides with the Scan and Alert logs, however no logs originated from this IP in either the Scan or Alert logs.

The IP address 195.101.94.208 appears 461 times in the Scan logs and 458 times in the Alert log with the 'Queso fingerprinting' alert. In both the Scan and Alert logs the scanner looks at the same 22 MY.NET addresses as seen in Table 3.5.2.

MY.NET.100.158	MY.NET.140.2	MY.NET.162.235	MY.NET.5.95
MY.NET.100.237	MY.NET.145.18	MY.NET.162.87	MY.NET.70.231
MY.NET.106.222	MY.NET.145.76	MY.NET.163.133	MY.NET.99.174
MY.NET.109.71	MY.NET.145.9	MY.NET.179.77	MY.NET.99.85
MY.NET.130.123	MY.NET.150.83	MY.NET.179.78	
MY.NET.130.91	MY.NET.157.52	MY.NET.179.80	

Table 3.5.2

These addresses are of a relatively random nature and are the likely result of some earlier reconnaissance scan and are likely to be the target of future attacks or exploit attempts. It would be prudent at this time to add this IP to the Watchlist or have it blocked at the firewall or gateway router. These listed MY.NET addresses should be examined closely by the IT staff for signs of compromise or vulnerabilities.

## **3.6 Detects**

The following detects were identified to bring to your attention because of the potential security concerns. External worm traffic was weeded out, as well as Watchlist traffic, since this was previously analyzed and found to be of a lesser security concern.

### **3.6.1 SYN-FIN Scan!**

#### **3.6.1.1 Description**

This alert was generated by the source host using a tool like [Synscan](#) to scan the University's net block. This involved the scanning of 7765 addresses inside the MY.NET/16 over a 17-minute time period starting at 2016hrs on the 14<sup>th</sup> September. The same host appears in 7872 entries of the scan logs for the same time period, making him one of the top talkers for the five-day period covered in this report.

```
09/14-20:16:38.693046 [**] SYN-FIN scan! [**] 195.199.74.91:21 -> MY.NET.5.14:21
09/14-20:16:39.032104 [**] SYN-FIN scan! [**] 195.199.74.91:21 -> MY.NET.5.31:21
09/14-20:16:39.272141 [**] SYN-FIN scan! [**] 195.199.74.91:21 -> MY.NET.5.43:21
```

Though the mechanics of the Synscan tool are covered in Section 2, Detect 1, it is important to mention the basics here.

The source host 195.199.74.91 scans the MY.NET/16 with both the SYN and FIN flags are set which is the Snort rule that triggered the logging of this traffic. This snippet from the Scans log (Table 3.6.1) show the first five packets logged. As the numbers show this is a very fast scan that only logs attempted connections to live hosts.

Date	Time	Source IP	Src Port	Dest IP	Dst Port	TCP Flags
00/14	8:16:18 PM	195.199.74.91	21	MY.NET.1.2	21	SYNFIN
00/14	8:16:28 PM	195.199.74.91	21	MY.NET.3.2	21	SYNFIN
00/14	8:16:38 PM	195.199.74.91	21	MY.NET.5.14	21	SYNFIN
00/14	8:16:39 PM	195.199.74.91	21	MY.NET.5.31	21	SYNFIN
00/14	8:16:39 PM	195.199.74.91	21	MY.NET.5.43	21	SYNFIN

Table 3.6.1

The following Table (3.6.2) is a snippet from the Alerts log showing the same activity.

Time	Alert	Source IP	Src Port	Dest IP	Dst Port
20:16:18.055121	SYN-FIN scan!	195.199.74.91	21	MY.NET.1.2	21
20:16:28.252454	SYN-FIN scan!	195.199.74.91	21	MY.NET.3.2	21
20:16:38.693046	SYN-FIN scan!	195.199.74.91	21	MY.NET.5.14	21
20:16:39.032104	SYN-FIN scan!	195.199.74.91	21	MY.NET.5.31	21
20:16:39.272141	SYN-FIN scan!	195.199.74.91	21	MY.NET.5.43	21

Table 3.6.2

The actual rule used to trigger the logging is unknown. No version or rule set was supplied with the information for this report. However the rule that triggered would look very much like this rule in the Snort Standard Rule Set 1.8.1:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN
FIN";flags:SF;reference:arachnids,198; classtype:attempted-
recon; sid:624; rev:1;)
```

Syncan versions 1.5/1.6 work on the principle of scanning by avoiding IDS and firewalls with both the SYN and FIN flags. This will solicit a SYN ACK from live hosts as the TCP protocol requires as per [RFC 793](#), and at the same time will avoid logging by older IDS/firewalls that require a completed three way handshake to trigger. Once the attacker receives a reply from a scanned host he then sends a RESET to tear down the connection. Once done he then sends a legitimate SYN to open a connection and grab the FTP banner. This banner information is used by several worms, such as Ramen, to identify vulnerable systems, or by an attacker to see if a host might be vulnerable to one of many other known exploits against [FTP](#).

In all 121 hosts responded to this scan, as shown in Table 3.6.3. What this indicates is that all of these hosts have some form of FTP service running. What it does not indicate is what type of service is running on what OS, or what level of security patching has been undertaken on these hosts.

MY.NET.5.14	MY.NET.84.196	MY.NET.106.222	MY.NET.116.53	MY.NET.162.30
MY.NET.5.31	MY.NET.84.216	MY.NET.109.88	MY.NET.116.86	MY.NET.162.31
MY.NET.5.95	MY.NET.84.224	MY.NET.109.89	MY.NET.130.42	MY.NET.162.62
MY.NET.27.3	MY.NET.85.127	MY.NET.110.28	MY.NET.130.181	MY.NET.162.82
MY.NET.53.228	MY.NET.86.17	MY.NET.110.56	MY.NET.130.182	MY.NET.162.109
MY.NET.53.229	MY.NET.86.21	MY.NET.110.165	MY.NET.130.190	MY.NET.162.235
MY.NET.70.13	MY.NET.86.112	MY.NET.110.172	MY.NET.130.123	MY.NET.162.91
MY.NET.70.14	MY.NET.86.116	MY.NET.111.30	MY.NET.136.2	MY.NET.162.240
MY.NET.70.15	MY.NET.87.46	MY.NET.111.38	MY.NET.139.27	MY.NET.162.241
MY.NET.70.76	MY.NET.87.188	MY.NET.111.91	MY.NET.139.55	MY.NET.162.242
MY.NET.70.93	MY.NET.70.13	MY.NET.111.162	MY.NET.140.24	MY.NET.163.43
MY.NET.70.113	MY.NET.70.113	MY.NET.111.197	MY.NET.140.74	MY.NET.163.56
MY.NET.70.170	MY.NET.70.231	MY.NET.111.234	MY.NET.145.7	MY.NET.163.98
MY.NET.70.205	MY.NET.99.85	MY.NET.113.208	MY.NET.145.75	MY.NET.163.127
MY.NET.70.76	MY.NET.99.121	MY.NET.113.207	MY.NET.146.20	MY.NET.165.28
MY.NET.70.231	MY.NET.99.122	MY.NET.113.211	MY.NET.150.16	MY.NET.177.37
MY.NET.70.13	MY.NET.99.172	MY.NET.113.213	MY.NET.150.31	MY.NET.178.166
MY.NET.70.13	MY.NET.100.15	MY.NET.113.221	MY.NET.150.195	MY.NET.178.254
MY.NET.70.113	MY.NET.100.120	MY.NET.113.223	MY.NET.157.52	MY.NET.179.78
MY.NET.70.76	MY.NET.104.213	MY.NET.114.45	MY.NET.157.150	MY.NET.179.81
MY.NET.80.144	MY.NET.105.40	MY.NET.114.116	MY.NET.158.73	MY.NET.180.13
MY.NET.82.121	MY.NET.105.204	MY.NET.115.114	MY.NET.162.14	MY.NET.180.32
		MY.NET.180.40		
		MY.NET.180.60		

Table 3.6.3

Being a University and having an open door for information sharing and collaboration several recommendations must be made to mitigate the risk of compromise on one or more hosts running FTP services.

Previous GCIA reports include many examples of SIN/FIN alerts including Wes Bateman’s (0385) report and one done by David R. Williams (0510).

### **3.6.1.2 Recommendations**

All of the listed hosts that responded to this scan should be, though time consuming, looked at closely for any evidence of exploitation. At the same time the FTP services should be verified as having the latest vendor security patches installed and that there is an actual need for these services to be running.

An alternative approach might be to block all FTP traffic at the firewall with the exception of those registered with the IT staff as being essential and up to date with patching.

Up to date signatures for both network and host anti virus programs should be maintained and enforced by the IT staff.

### **3.6.1.3 References**

<http://www.faqs.org/rfcs/rfc793.html>

<http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=ftp>

<http://www.psychoid.lam3rz.de/synscan1.6.tar.gz>

<http://www.cve.mitre.org/cve/>

[http://www.whitehats.ca/main/publications/external\\_pubs/scanner\\_fingerprints/scanner\\_fingerprints.html](http://www.whitehats.ca/main/publications/external_pubs/scanner_fingerprints/scanner_fingerprints.html)

## **3.6.2 SMB Name Wildcard**

### **3.6.2.1 Description**

The Network Basic Input/Output System, or [NETBIOS](#), is a client to server communications application used on Windows, Ethernet, or Token Ring based networks. Port 137 is the NETBIOS Name Service, which is used to resolve IP addresses into NETBIOS names on systems with NETBIOS enabled. There are three types of NETBIOS services, the datagram service, the session service, and the name service. If an attacker can successfully copy the name table from the server he has access to the following information.

1. The servers NETBIOS name
2. The domain names for Windows NT workgroups
3. The log in names of all the users currently logged in to that server, including the Administrator.

This information can be made available from a host with the nbtstat -a command.

The **Server Message Block Name Wildcard** alert was logged 2265 times over the reporting period. These alarms were all generated from external sources scanning port 137 of the MY.NET/16 using UDP, and may be looking for open shares as part of a reconnaissance probe before an actual attack. The only exceptions are host's 192.168.5.2 and 10.2.70.46, which, by the traffic generated, appear to be internal [WINS servers](#). These hosts look like they are part of an internal network that has somehow generated traffic that appears outside of its network gateway. It is impossible to say with complete accuracy that these two hosts are part of the Universities infrastructure or addresses leaking out of another networks gateway. Leakage of this nature was recently

discovered on my employer's network, and was traced to an external network miss configurations.

A total of 139 different hosts generated these alarms over a five-day period.

```
09/17-12:04:44.421654 [**] SMB Name Wildcard [**] 80.11.207.237:1025 ->MY.NET.133.143:137
09/17-11:57:45.354942 [**] SMB Name Wildcard [**] 80.11.207.237:1025 ->MY.NET.133.104:137
```

All of these hosts are external with the exception of the one internal host. This can probably be accounted for if the rule that triggers this alarm has been tightened to only fire on the external source and not internal to internal traffic as was noted in many previous reports to GCIA. Most of these alarms look to be a quirk of Windows and is not necessarily someone probing these hosts.

While often considered normal noisy network traffic there is one host that is definitely probing the MY.NET/16 hosts. IP 80.11.207.237 is responsible for 1163 alerts that have port 137 as the destination port but have an ephemeral port for a source. Normal NETBIOS traffic would be conducted port 137 to port 137. Without access to firewall logs or an internal sensor it is unknown if this attacker was able to gain any useful information. It is interesting to note that our source IP 80.11.207.237 also triggers several TFTP alarms on both the 16<sup>th</sup> and 17<sup>th</sup> of September. This is looked at closer in the Internal Systems of Interest section (Section 3.7).

Correlation for this alert can be found in many previous GCIA reports, notably Brain K. Sheffler (0531), Pedro Bueno (0518), and Randall Gillipsie (0517).

### **3.6.2.2 Recommendations**

With the type of information that is potentially available from reconnaissance of this type the major recommendation that can be made is to completely block all traffic to the NETBIOS ports of 137, 138, and 139 at the firewall. Any trusted host that requires access on these ports can be configured into the firewall Access Control List.

Tweaking the Snort rule that triggers this alert so that it does not fire on seemingly regular NETBIOS traffic would be a good idea to cut down on the number of alerts an analyst has to inspect. Having another sensor on the interior of the firewall would certainly help in the analysis process.

Adding the 80.11.207.237 address to the Watchlist or blocking it completely would also be prudent.

### **3.6.2.3 References**

<http://archives.neohapsis.com/archives/snort/2000-01/0222.html>  
<http://lists.jammed.com/incidents/2001/05/0034.html>  
[http://www.finchhaven.com/pages/incidents/030102\\_udp\\_137.html](http://www.finchhaven.com/pages/incidents/030102_udp_137.html)  
[http://support.baynetworks.com/library/tpubs/html/router/soft1200/117358AA/B\\_39.HTM](http://support.baynetworks.com/library/tpubs/html/router/soft1200/117358AA/B_39.HTM)  
<http://www.j51.com/~sshay/tcpip/wins/wins.htm>  
<http://ourworld.compuserve.com/homepages/timothydevans/contents.htm>

## **3.6.3 IRC Evil – Running XDCC**

### **3.6.3.1 Description**

Internet Chat Relay, or IRC, is a world wide system of computers connected together for communicating with other people, much like chat rooms. IRC is the play ground of 'script kiddies' and 'warez' communities, and is often used to transfer pirated software and movies between them. Warez groups will typically use bots or servers, such as XDCC to do this.

An XDCC server is on that allows remote users to access files on a host that is offering them up in 'packs'. This is done by the user advertising his packs on IRC channels for members of those channels to download. These servers generally offer out movies, games, and music over high-speed lines. Just for this reason many XDCC servers have been compromised and are used without the hosts' knowledge.

Generally a tool such as [X-Scan](#) is used over NETBIOS to find unprotected computers that have no log in passwords. The attacker then uploads several utilities, including servu FTP server, and the XDCC bot iroffer, and runs them which results in an XDCC bot joining a designated IRC channel. Now the attacker uses the servu FTP server to download files onto the victims' machine, packages them up and use iroffer to advertise themselves to the IRC channel. Over the five day reporting period 112 alerts which were generated by two MY.NET hosts, MY.NET.84.172 and MY.NET.91.242, that are running IRC XDCC servers and making connections to 6 separate hosts. These two addresses do not appear in any other alerts other than the IRC Evil alerts. This was researched from these examples of the alarm

```
09/17-02:33:43.195628 [**] IRC evil - running XDCC [**] MY.NET.91.242:2832 ->
66.250.160.123:6667
09/17-02:40:33.197179 [**] IRC evil - running XDCC [**] MY.NET.91.242:3356 ->
66.250.160.123:6667
09/17-02:40:41.195719 [**] IRC evil - running XDCC [**] MY.NET.91.242:4701 ->
66.250.160.123:6667
```

The destination port of 6667 is used by IRC Server, though without the rule set that Snort is using to generate these alarms, one cannot say what the rule actually was. Combing through Snort rule set 1.8.1 I could not find an exact match to IRC Evil – running XDCC.

I found many references to IRC being used for software pirating, mostly from unknowing hosts that have been taken over for this purpose. [Securiteam](#) posted an informative article on this on 4/5/2002. A more in depth look at this procedure has been written by [TonikGin](#), as well as a good run down of the tools involved.

There is no doubt that these two hosts are running XDCC servers, but there is no way of knowing if these systems have been compromised or are being used by their owners.

### **3.6.3.2 Recommendations**

The immediate examination of these two systems is paramount. If the hosts have been compromised then their use for other exploits of internal systems is only a matter of time. A good paper on the removal of XDCC from a system can be found at the [Duke University](#) site.

To prevent the exploitation of a system in order to install and serve up illegal files, it is vital to educate the network users about the necessity of using strong passwords to protect their computers. Bad guys will use scanners to look for systems with weak or no passwords, and again NETBIOS is used as part of this exploit and should be block at the firewall. What doesn't get in cannot embarrass you later.

### **3.6.3.3 References**

<http://www.dslreports.com/faq/4493>  
<http://www.russonline.net/tonikgin/EduHacking.html>  
<http://members.aol.com/lamesn/myhomepage/XDCC.htm>  
<http://www.astalavista.com/library/auditing/netbios/netbios-tutorial.txt>  
<http://security.duke.edu/cleaning/xdcc.html>  
<http://www.securiteam.com/securitynews/5ZP021575W.html>  
[http://www.infoslash.org/topic.asp?TOPIC\\_ID=194&FORUM\\_ID=7&CAT\\_ID=2&Forum\\_Title=Security+News&Topic\\_Title=Massive+EDU+Hacking+Tactics+Exposed](http://www.infoslash.org/topic.asp?TOPIC_ID=194&FORUM_ID=7&CAT_ID=2&Forum_Title=Security+News&Topic_Title=Massive+EDU+Hacking+Tactics+Exposed)

## 3.6.4 External RPC call

### 3.6.4.1 Description

This alert was triggered 2392 times by four IP addresses over the five day reporting period. These hosts are 61.128.128.52, 80.11.207.237, 136.145.82.46, and 217.83.2.100. All of these hosts also appear in the scan logs a total of 1331 times in the same time frame looking for the portmapper listening on port 111. The most prevalent is 80.11.207.237 that is making these calls in conjunction with SMB Name Wildcard attempts. An example of the Alert generated is:

```
09/13-07:30:03.454882 [**] External RPC call [**] 61.128.128.52:54535 ->MY.NET.133.4:111
09/13-07:30:03.459749 [**] External RPC call [**] 61.128.128.52:54542 ->MY.NET.133.11:111
09/13-07:30:03.460421 [**] External RPC call [**] 61.128.128.52:54538 ->MY.NET.133.7:111
```

The Remote Procedure Call was developed by Sun Microsystems for UNIX, and now LINUX, operating systems as a client/server infrastructure to share services over a number of different operating systems. This allows a client to request a service from a remote server on the network, independent of network setup, and run that service as if it were on the client.

The following is a list of services that can be found offered on systems with RPC running. This list is quoted from [University of Cambridge Computing Service](#):

rpcbind

This service name corresponds to the portmapper itself. If you want to run *any* RPC service then you need to be running the portmapper.

nfs, mountd, nfs\_acl

The **Network Filing System**. If you want to share some of your files with other systems then you must run these services. `nfs_acl` is the same as `nfs` but has support for Access Control Lists as well. See CERT advisory [CA-98.12.mountd](#) for details of a common security hole in this service.

status, llockmgr, nlockmgr

These services are used for file locking over NFS. You need them if you are exporting your own files or importing someone else's.

walld

A utility for letting people send messages to every user of the system. Rarely useful and more often a pain in the arse.

rstatd

A utility for letting remote systems know your load average. Rarely useful but the `perfmeter` tool needs it.

rusersd

A service for letting remote systems know which of your users are logged on. Rarely useful.

rquotad

If you export or import file systems with quotas on them then you need to run this service. Otherwise you don't.

bootparam

A silly RPC based replacement for `bootp`. If you are a boot server you may need this, otherwise not.

ypbind

All systems in a YP (a.k.a. NIS) domain need to be running this service. If you are not using YP then you should not be running it.

ypserv

All YP servers (Master and Slave servers) should be running this service. YP clients should not.

tooltalk

This is used for some graphical operations like drag and drop, we think. See CERT advisory [CA-98.11.tooltalk](#) for details about a common security hole in this service.

cmsd

This is the CDE Calendar Manager service which is rarely used and was the source of a serious security hole. (See CERT advisory [CA-99-08-cmsd](#) for details.)

All RPC services are assigned a service number at system start up. The portmapper service, or `rpcbind`, is run in order to convert service numbers into TCP/IP usable port numbers and as the RPC server is started up it tells the portmapper what services are being offered on what port. These ports can vary from system to system, but are usually found in the 32770-32789 port range. In essence the portmapper maps the incoming RPC to the port the service is listening on. Once a system is identified as running this service on port 111 the attacker can query portmapper with the command:

```
rpcinfo -p (hostname/IP)
```

For all RPC services that are running, the query will return:

- Service Name
- Service Number
- Port Number

Once this information is picked up by the attacker the actual exploit attempt will probably not be long after. Remote Procedure Call has many known vulnerabilities and [CVE's](#) listed against the services that are run through it.

### **3.6.4.2 Recommendations**

The primary recommendation to be made is to block the port 111 and the loop back ports, 32770-32789(TCP and UDP) at the firewall or gateway. An audit of RPC services being run should be conducted and the removal or shutting down of all unnecessary services should also be considered. For all RPC services that

are deemed essential and are continuing to run, the latest vendor patch must be applied to protect against exploitation.

### **3.6.4.3 References**

<http://www-uxsup.csx.cam.ac.uk/security/probing/about/sunrpc.html>

<http://www.sei.cmu.edu/str/descriptions/rpc.html>

<http://www.sans.org/top20/#U1>

<http://www.linux-nis.org/nis-howto/HOWTO/portmapper.html>

“Intrusion Signatures and Analysis” by Northcutt, Cooper, Fearnow, and Frederick

## **3.7 Internal Systems of Interest**

### **3.7.1 Nimda Infection**

On the 15<sup>th</sup> September MY.NET.115.115 started a scan of four complete class C subnets on the 209.10/16 net block. Over the course of this scan this address was logged a total of 6979 times.

```
Sep 15 08:34:06 130.85.115.115:2042 -> 209.10.100.1:445 SYN *****S*
Sep 15 08:34:06 130.85.115.115:2043 -> 209.10.100.1:139 SYN *****S*
Sep 15 08:34:05 130.85.115.115:2044 -> 209.10.100.1:80 SYN *****S*
Sep 15 08:34:06 130.85.115.115:2046 -> 209.10.100.1:1433 SYN *****S*
Sep 15 08:34:08 130.85.115.115:137 -> 209.10.100.1:137 UDP
```

The targeted ports of this scan are as listed.

TCP port 80	HTTP
TCP port 139	NETBIOS Session Service
TCP port 445	Win2K Server Message Block
TCP port 1433	Microsoft SQL Server
UDP port 137	NETBIOS Name Service

MY.NET.115.115 was also logged in the Alert logs 636 times for the same two alerts. These alerts all coincide with the port 80 of the above mentioned scans.

```
spp_http_decode:ISS Unicode attack detected
NIMDA – Attempt to execute cmd from campus host
```

This host is obviously infected with [NIMDA](#), or one of the 'mutated' forms of the worm. The immediate removal from the network is imperative. The IT staff should ensure that a Virus Scanner with the latest updated signatures is installed on this host and run. This worm is not new so the likelihood that this system has been infected with other newer worms or exploits is very real. A thorough audit of this system should be under taken before this host is put back on the network.

### **3.7.2 Possible Trojan Horses**

On 13<sup>th</sup> September the 217.148.2.61 conducted a scan of four class C subnets of the MY.NET network on port 27374. Subnets .132, .133, .134, and 135 were scanned logging a total of 795 times in the Alert logs and 777 times in the Scan logs. The following is an excerpt taken from the Alert logs.

```
09/13-16:53:57.386835 [**] Possible trojan server activity [**] 217.148.2.61:4912 ->
MY.NET.132.4:27374
09/13-16:53:57.531308 [**] Possible trojan server activity [**] 217.148.2.61:4915 ->
MY.NET.132.7:27374
09/13-16:53:57.537717 [**] Possible trojan server activity [**] 217.148.2.61:4916 ->
MY.NET.132.8:27374
```

The 27374 port is used by many Trojans and is scanned by hosts wishing to connect and use already installed malware. There are 14 different Trojans listed on the [Treachery Unlimited Port Lookup](#), including SubSeven, Lion, Ramen and The Saint. Seen next is an example taken from the Scan logs.

```
09/13-16:53:58.603634 [**] Possible trojan server activity [**] MY.NET.132.1:27374 -> 217.148.2.61:4909
09/13-16:53:58.633539 [**] Possible trojan server activity [**] MY.NET.132.24:27374 -> 217.148.2.61:4932
09/13-16:53:59.584050 [**] Possible trojan server activity [**] MY.NET.132.30:27374 -> 217.148.2.61:4938
09/13-16:53:59.785630 [**] Possible trojan server activity [**] MY.NET.132.17:27374 -> 217.148.2.61:4925
09/13-16:53:59.844695 [**] Possible trojan server activity [**] MY.NET.132.22:27374 -> 217.148.2.61:4930
09/13-16:54:34.136720 [**] Possible trojan server activity [**] MY.NET.133.17:27374 -> 217.148.2.61:1206
09/13-16:55:08.908819 [**] Possible trojan server activity [**] MY.NET.134.10:27374 -> 217.148.2.61:1454
09/13-16:55:08.956275 [**] Possible trojan server activity [**] MY.NET.134.11:27374 -> 217.148.2.61:1455
09/13-16:55:41.844413 [**] Possible trojan server activity [**] MY.NET.135.1:27374 -> 217.148.2.61:1699
```

As a result of this scan nine MY.NET address responded from port 27374 telling us that a Trojan is most likely installed on these hosts. An immediate inspection of these system responding should take place. The IT staff must ensure that anti-virus software is installed on all mentioned hosts and that it is kept up to date with current releases.

### **3.7.3 IRC XDCC**

These two hosts, MY.NET.84.172 and MY.NET.91.242, are mentioned again here, as the probability for these hosts to be compromised is rather high. Please refer to the above detect, section 3.5.3, for further information.

### **3.7.4 TFTP – Internal/External Connections**

This IP of 80.11.207.237 has been mentioned several times in this report as being the source of several large scans and numerous alerts. On 16<sup>th</sup> September at 11:36 am this IP started a port scan of MY.NET.114.110 that hit 799 different ports before he was done. At the same time this external address also connected to a TFTP service that was running on MY.NET.114.110 and was logged in the Alert logs. An example taken from the Alert logs looks like this:

```
09/16-11:37:10.068925 [**] TFTP - External TCP connection to internal tftp server [**] 80.11.207.237:3857 ->
MY.NET.114.110:69
09/16-13:20:23.351027 [**] TFTP - Internal UDP connection to external tftp server [**] MY.NET.116.47:3422 ->
80.11.207.237:69
09/17-11:16:05.817822 [**] TFTP - Internal UDP connection to external tftp server [**] MY.NET.132.23:1044 ->
80.11.207.237:69
```

While there are several other TFTP connections to and from the MY.NET addresses, none of the external IP's counted in any other events logged.

Two addresses, MY.NET.116.47 and MY.NET.132.23, show up in the Alert logs with connection attempts over UDP to 80.11.207.237 on the 16<sup>th</sup> and 17<sup>th</sup> respectively. Many common Trojans communicate to their controllers using TFTP and one should assume for the sake of security that this is what is occurring with these two hosts.

Because of the activity that the 80.11.207.237 host has been involved in it would be prudent to inspect these three hosts for evidence of compromise. These services should be turned off and blocked unless they are found to be of a legitimate nature. Again, adding 80.11.207.237 to the Watchlist or outright blocking it would be prudent to keep an eye on or prevent him from connecting to any MY.NET hosts.

### **3.7.5 Port Scanning**

Two internal MY.NET hosts have been found in the Scan logs using [Nmap](#), or a similar tool, to scan external addresses. Each MY.NET address targets one specific external host and does a large port scan over a wide range of reserved and ephemeral ports (see Table 3.7.5.1). While this might be the work of an infecting worm or exploitation of the system, this analyst believes it is in fact the work of the host user. A more in depth look at these systems would be recommended if more than one destination was targeted.

<b>MY.NET Address</b>	<b>Target</b>	<b>Number of Ports Scanned</b>
MY.NET.163.132	128.183.252.83	538
MY.NET.179.78	66.80.148.137	460

Table 3.7.5.1

As you can see from the log snips below MY.NET.163.132 is scanning his target by sequential ports, while MY.NET.179.78 has randomized his target ports.

```
Sep 17 05:17:30 MY.NET.163.132:32898->128.183.252.83:18871 SYN *****S*
Sep 17 05:17:30 MY.NET.163.132:32899->128.183.252.83:18872 SYN *****S*
Sep 17 05:17:30 MY.NET.163.132:32901->128.183.252.83:18874 SYN *****S*
Sep 17 05:17:30 MY.NET.163.132:32902->128.183.252.83:18875 SYN *****S*
Sep 17 05:17:30 MY.NET.163.132:32903->128.183.252.83:18876 SYN *****S*
Sep 17 05:17:31 MY.NET.163.132:32905->128.183.252.83:18878 SYN *****S*
```

```
Sep 16 09:04:05 MY.NET.179.78:53379 -> 66.80.148.137:63 SYN *****S*
Sep 16 09:04:05 MY.NET.179.78:53380 -> 66.80.148.137:474 SYN *****S*
Sep 16 09:04:05 MY.NET.179.78:53381 -> 66.80.148.137:2065 SYN *****S*
Sep 16 09:04:05 MY.NET.179.78:53382 -> 66.80.148.137:5002 SYN *****S*
Sep 16 09:04:05 MY.NET.179.78:53383 -> 66.80.148.137:1395 SYN *****S*
Sep 16 09:04:05 MY.NET.179.78:53384 -> 66.80.148.137:918 SYN *****S*
```

What the overall intent of these scans is remains unknown. However the potential information gained can be used to attempt an exploit against these systems in the future. The 128.183.252.83 belongs to NASA.gov, and the reader can surmise the potential embarrassment to the University by having its name associated with a system intrusion at NASA. This could in fact jeopardize future projects with NASA and should be treated very seriously by the University.

These two MY.NET addresses belong to net blocks that are part of the residential network, deduced by the heavy P2P and gaming traffic. The

individual users should be counselled in respect to the Universities Acceptable Use Policy, and if any contrary activity takes place again the users should be denied access.

If no Acceptable Use Policy is in use, then the University should consider having every user read and sign such a policy. Examples of [AUP's](#) can be found via the Internet on many sites.

## **3.8 External Addresses of Interest**

### **3.8.1 80.11.207.237**

This host was chosen in this section because of the activities he was involved in, including scanning, SMB Name Wildcard alerts, External RPC call alerts, and connection to internal TFTP services. As listed by [Ripe.net](#):

```
inetnum: 80.11.207.0 - 80.11.207.255
netname: IP2000-ADSL-BAS
descr: BSANN101 Annecy Bloc2
country: FR
admin-c: WITRI-RIPE
tech-c: WITRI-RIPE
status: ASSIGNED PA
remarks: for hacking, spamming or security problems send mail to
remarks: postmaster@wanadoo.fr AND abuse@wanadoo.fr
remarks: for ANY problem send mail to
          gestionip.ft@francetelecom.com
mnt-by: FT-BRX
changed: gestionip.ft@francetelecom.com 20011011
source: RIPE

route: 80.11.192.0/19
descr: France Telecom
descr: Wanadoo Interactive
remarks: -----
remarks: For Hacking, Spamming or Security problems
remarks: send mail to abuse@wanadoo.fr ONLY
remarks: -----
origin: AS3215
mnt-by: RAIN-TRANSPAC
mnt-by: FT-BRX
changed: karim@rain.fr 20020226
source: RIPE
```

### **3.8.2 217.148.2.61**

This host scanned many systems in the MY.NET network, appearing a total 777 times in the Scan logs and 795 times in the Alert logs. The scanner received replies from nine separate MY.NET hosts that appear to be running a Trojan that communicates over port 27374. As listed by [Ripe.net](#):

**inetnum:** 217.148.0.0 - 217.148.3.255  
**netname:** SHLINK  
**descr:** MTF Schaffhausen AG  
**country:** CH  
**admin-c:** [SHLI1-RIPE](#)  
**tech-c:** [SHLI1-RIPE](#)  
**status:** ASSIGNED PA  
**notify:** message@shlink.ch  
**mnt-by:** [AS20988-MNT](#)  
**changed:** roland.zanella@mtf.ch 20021024  
**source:** RIPE  
**route:** 217.148.0.0/20  
**descr:** CH-MTFSSH-NET  
**origin:** [AS20988](#)  
**mnt-by:** [SB9168-MNT](#)  
**changed:** stefano.buonaiuto@mtf.ch 20010727  
**source:** RIPE

### **3.8.3 195.199.74.91**

The host 195.199.74.91 was logged using Synscan1.5/1.6 to scan port 21 of 7765 hosts of the MY.NET/16. He received replies from many hosts running FTP services and it can be expected that this IP will be involved in exploit attempts in the near future. As listed by [Ripe.net](#):

**inetnum:** 195.199.74.64 - 195.199.74.95  
**netname:** SULI-3507  
**descr:** Pedagogic Institute  
**descr:** Gyor  
**country:** HU  
**admin-c:** [FA2164-RIPE](#)  
**tech-c:** [FA2164-RIPE](#)  
**status:** ASSIGNED PA  
**mnt-by:** [RIPE-NCC-NONE-MNT](#)  
**changed:** hostmaster@elender.hu 20000419  
**source:** RIPE  
**route:** 195.199.0.0/16  
**descr:** MKM Net  
**origin:** [AS8512](#)  
**mnt-by:** [AS8512-MNT](#)  
**changed:** Nandor.Horvath@elender.hu 19971205  
**changed:** balazs.lengyak@elender.hu 20000508  
**source:** RIPE

### **3.8.4 195.101.94.208**

This IP appears in the OOS logs scanning specific addresses of the MY.NET network in what appears to be an Operating System fingerprinting attempt. The IP is also logged 461 times in the Scan logs and 458 times in the Alert log with the 'Queso fingerprinting' alert. Because of the type of information that can be gathered it can be assumed that this IP will reappear associated with an exploit or an exploit attempt. As listed by [Ripe.net](#):

**inetnum:** 195.101.94.0 - 195.101.94.255  
netname: FR-ECHO  
descr: Socite ECHO  
country: FR  
admin-c: [CR308-RIPE](#)  
tech-c: [CR308-RIPE](#)  
status: ASSIGNED PA  
notify: addr-reg@rain.fr  
mnt-by: [RAIN-TRANSPAC](#)  
changed: noc@rain.fr 19970514  
source: RIPE  
**route:** 195.101.94.0/24  
descr: ECHO  
**origin:** [AS8891](#)  
mnt-by: [OLEANE-NOC](#)  
changed: hostmaster@oleane.net 19980929  
source: RIPE

### **3.8.5 136.145.82.46**

This host appears 207 times in the Scan logs and 206 times in the Alert logs, triggering on the External RPC call rule. He is actively scanning port 111, which may be in advance of an exploit attempt. As listed by [ARIN](#):

OrgName: University of Puerto Rico  
OrgID: [UPR](#)  
NetRange: [136.145.0.0 - 136.145.255.255](#)  
CIDR: 136.145.0.0/16  
NetName: [CUN](#)  
NetHandle: [NET-136-145-0-0-1](#)  
Parent: [NET-136-0-0-0-0](#)  
NetType: Direct Assignment  
NameServer: UPR1.UPR.CLU.EDU  
NameServer: TRANTOR.UMD.EDU  
Comment:  
RegDate: 1989-08-29  
Updated: 1993-03-22  
TechHandle: [FGR-ARIN](#)  
TechName: Ramos, Felix  
TechPhone: 8092500000

### **3.9 Top Talkers**

While preparing this part of the report it became clear that a list generated on the total number of scans or alerts from the most prolific of IP's was not going to be of any real value. When divided into internal and external traffic it becomes easier to quantify.

The internal traffic is made up of mostly Peer-to-Peer and gaming traffic, while the external traffic is made up with a predominance of port 80 scanning, most likely from Internet worms scanning for web servers.

With this type of traffic removed from the list we can see a more true reflection of the real top talkers and what they are doing, as shown in Table 3.9.1. The large scans for port 21 were included because of the number of hosts running FTP and the security risk involved.

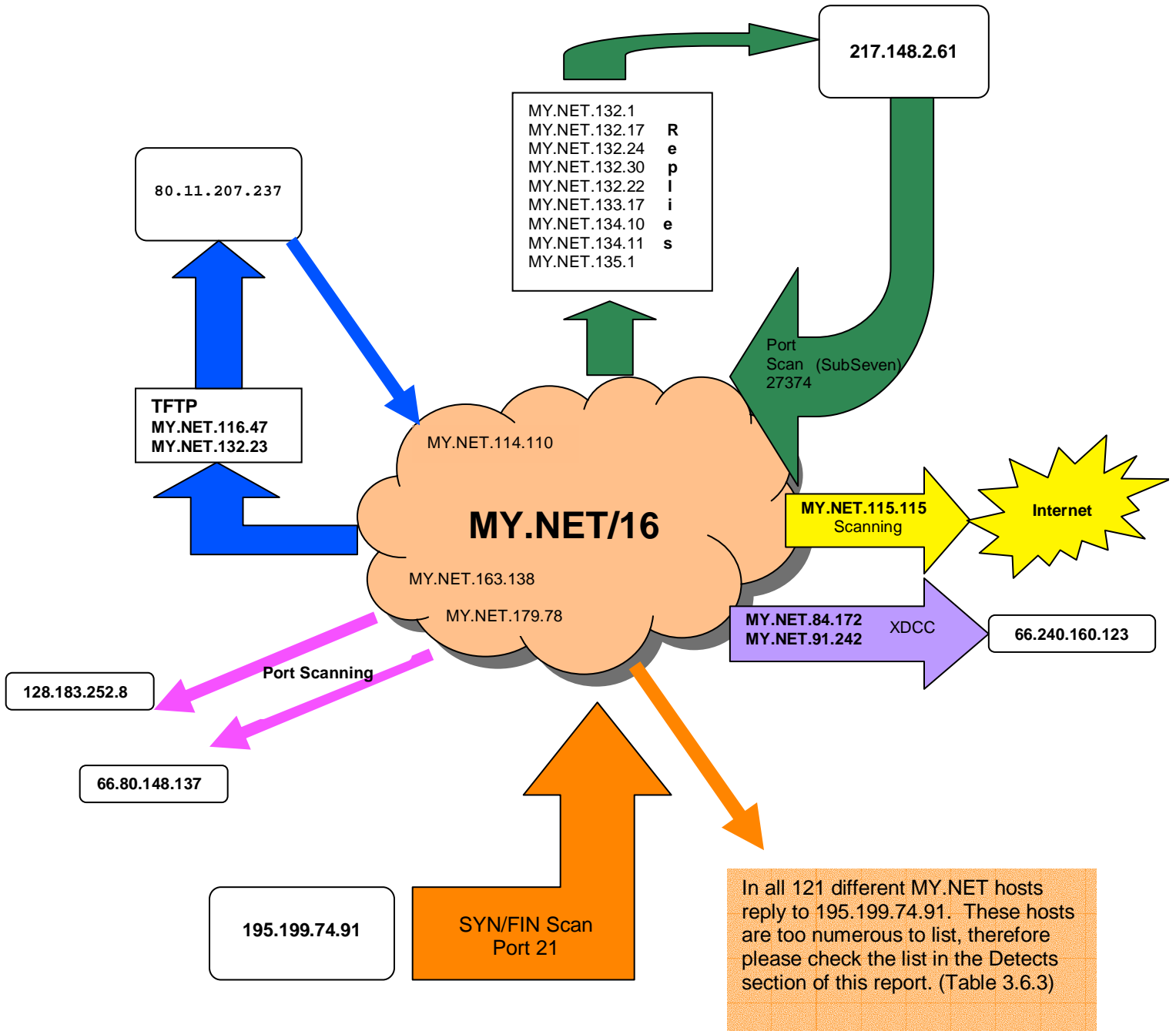
<b>Source IP</b>	<b>Total Events Logged</b>	<b>Type</b>
61.18.71.113	13718	Port 21 Scan
164.2.255.244	12659	Port 21 Scan
211.161.255.184	12322	Port 21 Scan
194.119.255.146	9641	Port 21 Scan
195.199.74.91	7765	SIN-FIN Scan
213.11.92.241	7425	Port 22 Scan
211.159.67.178	7381	Scan of port 1524, Trinoo Trojan
MY.NET.115.115	6979	Scan of ports 80, 137, 139, 445, 1433
205.188.228.129	4977	Scan of port 6970, Gatecrasher Trojan
80.11.207.237	1813	Scan of ports 80, 137, 139, 445, 1433

Table 3.9.1

Note that this list contains mostly external addresses. Once the regular web traffic, P2P traffic, and gaming traffic are removed, the most prolific talkers of interest are all external, with the one exception being MY.NET.115.115. Though there are other internal talkers, their number of events generated excluded them from this list.

### 3.10 Link Graph of Events of Interest

This diagram contains a very basic look at several events of interest that are of some concern to this analyst. All of the events are covered in depth in different parts of this report. All related events and replies from MY.NET hosts are delivered in the same colours. While being an over simplified look at events I believe it to be understandable for all knowledge levels of viewers.



### **3.11 Defensive Recommendations**

1. Upgrade Snort and Snort rule sets
2. Remove Snort HTTP pre-processor
3. Deploy internal IDS sensors
4. Install and upgrade anti virus software
5. Education of users in security practices
6. Installation of traffic content and prevention software(e.g. SurfinShield)
7. Auditing of FTP services
8. Review of Peer to Peer software usage
9. Base lining of servers and system hosts
10. Inspection of all suspected compromised hosts
11. Blocking of NETBIOS at firewall
12. Hire trained computer security staff for implementation and maintenance of IDS, and system security upgrades.
13. Implement and enforce an AUP.

While many recommendations have been made through out this report on specific events and vulnerabilities, there are several more recommendations that can be made that will increase the security level of the University.

1. There are several problems with the Snort IDS that should be addressed. Not having the rule set used by the University for their Snort sensor and having to try to match the alerts to the rules it became clear that the University is using a fairly out dated set of rules. It is imperative that the rule set be kept up dated on a regular basis in order to keep up with new exploits and attacks. New rule sets can be downloaded right from [Snort](#).
2. The HTTP pre-processor installed on Snort is generating a huge amount of traffic in the logs that are false positives. With the excessive amount of logs to comb through an analyst will quickly become swamped and may be diverted from a serious breach or exploit. With this in mind the pre-processor should be removed, though with the newer rule sets for Snort installed the nefarious traffic will be logged.
3. The logs provided for this report have no traffic going from MY.NET to MY.NET hosts. Obviously the sensor for the University is located outside of the firewalls and does not log internal traffic. Another sensor should be deployed internally to capture any traffic that might have internal hosts attacking internal victims. One can also use this sensor to correlate attacks logged on the external sensor to ensure that internal hosts have not been compromised.
4. The University should also ensure that anti virus software is installed and kept updated on all infrastructure hosts. The IT staff can configure automatic updates to ensure that all systems receive upgrades in a timely manner. Most

major vendors of anti virus software release updates at least once a week if not sooner. These AV programs can prevent virus infection as well as preventing many Trojans from being installed.

5. Users of the student network should also be required to run AV software and audits should be conducted to enforce this. Users that are found to not run updated AV should be removed from the network until such software is installed and upgraded. Education of users with respect to running AV software and securing their systems would also be beneficial. While being an extra burden on the IT staff the overall gains in security are well worth it.

6. Another solution to Trojans that might be considered is software that will run incoming traffic through a 'sandbox' before allowing it onto the network. Finjan's [SurfinShield Corporate](#) is an example of this type of software that will ensure malicious code, including Active X and executables, are prevented from being run on the network. While no product is 100% these types of products should be considered on top of updated anti virus software.

7. There are many FTP services being run on this University network. The IT staff should audit these systems to ensure that all vendor required security patches are installed and functioning. Systems that do not meet this standard should be removed until they are upgraded to ensure that exploitation does not happen. The University can still share its resources with others while protecting itself from un-authorized intrusions.

8. The student body is very active in the use of file sharing programs such as KaZaa and WinMX. There are several viruses that spread through peer-to-peer software such as the [W32/Benjamin.worm](#) and the [W32/Hobbit](#) family. The IT department should review the use of these programs for the security aspects, licensing legal issues, and bandwidth hogging concerns. Again insuring that anti virus software is installed will help prevent infection of users.

9. Implementing a ridged baseline for servers and hosts would allow the IT department to be on top of all system configuration issues such as security and patching. Periodic auditing of the network is also imperative to ensure no systems are left unprotected.

10. A complete and thorough inspection of all the suspected compromised and infected hosts in the MY.NET should take place as early as possible. This will also help to contain and prevent further infection and possible use of University property for illegal activities.

11. One should also consider the blocking of all NETBIOS ports, 137,138, 139, and 443, at the firewall to help prevent information gathering prior to an actual attack.

12. The hiring of dedicated computer security professionals is another option that should be explored by the University. This would take a great load off of the regular IT support staff and would allow the experts to concentrate on the security aspects of the network and hosts. These experts would be responsible for the deployment and maintenance of IDS and security upgrades. While expensive, the dollar cost should be outweighed by the potential gains in information security.

13. Implementing and enforcing an Acceptable Use Policy will reduce internal malicious activity, or it can at least give the IT staff the mechanism to remove the offender from the Network.

### **3.12 Conclusion**

In conclusion, the University is at an acceptable level in respect to security. With the large number of users and systems the overall level of exploit and compromise is low. However, if the University will implement the Defensive Recommendations proposed, the level of security and event logging will greatly improve allowing the University to operate in an uninterrupted and secure manner.

## **4. References**

### **The State of Intrusion Detection**

<http://www.busybox.net/>  
<http://trinix.sourceforge.net/>  
<http://www.bernd-roehling.de/trinix/doc/>  
<http://home.purplehaze.ch/~olivier/pub/writings/trinix.html>  
<http://www.usenix.org/publications/login/2000-12/pdfs/forte.pdf>  
<http://www.landfield.com/isn/mail-archive/1998/Sep/0105.html>

### **Synscan**

[http://www.simovits.com/trojans/tr\\_data/y1386.html](http://www.simovits.com/trojans/tr_data/y1386.html)  
<http://www.cve.mitre.org/cve/>  
<http://isc.incidents.org>  
[http://www.whitehats.ca/main/publications/external\\_pubs/scanner\\_fingerprints/scanner\\_fingerprints.html](http://www.whitehats.ca/main/publications/external_pubs/scanner_fingerprints/scanner_fingerprints.html)  
<http://www.brianhouk.com/papers/acl-cisco-bhouk.htm#eac>  
<http://www.ripe.net/>  
<http://www.psychoid.lam3rz.de/synscan1.6.tar.gz>  
<http://lists.jammed.com/incidents/2001/10/0104.html>  
<http://cert.uni-stuttgart.de/archive/intrusions/2002/07/msg00045.html>  
rfc 791

### **DdoS Attack on IRC Server**

<http://www.ripe.net/>  
<http://www.cve.mitre.org/>  
<http://www.treachery.net/tools/ports/lookup.cgi>  
<http://cert.uni-stuttgart.de/archive/incidents/2000/12/msg00156.html>  
<http://cert.uni-stuttgart.de/archive/incidents/2000/12/msg00165.html>  
<http://www.lunarstorm.se/>  
<http://www.sans.org/newlook/resources/IDFAQ/traffic.htm>  
<http://www.ash.udel.edu/ash/teacher/AUP.html>  
E-mail to Lunarworks in Sweden (Appendix A)

### **HTTP Connect Attempt**

<http://mtip.net/aware/MarkLachnietChecklist.pdf>  
<http://www.kb.cert.org/vuls/id/150227>  
<http://www.ripe.net/>  
<http://www.apnic.net/apnic-bin/whois2.pl>  
<http://www.dshield.org>  
<http://www.web-cache.com/Writings/Internet-Drafts/draft-luotonen-web-proxy-tunneling-01.txt>  
<http://www.ietf.org/rfc/rfc2817.txt>  
<http://www.ietf.org/rfc/rfc2616.txt>  
<http://www.ietf.org/rfc/rfc2068.txt>

## Analyze This

<http://www.faqs.org/rfcs/rfc793.html>  
<http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=ftp>  
[http://www.whitehats.ca/main/publications/external\\_pubs/scanner\\_fingerprints/scanner\\_fingerprints.html](http://www.whitehats.ca/main/publications/external_pubs/scanner_fingerprints/scanner_fingerprints.html)  
<http://archives.neohapsis.com/archives/snort/2000-01/0222.html>  
<http://lists.jammed.com/incidents/2001/05/0034.html>  
[http://www.finchhaven.com/pages/incidents/030102\\_udp\\_137.html](http://www.finchhaven.com/pages/incidents/030102_udp_137.html)  
[http://support.baynetworks.com/library/tpubs/html/router/soft1200/117358AA/B\\_39.HTM](http://support.baynetworks.com/library/tpubs/html/router/soft1200/117358AA/B_39.HTM)  
<http://www.j51.com/~sshay/tcpip/wins/wins.htm>  
<http://ourworld.compuserve.com/homepages/timothydevans/contents.htm>  
<http://www.dslreports.com/faq/4493>  
<http://www.russonline.net/tonikgin/EduHacking.html>  
<http://members.aol.com/lamesn/myhomepage/XDCC.htm>  
<http://www.astalavista.com/library/auditing/netbios/netbios-tutorial.txt>  
<http://security.duke.edu/cleaning/xdcc.html>  
<http://www.securiteam.com/securitynews/5ZP021575W.html>  
[http://www.infoslash.org/topic.asp?TOPIC\\_ID=194&FORUM\\_ID=7&CAT\\_ID=2&Forum\\_Title=Security+News&Topic\\_Title=Massive+EDU+Hacking+Tactics+Exposed](http://www.infoslash.org/topic.asp?TOPIC_ID=194&FORUM_ID=7&CAT_ID=2&Forum_Title=Security+News&Topic_Title=Massive+EDU+Hacking+Tactics+Exposed)  
<http://www-uxsup.csx.cam.ac.uk/security/probing/about/sunrpc.html>  
<http://www.sei.cmu.edu/str/descriptions/rpc.html>  
<http://www.sans.org/top20/#U1>  
<http://www.linux-nis.org/nis-howto/HOWTO/portmapper.html>

As well, the following books were used through out this assignment for information and guidance.

Intrusion Signatures and Analysis by Northcutt, Cooper, Fearnow, and Frederick

Tcp/IP Illustrated, Vol 1 by W. Richard Stevens

Network Intrusion Detection Third Edition by Stephen Northcutt and Judy Novak

Hacking Exposed, Third Edition by McClure, Scambray and Kurtz

SANS Track 4 Course Books

## Appendix A

From: Andreas Ödman [mailto:andreas@lunarworks.com]  
Sent: September 4, 2002 6:05 AM  
To: 'alwilliams@XXXXX.com'  
Subject: RE: Sent: 3 september 2002 02:17

> Good Day,  
>  
> I am an Intrusion Detection Analyst currently working on my  
> GCIA <<http://www.giac.org/>> certification for the SANS  
> Institute. I am working on a Network Detect that involves an  
> IP address that belongs to ripe.net and would like to ask a  
> few questions regarding this. If I have contacted the wrong  
> person, would you please pass this on the appropriate person,  
> or send me a contact. On Aug 1st our network IDS picked up  
> the reflection of a Distributed Denial of Service attack  
> against the IP 80.68.100.52 which Whois has listed as  
> belonging to you. Here is a small sample of what we saw:  
>  
> The date is Aug01 (MmmDD) timezone GMT and the filter is -nvX  
> and host 80.68.100.52  
>  
> 15:01:36.825486 80.68.100.52.6667 > xxx.xxx.160.10.1024: R  
> [tcp sum ok] 0:0(0) ack 3282650728 win 0 (ttl 109, id 2493, len 40)  
> ...  
> As we can find no stimulus from our network to solicit this  
> traffic I am sure our class B address block was spoofed in a  
> DDoS attack against your IP. I was hoping that you might  
> have some information about this IP that you could share with  
> me. Since the source port of the above log is 6667 (Internet  
> Relay Chat) I was hoping if you could tell me if this was  
> indeed using IRC, or if not, do you have any idea what the  
> attacker(s) might be trying to do? I have found reference to  
> this type of attack using these ports as far back as Dec of  
> 2000 as a DDoS to take down IRC servers. Any logs,  
> correlation, or answers that you could send me would be  
> greatly appreciated,

Yes, you are correct. 80.68.100.52 is, or now was, the ip-adress for our irc-server. It now resides at 80.68.xxx.xxx (irc.lunarstorm.se). We are aware of this problem and in our best effort trying to fend this off. Our irc-community is small, and only serves as a small part of our larger community [www.lunarstorm.se](http://www.lunarstorm.se). We don't have any logs for this incident.

regards,  
Andreas Ödman  
LunarWorks AB

---

Södra Hamnvägen 2	Tel.....: +46 340 64 11 00
Box 50, 432 22 Varberg	Mob.....: +46 706 01 13 18

---

## Appendix B

### TCPDump

```
00:01:47.572889 61.172.246.78.43877 > xxx.xxx.210.179.80: S [tcp sum
ok] 150626275:150626275(0) win 1024 (ttl 44, id 35526, len 40)
0x0000      4500 0028 8ac6 0000 2c06 79d2 3dac f64e  E..(.....,y.=..N
0x0010      xxxx d2b3 ab65 0050 08fa 5fe3 08fa 5fe3  .....e.P._..._.
0x0020      5002 0400 a43a 0000 0000 0000 0000      P.....:.....
```

```
00:01:47.572889      Date/Time
61.172.246.78      Source IP
43877      Source Port
xxx.xxx.210.179      Destination IP
80      Destination Port
S [tcp sum ok]      TCP Flag set(SYN) with TCP Checksum OK
150626275:150626275      Sequence:Acknowledgement numbers
win 1024      Window size
id 35526      Packet ID number
len 40      Packet length in bytes
```

### Ngrep Output

```
T 2002/09/01 00:00:19.628022 61.172.246.78:58416 -> xxx.xxx.210.203:3128 [S]
```

```
T      Protocol (TCP)
2002/09/01 00:00:19.628022      Date/Time
61.172.246.78      Source IP
58416      Source Port
xxx.xxx.210.203      Destination IP
3128      Destination Port
[S]      TCP Flag set (SYN)
```

### Shadow Logs

```
00:00:19.628022 61.172.246.78.58416 > xxx.xxx.210.203.3128: S 1908257516:1908257516(0) win 1024
```

```
00:00:19.628022      Time
61.172.246.78      Source IP
58416      Source Port
xxx.xxx.210.203      Destination IP
3128      Destination Port
S      TCP Flag set (SYN)
1908257516:1908257516      Sequence:Acknowledgement numbers
win 1024      Window size in bytes
```

## Appendix C

### The Perl Scripts

```
#!/usr/bin/perl
# Convert alert.DDDDDD files to csv

while(<>) {
    next unless m/^\d/;
    next if m/spp_portscan/;

    chomp;
    ($date_time,$alert,$addrs) = split(/\s+\Q[**]\E\s+/);

    ($source, $dest) = ($addrs =~ m/(.*)\s+-->\s+(.*)/);
    ($date,$time) = split(/-/, $date_time);
    ($source_ip, $source_port) = split(/:/, $source);
    ($dest_ip, $dest_port) = split(/:/, $dest);

    print "$date,$time,$alert,$source_ip,$source_port,$dest_ip,$dest_port\n";
}
```

+++++

```
#!/usr/bin/perl
# Convert scans.DDDDDD files to csv

%months = ("Mar" => 3);

while (<>)
{
    next unless m/^[A-Z]/;
    chomp;
    ($month,$day,$time,$source,$dir,$dest,$proto,$flags) = split;

    $month = $months{$month};
    $date = sprintf("%02d/%02d", $month, $day);
    ($src_ip,$src_port) = split(/:/,$source);
    ($dst_ip,$dst_port) = split(/:/,$dest);

    print "$date,$time,$src_ip,$src_port,$dst_ip,$dst_port,$proto,$flags\n";
}
```