Global Information Assurance Certification Paper

GCIA Practical

Intrusion Detection In-Depth

Cory Steers
Version 3.1
May 05, 2002

Part 1 – Describe the State of Intrusion Detection

Intrusion Detection In-Depth

Cory Steers
Version 3.1
May 05, 2002

IDS – Eliminating Telnet Negotiation Strings in Telnet and FTP

Overview

Telnet and FTP sessions can contain telnet negotiation strings. By default all telnet and ftp clients and servers can map their respective terminal to the "network virtual terminal" (NVT), and the inverse. NVT is considered the lowest common denominator needed to allow useful communication; however, both clients and servers can request additional features to allow for more powerful communication. These requests are done via in-band signaling, and any byte of value 0xff (decimal 255) is the signal for this. I'll talk a little more about this later. For more detail, see RFC 854 [Postel and Reynolds 1983a].

Crackers soon discovered that these in-band requests could be used to fool Intrusion Detection Systems (IDS). Telnet and FTP applications see no differences between the strings "abcd" and "ab<in-band signal byte(s)>cd"; however, traditionally IDS products use string matching or pattern matching to look for attacks. It's extremely difficult with a simple regular expression pattern match to catch both strings above, without a large number of false positives.

This paper will demonstrate and discuss how a program called SideStep tries to evade IDS detection using in-band signaling, and how newer versions of snort are able to overcome this obstacle.

SideStep

Sidestep was written by Robert Graham and can be found at http://www.robertgraham.com/tmp/sidestep.html. Robert wrote this as a sample tool to demonstrate IDS evasion techniques. Sidestep has 6 different attacks, and can perform them in three modes. Normal mode does nothing to fool an IDS, while evasive mode uses (as the name implies) evasion techniques to perform the exploits without being detected.

There is also a "false" mode to test your IDS for a false positive, but this mode will not be discussed here.

SNORT

Snort is a popular open source network IDS (NIDS), and can be found at http://www.snort.org/. It was initially written by Marty Roesch to overcome some limitations he had with tcpdump, but people all over the world quickly started helping with enhancements and fixes. Today, Marty has a company called SourceFire, and snort is at the heart of his products.

CWD ~root exploit explained

In short, an FTP server vulnerable to this exploit will give an attacker access to the entire file system of that server with root permissions. One way to do this is to:
- ftp ftp.victim.com
- Hit enter a couple of times to fail the login process
- Type "quote user ftp" – equivalent to sending anonymous as the user ID
- Type "quote cwd ~root"
- Type "quote pass ftp" – equivalent to sending an informational password for an anonymous login

Now, you are logged in anonymously, your current directory is "/", and you have root permissions on that server.

According to Thomas Veit (http://www.imn.htwk-leipzig.de/~veit/thesis/Vulnerabilities/Application.specific/FTP/) this exploit relies on vulnerabilities in versions of the wu-ftpd FTP server created prior to 1993 (this does not mean that wu-ftpd is/was the only FTP server vulnerable to this). I'm not sure how accurate this information is, but this suggests that the exploit has been around for a while, and any recent version of an FTP server should not fall victim to it. I have; however, noticed that on my default load of Red Hat Linux, version 7.2, wu-ftpd's default configuration lets me change to the root

directory, but I do not have root permission to any of the files. I would only be able to read files with world readable permissions, and write over files with world write able permissions. I was not able to find out any information about this specific issue. CERT Advisory CA-2001-33 states there are multiple vulnerabilities in wu-ftpd, and talks about two of them. In any case, this is "open" enough to allow the SideStep program to work. That's enough for discussion in this paper.

The Details

Normal ftp

First I'll analyze Sidestep attempting the CWD ~root exploit in "normal" mode. Take a look at the next few packets below.

```
17:03:28.989767 192.168.0.181.1791 > 192.168.0.221.ftp: S
3138537281:3138537281(0) win 64240 <m
ss 1460> (DF)
0x0000   4500 002c db7e 4000 8006 4f75 c0a8 00b5        E..,.~@...Ou....
0x0010   c0a8 00dd 06ff 0015 bb12 4741 0000 0000        ..........GA....
0x0020   6002 faf0 c3f5 0000 0204 05b4 0000             `.............

17:03:28.989876 192.168.0.221.ftp > 192.168.0.181.1791: S
4007965980:4007965980(0) ack 31385372
82 win 5840 <mss 1460> (DF)
0x0000   4500 002c 0000 4000 4006 6af4 c0a8 00dd        E..,..@.@.j.....
0x0010   c0a8 00b5 0015 06ff eee4 b51c bb12 4742        ..............GB
0x0020   6012 16d0 0404 0000 0204 05b4                  `...........

17:03:28.990090 192.168.0.181.1791 > 192.168.0.221.ftp: . ack 1 win 64240 (DF)
0x0000   4500 0028 dc7e 4000 8006 4e79 c0a8 00b5        E..(.~@...Ny....
0x0010   c0a8 00dd 06ff 0015 bb12 4742 eee4 b51d        ..........GB....
0x0020   5010 faf0 37a0 0000 0000 0000 0000             P...7.........

17:03:29.010464 192.168.0.221.ftp > 192.168.0.181.1791: P 1:61(60) ack 1 win
5840 (DF)
0x0000   4500 0064 2bda 4000 4006 3ee2 c0a8 00dd        E..d+.@.@.>.....
0x0010   c0a8 00b5 0015 06ff eee4 b51d bb12 4742        ..............GB
0x0020   5018 16d0 ca20 0000 3232 3020 686f 7374        P.......220.host
0x0030   6e61 6d65 2e6c 6f63 616c 2046 5450 2073        name.local.FTP.s
0x0040   6572 7665 7220 2856 6572 7369 6f6e 2077        erver.(Version.w
0x0050   752d 322e 362e 312d 3230 2920 7265 6164        u-2.6.1-20).read
0x0060   792e 0d0a                                      y...

17:03:29.011777 192.168.0.181.1791 > 192.168.0.221.ftp: P 1:17(16) ack 61 win
64180 (DF)
0x0000   4500 0038 de7e 4000 8006 4c69 c0a8 00b5        E..8.~@...Li....
0x0010   c0a8 00dd 06ff 0015 bb12 4742 eee4 b559        ..........GB...Y
0x0020   5018 fab4 afab 0000 5553 4552 2061 6e6f        P.......USER.ano
```

```
0x0030   6e79 6d6f 7573 0d0a                              nymous..

17:03:29.014057 192.168.0.221.ftp > 192.168.0.181.1791: P 61:129(68) ack 17
win 5840 (DF)
0x0000   4500 006c 2bdc 4000 4006 3ed8 c0a8 00dd        E..l+.@.@.>.....
0x0010   c0a8 00b5 0015 06ff eee4 b559 bb12 4752        ...........Y..GR
0x0020   5018 16d0 3184 0000 3333 3120 4775 6573        P...1...331.Gues
0x0030   7420 6c6f 6769 6e20 6f6b 2c20 7365 6e64        t.login.ok,.send
0x0040   2079 6f75 7220 636f 6d70 6c65 7465 2065        .your.complete.e
0x0050   2d6d 6169 6c20 6164 6472 6573 7320 6173        -mail.address.as
0x0060   2070 6173 7377 6f72 642e 0d0a                  .password...
17:03:29.015311 192.168.0.181.1791 > 192.168.0.221.ftp: P 17:32(15) ack 129
win 64112 (DF)
0x0000   4500 0037 df7e 4000 8006 4b6a c0a8 00b5        E..7.~@...Kj....
0x0010   c0a8 00dd 06ff 0015 bb12 4752 eee4 b59d        ..........GR....
0x0020   5018 fa70 e441 0000 5041 5353 204d 6f7a        P..p.A..PASS.Moz
0x0030   696c 6c61 400d 0a                              illa@..
17:03:29.019127 192.168.0.221.ftp > 192.168.0.181.1791: P 129:177(48) ack 32
win 5840 (DF)
0x0000   4500 0058 2bdd 4000 4006 3eeb c0a8 00dd        E..X+.@.@.>.....
0x0010   c0a8 00b5 0015 06ff eee4 b59d bb12 4761        ..............Ga
0x0020   5018 16d0 6f03 0000 3233 3020 4775 6573        P...o...230.Gues
0x0030   7420 6c6f 6769 6e20 6f6b 2c20 6163 6365        t.login.ok,.acce
0x0040   7373 2072 6573 7472 6963 7469 6f6e 7320        ss.restrictions.
0x0050   6170 706c 792e 0d0a                            apply...

17:03:29.020092 192.168.0.181.1791 > 192.168.0.221.ftp: P 32:43(11) ack 177
win 64064 (DF)
0x0000   4500 0033 e07e 4000 8006 4a6e c0a8 00b5        E..3.~@...Jn....
0x0010   c0a8 00dd 06ff 0015 bb12 4761 eee4 b5cd        ..........Ga....
0x0020   5018 fa40 4407 0000 4357 4420 7e72 6f6f        P..@D...CWD.~roo
0x0030   740d 0a                                        t..
17:03:29.021033 192.168.0.221.ftp > 192.168.0.181.1791: P 177:206(29) ack 43
win 5840 (DF)
0x0000   4500 0045 2bde 4000 4006 3efd c0a8 00dd        E..E+.@.@.>.....
0x0010   c0a8 00b5 0015 06ff eee4 b5cd bb12 476c        ..............Gl
0x0020   5018 16d0 3d5d 0000 3235 3020 4357 4420        P...=]..250.CWD.
0x0030   636f 6d6d 616e 6420 7375 6363 6573 7366        command.successf
0x0040   756c 2e0d 0a                                   ul...
```

In case you don't recognize it, the above information is the dump of a sniffer trace using tcpdump with the hex and ASCII flags set. I've highlighted the interesting bytes in **bold blue**. If you've seen a lot of trace dumps before, you'll be very familiar with this. You may also be thinking that it looks pretty normal, so what's the big deal? Well, you're right. After all, this was the non-evasive attempt, remember?

You may have noticed that Robert Graham programmed sidestep to send the user, pass, and cwd commands in a different order than Thomas Veit specified in his document mentioned above. I really have no explanation for this. Both methods worked for getting logged in and changed to the "/" directory on my ftp server, but you're less likely to get error messages for failing to login with Robert's method. From a

"stealth" perspective, if you can evade IDS, evading the ftp server logs would be a good idea as well.

Next, I ran snort against the sniffer trace in IDS mode, with the "telnet_decode" preprocessor commented out in the configuration file. I'll discuss this preprocessor more in a minute. It reports that it found the FTP CWD ~root exploit. See below.

```
[**] [1:336:2] FTP CWD ~root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/31-17:03:29.020092 192.168.0.181:1791 -> 192.168.0.221:21
TCP TTL:128 TOS:0x0 ID:57470 IpLen:20 DgmLen:51 DF
***AP*** Seq: 0xBB124761  Ack: 0xEEE4B5CD  Win: 0xFA40  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0082]
[Xref => http://www.whitehats.com/info/IDS318]
```

Evasive FTP

Now we'll analyze Sidestep attempting the CWD ~root exploit in "evasive" mode. Take a look at the next few packets below.

```
17:04:27.174425 192.168.0.181.1793 > 192.168.0.221.ftp: S
3153102063:3153102063(0) win 64240 <m
ss 1460> (DF)
0x0000   4500 002c 4d7f 4000 8006 dd74 c0a8 00b5        E..,M.@....t....
0x0010   c0a8 00dd 0701 0015 bbf0 84ef 0000 0000        ................
0x0020   6002 faf0 8567 0000 0204 05b4 0000             `....g........

17:04:27.174534 192.168.0.221.ftp > 192.168.0.181.1793: S
4075965553:4075965553(0) ack 31531020
64 win 5840 <mss 1460> (DF)
0x0000   4500 002c 0000 4000 4006 6af4 c0a8 00dd        E..,..@.@.j.....
0x0010   c0a8 00b5 0015 0701 f2f2 4c71 bbf0 84f0        ..........Lq....
0x0020   6012 16d0 2a13 0000 0204 05b4                  `...*.......

17:04:27.174743 192.168.0.181.1793 > 192.168.0.221.ftp: . ack 1 win 64240 (DF)
0x0000   4500 0028 4e7f 4000 8006 dc78 c0a8 00b5        E..(N.@....x....
0x0010   c0a8 00dd 0701 0015 bbf0 84f0 f2f2 4c72        ..............Lr
0x0020   5010 faf0 5daf 0000 0000 0000 0000             P...].........


17:04:27.194998 192.168.0.221.ftp > 192.168.0.181.1793: P 1:61(60) ack 1 win
5840 (DF)
0x0000   4500 0064 f9f3 4000 4006 70c8 c0a8 00dd        E..d..@.@.p.....
0x0010   c0a8 00b5 0015 0701 f2f2 4c72 bbf0 84f0        ..........Lr....
0x0020   5018 16d0 f02f 0000 3232 3020 686f 7374        P..../..220.host
0x0030   6e61 6d65 2e6c 6f63 616c 2046 5450 2073        name.local.FTP.s
0x0040   6572 7665 7220 2856 6572 7369 6f6e 2077        erver.(Version.w
0x0050   752d 322e 362e 312d 3230 2920 7265 6164        u-2.6.1-20).read
0x0060   792e 0d0a                                      y...
```

```
17:04:27.196311 192.168.0.181.1793 > 192.168.0.221.ftp: P 1:19(18) ack 61 win
64180 (DF)
0x0000   4500 003a 507f 4000 8006 da66 c0a8 00b5        E..:P.@....f....
0x0010   c0a8 00dd 0701 0015 bbf0 84f0 f2f2 4cae        ..............L.
0x0020   5018 fab4 e3b8 0000 55ff f153 4552 2061        P.......U..SER.a
0x0030   6e6f 6e79 6d6f 7573 0d0a                       nonymous..

17:04:27.196493 192.168.0.221.ftp > 192.168.0.181.1793: . ack 19 win 5840 (DF)
0x0000   4500 0028 f9f4 4000 4006 7103 c0a8 00dd        E..(..@.@.q.....
0x0010   c0a8 00b5 0015 0701 f2f2 4cae bbf0 8502        ..........L.....
0x0020   5010 16d0 4182 0000                            P...A...

17:04:27.198576 192.168.0.221.ftp > 192.168.0.181.1793: P 61:129(68) ack 19
win 5840 (DF)
0x0000   4500 006c f9f5 4000 4006 70be c0a8 00dd        E..l..@.@.p.....
0x0010   c0a8 00b5 0015 0701 f2f2 4cae bbf0 8502        ..........L.....
0x0020   5018 16d0 5791 0000 3333 3120 4775 6573        P...W...331.Gues
0x0030   7420 6c6f 6769 6e20 6f6b 2c20 7365 6e64        t.login.ok,.send
0x0040   2079 6f75 7220 636f 6d70 6c65 7465 2065        .your.complete.e
0x0050   2d6d 6169 6c20 6164 6472 6573 7320 6173        -mail.address.as
0x0060   2070 6173 7377 6f72 642e 0d0a                  .password...

17:04:27.199830 192.168.0.181.1793 > 192.168.0.221.ftp: P 19:36(17) ack 129
win 64112 (DF)
0x0000   4500 0039 517f 4000 8006 d967 c0a8 00b5        E..9Q.@....g....
0x0010   c0a8 00dd 0701 0015 bbf0 8502 f2f2 4cf2        ..............L.
0x0020   5018 fa70 184d 0000 50ff f141 5353 204d        P..p.M..P..ASS.M
0x0030   6f7a 696c 6c61 400d 0a                         ozilla@..

17:04:27.203681 192.168.0.221.ftp > 192.168.0.181.1793: P 129:177(48) ack 36
win 5840 (DF)
0x0000   4500 0058 f9f6 4000 4006 70d1 c0a8 00dd        E..X..@.@.p.....
0x0010   c0a8 00b5 0015 0701 f2f2 4cf2 bbf0 8513        ..........L.....
0x0020   5018 16d0 950e 0000 3233 3020 4775 6573        P.......230.Gues
0x0030   7420 6c6f 6769 6e20 6f6b 2c20 6163 6365        t.login.ok,.acce
0x0040   7373 2072 6573 7472 6963 7469 6f6e 7320        ss.restrictions.
0x0050   6170 706c 792e 0d0a                            apply...

17:04:27.204659 192.168.0.181.1793 > 192.168.0.221.ftp: P 36:53(17) ack 177
win 64064 (DF)
0x0000   4500 0039 527f 4000 8006 d867 c0a8 00b5        E..9R.@....g....
0x0010   c0a8 00dd 0701 0015 bbf0 8513 f2f2 4d22        ..............M"
0x0020   5018 fa40 940c 0000 43ff f157 4420 7eff        P..@....C..WD.~.
0x0030   f172 6fff f16f 740d 0a                         .ro..ot..

17:04:27.205540 192.168.0.221.ftp > 192.168.0.181.1793: P 177:206(29) ack 53
win 5840 (DF)
0x0000   4500 0045 f9f7 4000 4006 70e3 c0a8 00dd        E..E..@.@.p.....
0x0010   c0a8 00b5 0015 0701 f2f2 4d22 bbf0 8524        ..........M"...$
0x0020   5018 16d0 6362 0000 3235 3020 4357 4420        P...cb..250.CWD.
0x0030   636f 6d6d 616e 6420 7375 6363 6573 7366        command.successf
0x0040   756c 2e0d 0a                                   ul...
```

Again, we have a dump of another sniffer trace, using tcpdump. As before, I've highlighted the interesting bytes in **bold blue**; however, there are some extra interesting bytes in this dump, so I put them in **bold red**.

These extra, non printable, bytes that break up words like USER, PASS, and CWD, are the "in-band" telnet negotiation commands. The 0xff byte is called ICA, for "interpret as command". The following byte, is the command. In this case, the 0xf1 byte is called NOP, for "no operation". Essentially, SideStep is telling the ftp server to treat the 0xf1 byte as a command, but then the command is to do nothing. The ftp server is OK with this, and still sees the words like USER and PASS correctly; however, the IDS sees 0x43fff15744 instead of 0x425744 for the text CWD. In a simple pattern matching scenario, these look very different.

So, when I ran snort against this trace, with the same configuration file (telnet_decode preprocessor turned off), it did not report any alerts. It simply didn't see the "CWD ~root" command.

The Solution

Snort overcomes this issue with its telnet_decode preprocessor. Its sole purpose is to look for these telnet commands and strip them out of the data. Then, when the telnet rules are applied, they correctly catch the attempted attack. After changing snort.conf to include the telnet_decode preprocessor, snort was able to flag the evasive trace as a possible attack. The compressed tar file of rules I downloaded from snort.org contained a snort.conf with telnet_decode turned on by default; however, you should make sure this is enabled in your configuration.

Snort has other preprocessors such as http_decode and rpc_decode. To discuss these, is outside the scope of this paper, but they perform the same service as telnet_decode. For example, most popular web servers support Unicode characters. You can read about Unicode here. Attackers sometimes use Unicode to attempt to fool IDS. If the IDS is looking for the word "attack" in a URL, "a%74%74ack" would not be found by the IDS, but have the same affect on the web server. The http_decode preprocessor translates Unicode characters to their ASCII equivalent. Sometimes attackers use this representation of characters

to attempt to evade and IDS looking for a particular string

Conclusion

In summary, I have discussed in detail a way to attempt to evade IDS detection, and how one IDS solution (Snort) overcomes the evasion technique. I'm sure other IDS solutions have similar methods for handling this type of evasion. Then again, I have no doubt there are also other IDS solutions that do not handle this type of situation.

References

SideStep: IDS evasion tool

Robert Graham
February 2001
http://www.robertgraham.com/tmp/sidestep.html

TCP/IP Illustrated, Volume 1

W. Richard Stevens
Addison-Wesley
ISBN: 0-201-63346-9

RFC 854

J. Postel and J. Reynolds
May 1983, Telnet Protocol Specification
http://www.ietf.org/rfc/rfc0854.txt?number=854

Snort

Marty Roesch
http://www.snort.org
http://www.sourcefire.com

Vulnerability Database

Thomas Veit
March 16, 1998
http://www.imn.htwk-leipzig.de/~veit/thesis/Vulnerabilities/Application.specific/FTP/

Part 2 – Network Detects

Intrusion Detection In Depth

GCIA Practical Assignment
Cory Steers
Version 3.1

Detect Number 1

1. Source of Trace

This trace was taken from a DSL connection with a simple topology. A hub connects the DSL modem, the victim machine, and the sniffer. The victim machine was a laptop running an un-patched server install of Red Hat 6.1.

2. Detect was generated by

This detect was generated by Snort version 1.8.6 with a standard configuration file. The only alert that Snort fired on was ICMP Traffic, but there was a lot of additional related TCP traffic.

The Alert:

```
[**] [1:480:2] ICMP PING speedera [**]
[Classification: Misc activity] [Priority: 3]
05/07-05:07:13.673522 4.47.159.59 -> 4.47.159.186
ICMP TTL:127 TOS:0x0 ID:64442 IpLen:20 DgmLen:104
Type:8  Code:0  ID:1280   Seq:39937  ECHO
```

The rule in icmp.rules that generated the alert:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING speedera";
content: "|3839 3a3b 3c3d 3e3f|"; depth: 100; itype: 8;  sid:480;
classtype:misc-activity; rev:2;)
```

The packet that caused the rule to fire:

```
05/07-05:07:13.673522 4.47.159.59 -> 4.47.159.186
ICMP TTL:127 TOS:0x0 ID:64442 IpLen:20 DgmLen:104
Type:8  Code:0  ID:1280   Seq:39937  ECHO
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ................
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
40 41 42 43 44 45 46 47 48 49 4A 4B              @ABCDEFGHIJK
```

3. Probability the Source Address was spoofed

The probability that the source address was spoofed is minimal. The traffic coming from this address consisted of TCP traffic as well as icmp. A three-way-handshake was completed making it all but impossible for the traffic to be spoofed.

4. Description of Attack

Speedera, among other things offers a service called "Global Traffic Management" (GTM). If you perform a DNS lookup to resolve the name of one of their GTM customers, Speedera will ping you. Based on latency information from their pings, they will return an address that is closest to your machine geographically. See http://www.linuxsecurity.com/articles/firewalls_article-2064.html for more details.

That is more of a description of Speedera than a description of the attack. This attack has both reconnaissance and attack characteristics. The traffic starts out with a single echo request followed by about 30 minutes of silence. Then, we get another echo request followed by a connection to the SMTP port and more silence. This scenario repeats itself for another hour. Except for the 1st time, the amount of silence is seconds instead of minutes. At this point, there is never more than one connection attempt per second. Then, we get several seconds worth of echo requests where the packet size is gradually increasing … about 32 bytes each time. This continues for about a minute and then suddenly there are multiple requests in the same second for TCP 389. The end looks like a high speed port scan trying to find any open services.

Most of the traffic seems like recon, but the increasing packet size of the echo requests and the explosion of TCP 389 traffic might be considered malicious. I found this detect very confusing. At times, the attacker was pinging once every 3 or 4 seconds, but at other times he was sending me 400+ byte pings or flooding me with LDAP requests. Seems unusual to connect to an open port multiple times and never attempt any type of compromise. I find it equally unusual to connect to a closed port repeatedly.

5. Attack Mechanism

You don't automatically assume this was an attack. In fact, if it wasn't for the fact that this packet didn't originate from a Speedera network,

you would probably just ignore this alert. If you look at the full contents of the binary trace; however, you'll see that there was a lot of activity from this source address that snort didn't alert on. This type of situation is a good example of why you should record and keep full traces in addition to alerting in NIDS mode. Below are some packet snippets of interest from the trace.

Snort dump of Speedera alert

```
05/07-05:07:13.673522 4.47.159.59 -> 4.47.159.186
ICMP TTL:127 TOS:0x0 ID:64442 IpLen:20 DgmLen:104
Type:8  Code:0  ID:1280   Seq:39937  ECHO
0x0000: 00 50 04 F2 34 74 00 02 3B 00 AB 15 08 00 45 00  .P..4t..;.....E.
0x0010: 00 68 FB BA 00 00 7F 01 F8 86 04 2F 9F 3B 04 2F  .h........./.;./
0x0020: 9F BA 08 00 D3 54 05 00 9C 01 00 01 02 03 04 05  .....T..........
0x0030: 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15  ................
0x0040: 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25  .......... !"#$%
0x0050: 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35  &'()*+,-./012345
0x0060: 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45  6789:;<=>?@ABCDE
0x0070: 46 47 48 49 4A 4B                                FGHIJK
```

Snort dump of suspicious smtp traffic

```
05/07-04:59:55.348266 4.47.159.59:2240 -> 4.47.159.186:25
TCP TTL:127 TOS:0x0 ID:44984 IpLen:20 DgmLen:48 DF
******S* Seq: 0x281AC0E  Ack: 0x0  Win: 0x16D0  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

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

05/07-04:59:55.348266 4.47.159.186:25 -> 4.47.159.59:2240
TCP TTL:64 TOS:0x0 ID:2244 IpLen:20 DgmLen:48 DF
***A**S* Seq: 0x254828E7  Ack: 0x281AC0F  Win: 0x7D78  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

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

05/07-04:59:55.388262 4.47.159.59:2240 -> 4.47.159.186:25
TCP TTL:127 TOS:0x0 ID:45240 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x281AC0F  Ack: 0x254828E8  Win: 0x16D0  TcpLen: 20

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

05/07-04:59:55.388262 4.47.159.59:2240 -> 4.47.159.186:25
TCP TTL:127 TOS:0x0 ID:45496 IpLen:20 DgmLen:40 DF
***A***F Seq: 0x281AC0F  Ack: 0x254828E8  Win: 0x16D0  TcpLen: 20

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

05/07-04:59:55.388262 4.47.159.186:25 -> 4.47.159.59:2240
TCP TTL:64 TOS:0x0 ID:2245 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x254828E8  Ack: 0x281AC10  Win: 0x7D78  TcpLen: 20

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

```
05/07-04:59:55.898210 4.47.159.186:25 -> 4.47.159.59:2240
TCP TTL:64 TOS:0x0 ID:2249 IpLen:20 DgmLen:126 DF
***AP*** Seq: 0x254828E8  Ack: 0x281AC10  Win: 0x7D78  TcpLen: 20
32 32 30 20 6C 6F 63 61 6C 68 6F 73 74 2E 6C 6F  220 localhost.lo
63 61 6C 64 6F 6D 61 69 6E 20 45 53 4D 54 50 20  caldomain ESMTP
53 65 6E 64 6D 61 69 6C 20 38 2E 39 2E 33 2F 38  Sendmail 8.9.3/8
2E 39 2E 33 3B 20 54 75 65 2C 20 37 20 4D 61 79  .9.3; Tue, 7 May
20 32 30 30 32 20 30 39 3A 35 39 3A 33 30 20 2D   2002 09:59:30 -
30 35 30 30 0D 0A                                0500..

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

05/07-04:59:55.898210 4.47.159.186:25 -> 4.47.159.59:2240
TCP TTL:64 TOS:0x0 ID:2250 IpLen:20 DgmLen:40 DF
***A***F Seq: 0x2548293E  Ack: 0x281AC10  Win: 0x7D78  TcpLen: 20

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

05/07-04:59:55.938206 4.47.159.59:2240 -> 4.47.159.186:25
TCP TTL:127 TOS:0x0 ID:46008 IpLen:20 DgmLen:40 DF
*****R** Seq: 0x281AC10  Ack: 0x252B8ED5  Win: 0x0  TcpLen: 20

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

05/07-04:59:55.948205 4.47.159.59:2240 -> 4.47.159.186:25
TCP TTL:127 TOS:0x0 ID:46264 IpLen:20 DgmLen:40
*****R** Seq: 0x281AC10  Ack: 0x281AC10  Win: 0x0  TcpLen: 20

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

The types of traffic coming from this same source varies from different types of icmp packets to smtp, snmp, ldap, pop3, http, imap, and more. There were at least three different types of ping clients reported by snort, but those could have been false positives. I'm surprised snort didn't report some type of nmap scan or something. Even though there were gaps in the trace, there were too many periods of high traffic to be a low and slow "under the radar" scan.

The various popular tcp ports being accessed continues to lead you down the path of scan/reconnaissance; however, this is repeated 4 times identically over a 20 minute period. Each time, it seems like they are looking for a listening port. The client issues no protocol commands … just a tcp reset.

6. Correlations

I was not able to correlate this to anything specific. It's pretty obvious that this is some type of scan, but since snort was configured to

only catch traffic to the victim machine, it's not immediately clear whether or not this was part of a network scan or a specific scan of just this host. It's also not clear whether this was a new, never before seen, tool being used to perform the scan or just some clever wrappers around something like nmap in order to fool the port scan detection in my IDS. I scanned the internet looking for other people experiencing the same type of traffic, but I had no luck.

7. Evidence of active Targeting.

There is quite a bit of evidence to support Active Targeting; however, since my IDS was only looking for traffic to my IP address and the connection was a DSL line, it's not conclusive that it was active targeting. My IP address could have just been one of many on the DSL range of IP's.

8. Severity

Criticality = 1. This is just a "extra" system of mine used for experimenting. If I had to completely reload it due to a compromise, it wouldn't cost me anything but time.

Lethality = 3. Even though the traffic seems like just reconnaissance type traffic, it was severe enough to suspect I'll see this person again with more malicious traffic.

System Countermeasures = 1. This was an un-patched load of Red Hat Linux 6.1. I don't specifically remember when 6.1 came out, but it's rapidly approaching at least 2 years old. There were no firewall rules and no tcpd (TCP wrappers) protection on the box. The only way to make it less secure is to put Windows 95 on it. ☺

Network Countermeasures = 1. Again, this is a home "class" DSL connection with no firewall of any type in front of it.

Severity = (1 + 3) – (1 + 1) = 4 – 2 = 2.

9. Defensive Recommendation

A few simple tweaks would keep this system secure from attackers. First you always want to keep your system patched. Since this is a Red Hat system, Red Hat's up2date service would easily facilitate this. Next, turn off any unused services. If this is just a client machine, everything should be turned off. Perhaps something like sshd could be left on, if remote management was needed. Finally, some type of firewall should be protecting this box. A network based firewall is always an option, but since this is a Linux box you could just as easily deploy iptables in a host based fashion.

10. Multiple Choice Test Question

Below are some of the contents of a snort alert file:

```
[**] [1:372:4] ICMP PING Delphi-Piette Windows [**]
[Classification: Misc activity] [Priority: 3]
05/07-04:26:51.320853 a.b.c.5 -> x.y.z.68
ICMP TTL:63 TOS:0x0 ID:29843 IpLen:20 DgmLen:84
Type:8  Code:0  ID:1280  Seq:32512  ECHO
[Xref => http://www.whitehats.com/info/IDS155]

[**] [1:376:4] ICMP PING Microsoft Windows [**]
[Classification: Misc activity] [Priority: 3]
05/07-04:59:55.268274 a.b.c.5 -> x.y.z.68
ICMP TTL:127 TOS:0x0 ID:44728 IpLen:20 DgmLen:78
Type:8  Code:0  ID:1280  Seq:32001  ECHO
[Xref => http://www.whitehats.com/info/IDS159]

[**] [1:480:2] ICMP PING speedera [**]
[Classification: Misc activity] [Priority: 3]
05/07-05:07:14.733414 a.b.c.5 -> x.y.z.68
ICMP TTL:127 TOS:0x0 ID:64698 IpLen:20 DgmLen:124
Type:8  Code:0  ID:1280  Seq:40193  ECHO
```

Below are the respective snort filters that caused the alerts

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

50696e67696e672066726f6d2044656c = asci "Pinging from Del"
```

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING Microsoft
Windows"; content:"|30313233343536373839616263646566768696a6b6c6d6e6f70|";
itype:8; depth:32; reference:arachnids,159; sid:376;  classtype:misc-activity;
rev:4;)

30313233343536373839616263646566768696a6b6c6d6e6f70 =
   asci "0123456789abcdefghijklmnop"

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING speedera";
content: "|3839 3a3b 3c3d 3e3f|"; depth: 100; itype: 8;  sid:480;
classtype:misc-activity; rev:2;)

38393a3b3c3d3e3f = asci "89:;<=>?"
```

Question: Looking at the complexity of the "content" that each filter has, which filter would be the least likely to fire a "false positive"?

A. ICMP PING Delphi-Piette Windows
B. ICMP PING Microsoft Windows
C. ICMP PING speedera
D. They are all equally capable of a false positive
E. It is impossible for any of them to fire a false positive

Answer: A – The other two alerts fire anytime there is a set of consecutive ASCII values placed together. Delphi is random enough to make it less likely to be false.

Detect Number 2

1. Source of Trace

This trace was taken from a DSL connection. The topology is simple. A hub connects the DSL modem, the victim machine and the sniffer. The victim machine was a laptop running an un-patched server install of Red Hat 6.1.

2. Detect was generated by

This detect was generated by Snort version 1.8.6 with a standard configuration file. Below is the alert snort generated:

The Alert:

```
[**] [1:553:2] INFO FTP anonymous login attempt [**]
[Classification: Misc activity] [Priority: 3]
05/01-12:51:34.540303 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6741 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0xCDE6EF91  Ack: 0xA9388011  Win: 0x43F5  TcpLen: 20
```

The Snort Rule that fired:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"INFO FTP anonymous FTP";
content:"anonymous"; nocase; flags:A+; classtype:not-suspicious; sid:553;
rev:1;)
```

The Packet the caused the Rule to Fire:

```
05/01-12:51:34.540303 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6741 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0xCDE6EF91  Ack: 0xA9388011  Win: 0x43F5  TcpLen: 20
55 53 45 52 20 61 6E 6F 6E 79 6D 6F 75 73 0D 0A  USER anonymous..
```

3. Probability the Source Address was Spoofed

The probability that this address is spoofed is very low. This alert was generated by an FTP connection. Once the conversation has progressed to the point where "anonymous" is presented as the user ID, the three-way-handshake has already occurred. It is difficult to spoof a TCP conversation. Subtracting the TTL from 128 (which is the initial TTL on a windows PC) and then comparing that to the number of hops between

the victim and the attacker would further correlate this; however, there was some type of device between the victim and the attacker, that was blocking traceroute packets. The last hop I could see was not even in the 80.* address space.

4. Description of Attack

This attack is an attempt to find ftp servers allowing anonymous access. It's probable, but not definite, that the tool used in this case is Grim's Ping (http://grimsping.cjb.net). What tipped me off was that the anonymous password provided for logging into the ftp server was Wgpuser@home.com. Grim's Ping generates a list of possible public ftp servers with anonymous write access. Below is the packets that fired the snort alert.

```
05/01-12:51:34.540303 4.47.159.186:21 -> 80.11.195.55:3482
TCP TTL:64 TOS:0x10 ID:1243 IpLen:20 DgmLen:108 DF
***AP*** Seq: 0xA9388011  Ack: 0xCDE6EFA1  Win: 0x7D78  TcpLen: 20
33 33 31 20 47 75 65 73 74 20 6C 6F 67 69 6E 20  331 Guest login
6F 6B 2C 20 73 65 6E 64 20 79 6F 75 72 20 63 6F  ok, send your co
6D 70 6C 65 74 65 20 65 2D 6D 61 69 6C 20 61 64  mplete e-mail ad
64 72 65 73 73 20 61 73 20 70 61 73 73 77 6F 72  dress as passwor
64 2E 0D 0A                                      d...

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

05/01-12:51:34.720284 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6746 IpLen:20 DgmLen:63 DF
***AP*** Seq: 0xCDE6EFA1  Ack: 0xA9388055  Win: 0x43B1  TcpLen: 20
50 41 53 53 20 57 67 70 75 73 65 72 40 68 6F 6D  PASS Wgpuser@hom
65 2E 63 6F 6D 0D 0A                             e.com..

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

5. Attack Mechanism

The attack works like a user trying to log onto an ftp server. The 3-way handshake is performed and then the program attempts to log on anonymously with user "anonymous" and password Wgpuser@home.com. It then abruptly sends a FIN packet and dumps the connection.

There is significance in the "W" part of the password. A conversation on the Dshield mailing list (http://www1.dshield.org/pipermail/dshield/2001-October/001677.html) hints that it rotates that character from Q through Z. Apparently, previous versions of Grim's ping would use guest@here.com as the password, and that was too easy to block. Since I wasn't allowing guest login, it quickly went away. Below is the complete trace captured.

```
05/01-12:51:33.850373 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6718 IpLen:20 DgmLen:48 DF
******S* Seq: 0xCDE6EF90  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

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

05/01-12:51:33.850373 4.47.159.186:21 -> 80.11.195.55:3482
TCP TTL:64 TOS:0x0 ID:1236 IpLen:20 DgmLen:48 DF
***A**S* Seq: 0xA9387F95  Ack: 0xCDE6EF91  Win: 0x7D78  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

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

05/01-12:51:34.040354 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6729 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xCDE6EF91  Ack: 0xA9387F96  Win: 0x4470  TcpLen: 20

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

05/01-12:51:34.090349 4.47.159.186:1047 -> 80.11.195.55:113
TCP TTL:64 TOS:0x0 ID:1238 IpLen:20 DgmLen:60 DF
******S* Seq: 0xA91ABF1A  Ack: 0x0  Win: 0x7D78  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 13504561 0 NOP WS: 0

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

05/01-12:51:34.280329 80.11.195.55:113 -> 4.47.159.186:1047
TCP TTL:114 TOS:0x0 ID:6735 IpLen:20 DgmLen:40
***A*R** Seq: 0x0  Ack: 0xA91ABF1B  Win: 0x0  TcpLen: 20

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

05/01-12:51:34.340323 4.47.159.186:21 -> 80.11.195.55:3482
TCP TTL:64 TOS:0x10 ID:1241 IpLen:20 DgmLen:163 DF
***AP*** Seq: 0xA9387F96  Ack: 0xCDE6EF91  Win: 0x7D78  TcpLen: 20
32 32 30 20 63 68 63 67 69 6C 32 2D 61 72 38 2D  220 chcgil2-ar8-
34 2D 34 37 2D 31 35 39 2D 31 38 36 2E 63 68 63  4-47-159-186.chc
67 69 6C 32 2E 64 73 6C 2D 76 65 72 69 7A 6F 6E  gil2.dsl-verizon
2E 6E 65 74 20 46 54 50 20 73 65 72 76 65 72 20  .net FTP server
28 56 65 72 73 69 6F 6E 20 77 75 2D 32 2E 35 2E  (Version wu-2.5.
30 28 31 29 20 54 75 65 20 53 65 70 20 32 31 20  0(1) Tue Sep 21
31 36 3A 34 38 3A 31 32 20 45 44 54 20 31 39 39  16:48:12 EDT 199
39 29 20 72 65 61 64 79 2E 0D 0A                 9) ready...

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

05/01-12:51:34.540303 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6741 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0xCDE6EF91  Ack: 0xA9388011  Win: 0x43F5  TcpLen: 20
55 53 45 52 20 61 6E 6F 6E 79 6D 6F 75 73 0D 0A  USER anonymous..
```

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

05/01-12:51:34.540303 4.47.159.186:21 -> 80.11.195.55:3482
TCP TTL:64 TOS:0x10 ID:1242 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xA9388011  Ack: 0xCDE6EFA1  Win: 0x7D78  TcpLen: 20

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

05/01-12:51:34.540303 4.47.159.186:21 -> 80.11.195.55:3482
TCP TTL:64 TOS:0x10 ID:1243 IpLen:20 DgmLen:108 DF
***AP*** Seq: 0xA9388011  Ack: 0xCDE6EFA1  Win: 0x7D78  TcpLen: 20
33 33 31 20 47 75 65 73 74 20 6C 6F 67 69 6E 20  331 Guest login
6F 6B 2C 20 73 65 6E 64 20 79 6F 75 72 20 63 6F  ok, send your co
6D 70 6C 65 74 65 20 65 2D 6D 61 69 6C 20 61 64  mplete e-mail ad
64 72 65 73 73 20 61 73 20 70 61 73 73 77 6F 72  dress as passwor
64 2E 0D 0A                                      d...

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

05/01-12:51:34.720284 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6746 IpLen:20 DgmLen:63 DF
***AP*** Seq: 0xCDE6EFA1  Ack: 0xA9388055  Win: 0x43B1  TcpLen: 20
50 41 53 53 20 57 67 70 75 73 65 72 40 68 6F 6D  PASS Wgpuser@hom
65 2E 63 6F 6D 0D 0A                             e.com..

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

05/01-12:51:34.730283 4.47.159.186:21 -> 80.11.195.55:3482
TCP TTL:64 TOS:0x10 ID:1244 IpLen:20 DgmLen:73 DF
***AP*** Seq: 0xA9388055  Ack: 0xCDE6EFB8  Win: 0x7D78  TcpLen: 20
35 33 30 20 43 61 6E 27 74 20 73 65 74 20 67 75  530 Can't set gu
65 73 74 20 70 72 69 76 69 6C 65 67 65 73 2E 0D  est privileges..
0A                                               .

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

05/01-12:51:34.990257 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6761 IpLen:20 DgmLen:40 DF
***A***F Seq: 0xCDE6EFB8  Ack: 0xA9388076  Win: 0x4390  TcpLen: 20

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

05/01-12:51:34.990257 4.47.159.186:21 -> 80.11.195.55:3482
TCP TTL:64 TOS:0x10 ID:1245 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xA9388076  Ack: 0xCDE6EFB9  Win: 0x7D78  TcpLen: 20

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

05/01-12:51:34.990257 4.47.159.186:21 -> 80.11.195.55:3482
TCP TTL:64 TOS:0x10 ID:1246 IpLen:20 DgmLen:77 DF
***AP*** Seq: 0xA9388076  Ack: 0xCDE6EFB9  Win: 0x7D78  TcpLen: 20
32 32 31 20 59 6F 75 20 63 6F 75 6C 64 20 61 74  221 You could at
20 6C 65 61 73 74 20 73 61 79 20 67 6F 6F 64 62   least say goodb
79 65 2E 0D 0A                                   ye...

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

05/01-12:51:35.000256 4.47.159.186:21 -> 80.11.195.55:3482
TCP TTL:64 TOS:0x10 ID:1247 IpLen:20 DgmLen:40 DF
***A***F Seq: 0xA938809B  Ack: 0xCDE6EFB9  Win: 0x7D78  TcpLen: 20

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

```
05/01-12:51:35.230232 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6793 IpLen:20 DgmLen:40 DF
*****R** Seq: 0xCDE6EFB9  Ack: 0x0  Win: 0x0  TcpLen: 20

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

05/01-12:51:35.240231 80.11.195.55:3482 -> 4.47.159.186:21
TCP TTL:114 TOS:0x0 ID:6795 IpLen:20 DgmLen:40
*****R** Seq: 0xCDE6EFB9  Ack: 0xCDE6EFB9  Win: 0x0  TcpLen: 20

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

6. Correlations

A quick search for either Wgpuser@home.com or "grim's ping" on google returns several links of interest. This link on Security Incidents Reporting mailing lists (http://lists.insecure.org/incidents/2002/Jul/0017.html) starts a thread where someone is asking about the scan.

7. Evidence of Active Targeting

There is not much evidence of active targeting. There is only one attempt to connect to the ftp server and no other traffic from this host in the same 24 hour period. Since snort was focused on the server's IP address only, it's hard to verify that other addresses were attempted, but this seems to be the case.

8. Severity

Criticality = 1. This server is of no real value to me.

Lethality = 3. Even though this server is not valuable to me, it could be a launching pad for attacks against other people. This particular attack is not to root the box, but to find storage for others; therefore I give it a 3.

System Countermeasures = 1. This was an un-patched load of Red Hat Linux 6.1. There were no firewall rules and no tcpd (TCP wrappers) protection on the box.

Network Countermeasures = 1. Again, this is a home "class" DSL connection with no firewall of any type in front of it.

Severity = (1 + 3) – (1 + 1) = 4 – 2 = 2.

9. Defensive Recommendation

I should reexamine the need for anonymous ftp on this server. If it is not needed, turn it off. If it is needed, then I should look into "chrooting" the ftp service. If configurable by the server, deny access to anyone using that password for access. Using the iptables --string functionality would help block this. Perhaps TCP wrappers would help me keep the bad guys at bay as well.

10. Test Question

```
12:51:33.850373 a.b.c.23.3482 > x.y.z.100.21: S 3454463888:3454463888(0) win
16384 <mss
1460,nop,nop,sackOK> (DF)

12:51:33.850373 x.y.z.100.21 > a.b.c.23.3482: S 2839052181:2839052181(0) ack
3454463889
win 32120 <mss 1460,nop,nop,sackOK> (DF)

12:51:34.040354 a.b.c.23.3482 > x.y.z.100.21: . ack 1 win 17520 (DF)

12:51:34.090349 x.y.z.100.1047 > a.b.c.23.113: S 2837102362:2837102362(0) win
32120 <mss
 1460,sackOK,timestamp 13504561 0,nop,wscale 0> (DF)

12:51:34.280329 a.b.c.23.113 > x.y.z.100.1047: R 0:0(0) ack 2837102363 win 0

12:51:34.340323 x.y.z.100.21 > a.b.c.23.3482: P 1:124(123) ack 1 win 32120
(DF) [tos 0x10]
```

In reference to the tcpdump above, what is the purpose of the non tcp port 21 traffic in the middle of the dump?

A. This is the "data" connection of a passive ftp session
B. That traffic is unrelated to the ftp communication
C. The server is attempting to get identification from the client
D. None of the above

Answer: C This is a call back to the "auth" port on the client. Many servers do this as an attempt to have more detailed logging.

Detect Number 3

1. Source of Trace

This trace was taken from a DSL connection with a simple topology. A hub connects the victim machine and the sniffer to a DSL modem. The victim machine was a laptop running an un-patched server install of Red Hat 6.1.

2. Detect was generated by

This detect was generated by Snort version 1.8.6 with a standard configuration file. Below is the alert snort generated.

The Alerts:

```
[**] [1:1256:3] WEB-IIS CodeRed v2 root.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
05/15-17:45:14.368070 4.47.146.28:2397 -> 4.47.159.186:80
TCP TTL:127 TOS:0x0 ID:58707 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x52BCC967  Ack: 0x614BEC6D  Win: 0x4470  TcpLen: 20
[Xref => http://www.cert.org/advisories/CA-2001-19.html]

[**] [1:1002:2] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
05/15-17:45:14.678038 4.47.146.28:2418 -> 4.47.159.186:80
TCP TTL:127 TOS:0x0 ID:58752 IpLen:20 DgmLen:120 DF
***AP*** Seq: 0x52CDB5B4  Ack: 0x618C3130  Win: 0x4470  TcpLen: 20
```

The Snort Rules that fired:

```
:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS CodeRed v2
root.exe access"; flags: A+; uricontent:"scripts/root.exe?"; nocase;
classtype:web-application-attack; sid: 1256; rev:2;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS cmd.exe access";
flags: A+; content:"cmd.exe"; nocase; classtype:web-application-attack;
sid:1002; rev:2;)
```

The Packets that caused the Rules to Fire:

```
05/15-17:45:14.368070 4.47.146.28:2397 -> 4.47.159.186:80
TCP TTL:127 TOS:0x0 ID:58707 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x52BCC967  Ack: 0x614BEC6D  Win: 0x4470  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 72 6F 6F  GET /scripts/roo
74 2E 65 78 65 3F 2F 63 2B 64 69 72 20 48 54 54  t.exe?/c+dir HTT
50 2F 31 2E 30 0D 0A 48 6F 73 74 3A 20 77 77 77  P/1.0..Host: www
0D 0A 43 6F 6E 6E 6E 65 63 74 69 6F 6E 3A 20 63  ..Connnection: c
6C 6F 73 65 0D 0A 0D 0A                          lose....

05/15-17:45:14.678038 4.47.146.28:2418 -> 4.47.159.186:80
TCP TTL:127 TOS:0x0 ID:58752 IpLen:20 DgmLen:120 DF
```

```
lines 1-22 Snort received signal 3, exiting
***AP*** Seq: 0x52CDB5B4  Ack: 0x618C3130  Win: 0x4470  TcpLen: 20
47 45 54 20 2F 63 2F 77 69 6E 6E 74 2F 73 79 73  GET /c/winnt/sys
74 65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63  tem32/cmd.exe?/c
2B 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A 48  +dir HTTP/1.0..H
6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E 65  ost: www..Connne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 0D 0A  ction: close....
```

3. Probability the Source Address was Spoofed

The probability of this traffic being spoofed is very low. These alerts were generated by an http (TCP port 80) connection. At the point where the snort rules fired, the three-way-handshake has already occurred. It is difficult to spoof a TCP conversation. The TTL's are 127, so this packet hasn't traveled very far. This further supports that the source machine is in the same network as the destination.

4. Description of attack

Code Red II works by compromising a Windows 2000 server running IIS. It can propagate to an NT server, but it will crash when it tries to execute. It has three main functions: Infect the local machine, propagate to other machines, and trojan the local drive on the machine. Details of how this works can be found at http://www.eeye.com/html/Research/Advisories/AL20010804.html. If I understand the eEye document correctly, the 1st alert is Code Red II attempting to propagate itself and the 2nd alert could be Nimda attempting to propagate. I say could be, because all we know is that something was attempting to connect to /c/winnt/system32/cmd.exe. I know that Code Red II trojans a server setting up shares on the system drive and that Nimda tries this share (among other things). So, it is either Nimda, or a Nimda variant trying to take advantage of a Code Red II victim.

Snort reported hundreds of these alerts every hour. They were all the same, so I didn't bother cluttering up the paper with them.

5. Attack Mechanism

Code Red II infects a machine just like the original Code Red using the .ida vulnerability. It has some checks built in to keep it from trying to infect a machine more than once. If this is the first infection attempt, it sets up the checks to prevent a future infection. It then spawns a thread to repeat those steps again. Those threads will jump to the propagation phase because they're not the first run. Finally, it jumps to the trojan stage.

The propagation stage is rather clever. It's smart enough to not reconnect to the machine it's running on. It will stay within the same class A range of IP addresses 1/2 of the time and the same class B range 3/8th of the time. It also has checks built in to keep it from propagating after October of 2001.

The final stage is to trojan the box. The code to keep it from attacking a server more than once and to stop trying to propagate after 10/2001 makes you think it has a finite lifespan. Technically, that may be true, but it before it dies Code Red II makes sure he leaves a few back doors first. This phase puts a command shell (cmd.exe) in the msadc and scripts web directories puts a trojan-ed explore.exe in the root directory of the system drive and creates virtual web paths for c:\ and d:\.

Nimda picks up where Code Red II leaves off. It takes advantage of all the Trojans that Code Red II establishes and propagates via email, network shares, vulnerable web browsers, and of course vulnerable web servers. Below is a list of possible crafted packets from http://www.securityspace.com/smysecure/w32_nmda_amm.html.

"GET /scripts/root.exe?/c+dir HTTP/1.0" 302 209 - - - "-" "-"
"GET /MSADC/root.exe?/c+dir HTTP/1.0" 302 209 - - - "-" "-"
"GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 302 209 - - - "-" "-"
"GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 302 209 - - - "-" "-"
"GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 302 209 - - - "-" "-"
"GET /_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 302 209 - - - "-" "-"
"GET /_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+d

ir HTTP/1.0" 302 209 - - - "-" "-"
"GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0"
302 209 - - - "-" "-"
"GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0"
302 209 - - - "-" "-"
"GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0"
302 209 - - - "-" "-"

The email propagation takes advantage of the "Automated Execution of Embedded MIME Types" vulnerability detailed here http://www.cert.org/advisories/CA-2001-06.html. Nimda propagates via the browser by modifying all files containing web content. If an unsuspecting person browses content on an infected web server it will download the worm to it's cache. Some browsers even execute the worm automatically. Nimda travels through network shares by placing copies of itself in all writable directories on a file system.

Here are some packets of interest from the snort trace.

```
05/15-17:45:14.368070 4.47.146.28:2397 -> 4.47.159.186:80
TCP TTL:127 TOS:0x0 ID:58707 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x52BCC967  Ack: 0x614BEC6D  Win: 0x4470  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 72 6F 6F  GET /scripts/roo
74 2E 65 78 65 3F 2F 63 2B 64 69 72 20 48 54 54  t.exe?/c+dir HTT
50 2F 31 2E 30 0D 0A 48 6F 73 74 3A 20 77 77 77  P/1.0..Host: www
0D 0A 43 6F 6E 6E 6E 65 63 74 69 6F 6E 3A 20 63  ..Connnection: c
6C 6F 73 65 0D 0A 0D 0A                          lose....

05/15-17:45:14.538053 4.47.146.28:2410 -> 4.47.159.186:80
TCP TTL:127 TOS:0x0 ID:58731 IpLen:20 DgmLen:110 DF
***AP*** Seq: 0x52C79506  Ack: 0x6181F350  Win: 0x4470  TcpLen: 20
47 45 54 20 2F 4D 53 41 44 43 2F 72 6F 6F 74 2E  GET /MSADC/root.
65 78 65 3F 2F 63 2B 64 69 72 20 48 54 54 50 2F  exe?/c+dir HTTP/
31 2E 30 0D 0A 48 6F 73 74 3A 20 77 77 77 0D 0A  1.0..Host: www..
43 6F 6E 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F  Connnection: clo
73 65 0D 0A 0D 0A                                se....

05/15-17:45:14.678038 4.47.146.28:2418 -> 4.47.159.186:80
TCP TTL:127 TOS:0x0 ID:58752 IpLen:20 DgmLen:120 DF
***AP*** Seq: 0x52CDB5B4  Ack: 0x618C3130  Win: 0x4470  TcpLen: 20
47 45 54 20 2F 63 2F 77 69 6E 6E 74 2F 73 79 73  GET /c/winnt/sys
74 65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63  tem32/cmd.exe?/c
2B 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A 48  +dir HTTP/1.0..H
6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E 65  ost: www..Connne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 0D 0A  ction: close....
```

6. Correlations

Here's the CERT references for these attacks:

http://www.cert.org/advisories/CA-2001-26.html
http://www.cert.org/incident_notes/IN-2001-09.html

Both links provide some detail on the attacks as well as links to other references. It is easy to list correlations for these attacks, but hard to find any more meaningful than the rest. Along with the original Code Red, I'd argue that there is more internet content about these attacks than anything else.

7. Evidence of Active Targeting

I'm not sure if this type of traffic is considered active targeting or not. Some people argue that propagation intelligence of Code Red II is active targeting; however, I do not personally agree. It's possible that the owner of the attacking computer isn't aware that the machine is attacking other computers, so it's hard for me to call it active targeting. Additionally, the nature of this worm is to attack multiple hosts.

8. Severity

Criticality = 1. This server is of no real value to me.

Lethality = 3. These attacks are extremely lethal; however, they are only successful to Microsoft IIS servers and a couple of Cisco products. This server is running the Apache web server on the Linux OS.

System Countermeasures = 3. This was an un-patched load of Red Hat Linux 6.1. There were no firewall rules and no tcpd (TCP wrappers) protection on the box; however, it is not vulnerable to these attacks.

Network Countermeasures = 1. Again, this is a home "class" DSL connection with no firewall of any type in front of it.

Severity = (1 + 3) – (3 + 1) = 4 – 4 = 0. The severity is very low, but I can't say it's 0. Realistically it's a very low 1.

9. Defensive Recommendations

The defenses are adequate to fend off this attack; however, it is a good opportunity to further secure the box. Is Apache needed? If not, disable it. If the computer is running samba, it's possible to passively spread Nimda, so we should make sure we're not running a samba server unless needed. It's always a good idea to put some type of firewall on or in front of a computer on the internet. With Linux it's free and easy to do so with iptables. TCP wrappers is another point of control too.

10. Multiple Choice Test Question

See the section of a trace dump below:

```
05/15-17:45:14.818024 a.b.c.47:2425 -> x.y.z.100:80
TCP TTL:127 TOS:0x0 ID:58769 IpLen:20 DgmLen:120 DF
***AP*** Seq: 0x52D3B946  Ack: 0x61712845  Win: 0x4470  TcpLen: 20
47 45 54 20 2F 64 2F 77 69 6E 6E 74 2F 73 79 73  GET /d/winnt/sys
74 65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63  tem32/cmd.exe?/c
2B 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A 48  +dir HTTP/1.0..H
6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E 65  ost: www..Connne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 0D 0A  ction: close....

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

05/15-17:45:14.818024 x.y.z.100:80 -> a.b.c.47:2425
TCP TTL:64 TOS:0x0 ID:34 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x61712845  Ack: 0x52D3B996  Win: 0x7D78  TcpLen: 20

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

05/15-17:45:14.818024 x.y.z.100:80 -> a.b.c.47:2425
TCP TTL:64 TOS:0x0 ID:35 IpLen:20 DgmLen:411 DF
***AP*** Seq: 0x61712845  Ack: 0x52D3B996  Win: 0x7D78  TcpLen: 20
48 54 54 50 2F 31 2E 31 20 34 30 34 20 4E 6F 74  HTTP/1.1 404 Not
20 46 6F 75 6E 64 0D 0A 44 61 74 65 3A 20 54 68   Found..Date: Th
75 2C 20 31 36 20 4D 61 79 20 32 30 30 32 20 30  u, 16 May 2002 0
33 3A 34 35 3A 32 34 20 47 4D 54 0D 0A 53 65 72  3:45:24 GMT..Ser
76 65 72 3A 20 41 70 61 63 68 65 2F 31 2E 33 2E  ver: Apache/1.3.
39 20 28 55 6E 69 78 29 20 20 28 52 65 64 20 48  9 (Unix)  (Red H
61 74 2F 4C 69 6E 75 78 29 0D 0A 43 6F 6E 6E 65  at/Linux)..Conne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 43 6F  ction: close..Co
6E 74 65 6E 74 2D 54 79 70 65 3A 20 74 65 78 74  ntent-Type: text
2F 68 74 6D 6C 0D 0A 0D 0A 3C 21 44 4F 43 54 59  /html....<!DOCTY
50 45 20 48 54 4D 4C 20 50 55 42 4C 49 43 20 22  PE HTML PUBLIC "
2D 2F 2F 49 45 54 46 2F 2F 44 54 44 20 48 54 4D  -//IETF//DTD HTM
4C 20 32 2E 30 2F 2F 45 4E 22 3E 0A 3C 48 54 4D  L 2.0//EN">.<HTM
4C 3E 3C 48 45 41 44 3E 0A 3C 54 49 54 4C 45 3E  L><HEAD>.<TITLE>
34 30 34 20 4E 6F 74 20 46 6F 75 6E 64 3C 2F 54  404 Not Found</T
```

```
49 54 4C 45 3E 0A 3C 2F 48 45 41 44 3E 3C 42 4F  ITLE>.</HEAD><BO
44 59 3E 0A 3C 48 31 3E 4E 6F 74 20 46 6F 75 6E  DY>.<H1>Not Foun
64 3C 2F 48 31 3E 0A 54 68 65 20 72 65 71 75 65  d</H1>.The reque
73 74 65 64 20 55 52 4C 20 2F 64 2F 77 69 6E 6E  sted URL /d/winn
74 2F 73 79 73 74 65 6D 33 32 2F 63 6D 64 2E 65  t/system32/cmd.e
78 65 20 77 61 73 20 6E 6F 74 20 66 6F 75 6E 64  xe was not found
20 6F 6E 20 74 68 69 73 20 73 65 72 76 65 72 2E   on this server.
3C 50 3E 0A 3C 2F 42 4F 44 59 3E 3C 2F 48 54 4D  <P>.</BODY></HTM
4C 3E 0A                                         L>.
```

The packets above are of a Nimda attack against a web server. The internet is flooded with this type of traffic because of its effective self-propagating properties and that most of its victims are unaware internet users with a Windows operating system on their computer. As a result, it is becoming somewhat common to see security administrators disable detection of this traffic so as not to needlessly fill up log files with attack attempts they are no longer vulnerable too.

Look again at the web servers response. How might a savvy cracker be able to use this to his advantage for stealth reconnaissance?

A. A cracker could use this to scan for web servers
B. A cracker could determine if the machine was vulnerable to RPC attacks
C. A cracker could use this to attempt to determine OS type and web server information
D. All of the above.

Answer: C. While choice A is true, this is not the best answer. Making the cmd.exe request would be a ton of overhead just to find a web server.

Part 3 – Analyze This

Intrusion Detection In Depth

GCIA Practical Assignment
Cory Steers
Version 3.1

Executive Summary

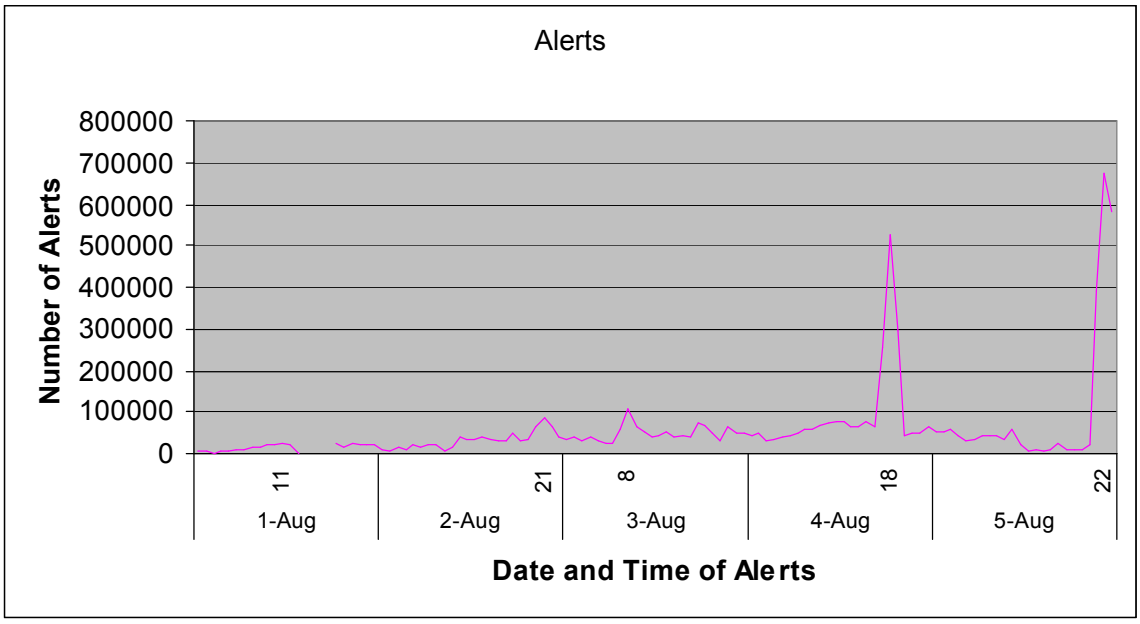

Figure 3.1

This is a security audit for the University of SANS GIAC (USG). USG has provided data from the Snort Intrusion Detection System (IDS) using a "fairly standard rulebase". In summary, USG is looking for an analysis of the state of its network in general, as well as any specific attacks or compromises, both possible and confirmed.

The chart above represents the number of alerts generated per hour for the 5 days of logs analyzed. The number above the date (along the X-axis) is the hour that generated the highest number of alerts for that day.

As you look at Figure 3.1 you may start to panic because it appears that even in the slow hours you're averaging more than 50,000 alerts per hour. First, the graph is a combination of alerts and scans. While scans can be an indicator of potential future attacks, they are by themselves mostly harmless. Likewise, as the details below will show, the majority of the alerts being generated are simply noise. This noise is masking the real dangerous activity and steps should be taken to reduce this noise as much as possible.

This is not to say that there have been no threats to the universities network. Only that it's difficult to see them … like a needle in a haystack. There is evidence to suggest that the university's computers, probably mostly student PCs, have been compromised. For details on actions to take, please see the Defensive Recommendations section below.

List of Files Analyzed

alert.020801.gz
alert.020802.gz
alert.020803.gz
alert.020804.gz
alert.020805.gz

oos_Aug.1.2002.gz
oos_Aug.2.2002.gz
oos_Aug.3.2002.gz
oos_Aug.4.2002.gz
oos_Aug.5.2002.gz

scans.020801.gz
scans.020802.gz
scans.020803.gz
scans.020804.gz
scans.020805.gz

Analysis Process Used

The following steps were used to analyze the information for each day. First, I wrote a PERL script (see appendix A) to process through the alert and scans files to get a list of "top talkers". That same script reported the top 10 alerts as well. Second, using the top alerts output, I sifted through the signatures for interesting and potentially serious alerts that didn't originate from the university. I then listed the top 10 talkers, per day, for the five days analyzed. Next, I picked out the

interesting external source addresses and listed their registration and DNS information. Finally, I compiled a list of defensive recommendations and wrote the executive summary. I used the PERL script listed in Appendix B to help retrieve the data for my graph in the executive summary.

Analysis of Detects and Correlations

NIMDA – Attempt to execute cmd from campus host
NIMDA – Attempt to execute root from campus host

```
08/05-09:14:50.113555  [**] NIMDA - Attempt to execute cmd from campus host
[**] 130.85.70.169:
1103 -> 65.54.250.121:80
08/05-13:22:36.967751  [**] NIMDA - Attempt to execute cmd from campus host
[**] 130.85.70.144:
1116 -> 207.46.235.150:80
08/05-21:21:55.661920  [**] NIMDA - Attempt to execute root from campus host
[**] 130.85.100.20
8:2008 -> 130.95.40.191:80
08/05-21:21:55.664339  [**] NIMDA - Attempt to execute root from campus host
[**] 130.85.100.20
8:2010 -> 130.7.64.55:80
```

These two alerts represent the 1st and 4th largest alerts for the days analyzed. In total, there were 1,360,018 alerts generated. What peaked my interest was that all but a handful of alerts were generated from 130.85.100.208, and all but about 10 alerts were generated on August 5th. I wanted to see if I could see what happened to this host.

On August 1st 130.85.100.208 received 13 RPC calls from various hosts to the portmapper port from two different sources:

```
08/01-07:46:32.754853  [**] External RPC call [**] 203.239.155.2:50865 ->
130.85.100.203:111
08/01-19:37:32.875003  [**] External RPC call [**] 202.108.109.100:654 ->
130.85.100.201:111
```

On August 4th, 11 SMB Name Wildcard alerts:

```
08/04-03:11:52.208588  [**] SMB Name Wildcard [**] 216.228.171.81:137 ->
130.85.100.201:137
```

They were all from the same address, but a different address than the 8/1 alerts. On August 5th, the NIMDA alerts didn't start up until about 9:21 PM. Prior to that, it did have a lot of other activity:

```
08/05-15:09:28.531870  [**] connect to 515 from outside [**]
216.241.23.211:3362 -> 130.85.100.204:515
…
08/05-15:47:14.743794  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1105 -> 192.168.0.1:69
…
08/05-15:49:03.249785  [**] spp_http_decode: IIS Unicode attack detected [**]
61.189.144.7:4935 -> 130.85.100.208:80
…
08/05-17:41:08.056810  [**] spp_http_decode: IIS Unicode attack detected [**]
66.30.130.39:3835 -> 130.85.100.208:80
08/05-17:41:08.056810  [**] spp_http_decode: IIS Unicode attack detected [**]
66.30.130.39:3835 -> 130.85.100.208:80
08/05-17:41:08.056810  [**] spp_http_decode: IIS Unicode attack detected [**]
66.30.130.39:3835 -> 130.85.100.208:80
08/05-17:41:08.206833  [**] spp_http_decode: IIS Unicode attack detected [**]
66.30.130.39:3839 -> 130.85.100.208:80
08/05-17:41:08.407605  [**] spp_http_decode: IIS Unicode attack detected [**]
66.30.130.39:3843 -> 130.85.100.208:80
…
```

Here's where it got interesting …

```
08/05-18:05:49.962351  [**] spp_http_decode: IIS Unicode attack detected [**]
65.162.184.6:50635 -> 130.85.100.208:80
08/05-18:05:50.875105  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1434 -> 169.254.97.57:69
08/05-18:05:56.094145  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1435 -> 169.254.97.57:69
08/05-18:05:57.852392  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1436 -> 169.254.97.57:69
08/05-18:05:57.853073  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1436 -> 169.254.97.57:69
…
08/05-18:08:06.085211  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1447 -> 169.254.97.57:69
08/05-18:08:06.469058  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1447 -> 169.254.97.57:69
…
08/05-20:46:04.774542  [**] spp_http_decode: IIS Unicode attack detected [**]
61.145.69.74:4807 -> 130.85.100.208:80
08/05-20:46:05.012671  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1913 -> 61.145.69.74:69
08/05-20:41:28.807450  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1906 -> 61.145.69.74:69
08/05-20:47:34.993803  [**] spp_http_decode: IIS Unicode attack detected [**]
61.145.69.74:3272 -> 130.85.100.208:80
08/05-20:47:36.043409  [**] spp_http_decode: IIS Unicode attack detected [**]
61.145.69.74:3299 -> 130.85.100.208:80
08/05-20:47:36.135686  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1914 -> 61.145.69.74:69
08/05-20:42:57.616171  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1907 -> 61.145.69.74:69
08/05-20:42:57.626536  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1907 -> 61.145.69.74:69
08/05-20:42:58.338763  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1908 -> 61.145.69.74:69
```

```
08/05-20:49:06.258802  [**] spp_http_decode: IIS Unicode attack detected [**]
61.145.69.74:3912 -> 130.85.100.208:80
…
08/05-20:55:14.606498  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1923 -> 61.145.69.74:69
08/05-21:20:59.260155  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1925 -> 211.141.120.18:69
…
08/05-21:21:04.380274  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1928 -> 211.141.120.18:69
08/05-21:21:06.650638  [**] spp_http_decode: IIS Unicode attack detected [**]
211.141.120.18:4853 -> 130.85.100.208:80
…
08/05-21:21:15.808352  [**] spp_http_decode: IIS Unicode attack detected [**]
211.141.120.18:3249 -> 130.85.100.208:80
08/05-21:21:16.092822  [**] TFTP - Internal UDP connection to external tftp
server [**] 130.85.100.208:1931 -> 211.141.120.18:69
08/05-21:21:55.661920  [**] NIMDA - Attempt to execute root from campus host
[**] 130.85.100.208:2008 -> 130.95.40.191:80
08/05-21:21:55.664339  [**] NIMDA - Attempt to execute root from campus host
[**] 130.85.100.208:2010 -> 130.7.64.55:80
```

It appears as though the traffic to 211.141.120.18 was the last traffic of this type before NIMDA kicks in. Since Nimda kicks in immediately, I would think this address was where it came from. So, it's not clear to me what all the other traffic was. Perhaps, 130.85.100.201 was compromised in some other way and just as it was beginning to be used for that purpose, one of the attackers accidentally infected it with NIMDA from his machine. There was some information in the scans and oos files for this host all week, but not much. None of it showed more insight into what happened here. DNS lookups on the attackers were inconclusive as well. They were either not found, or they were some type of dial-up or other dynamic IP internet accounts (DSL or cable modem).

The other curious thing are the alerts involving a 192.168.0 address. I wonder if this address was owned by some type of NAT-ing firewall device, instead of an actual PC or server. That device could have been configured to allow some services through so that the victim was still vulnerable. If this was the case and it was configured incorrectly, it could have been spewing the private addresses across the internet. Otherwise, it was some other device configured wrong and had nothing to do with the compromise.

Here is a list of CERT alerts for the various alerts generated for 130.85.100.208:

http://www.cert.org/advisories/CA-2001-26.html - for the NIMDA alert.

This was discussed heavily in the previous section.
http://www.kb.cert.org/vuls/id/111677 - is a Note about directory traversals via Unicode.
http://www.cert.org/advisories/CA-1991-18.html - a quick note about tftp. I don't believe tftp was the initial compromise, rather, I believe it was a means to get more trojan code to the box.

UDP SRC and DST outside network.

I decided to look at this alert for a couple of key reasons. There were a lot of alerts of this type recorded, and 63.250.213.12, which was one of the highest top talkers was the source address for many of these alerts.

```
08/01-07:15:03.298135  [**] UDP SRC and DST outside network [**]
63.250.213.12:1031 -> 233.28.65.148:5779
08/01-07:15:03.299081  [**] UDP SRC and DST outside network [**]
63.250.213.12:1031 -> 233.28.65.148:5779
08/01-07:15:04.473549  [**] UDP SRC and DST outside network [**]
63.250.213.12:1031 -> 233.28.65.148:5779
08/01-07:15:05.181023  [**] UDP SRC and DST outside network [**]
63.250.213.12:1031 -> 233.28.65.148:5779
08/01-07:15:06.072921  [**] UDP SRC and DST outside network [**]
63.250.213.12:1031 -> 233.28.65.148:5779
```

I thought this alert was important, but not because it represents a compromise or an attack against the university. My first thought was that this looked like someone was using university resources, or at least the network to initiate Denial of Service (DOS) or Distributed DOS (DDOS) attacks against the address 63.250.213.12. Further investigation revealed that ARIN reported the Orgname as Yahoo Broadcast Services, and a reverse DNS lookup came up with a DNS name of dal-qcwm213012.broadcast.com. Combine that with the 233 address mentioned above, and this starts looking like multicast traffic.

I assume that a university would not consider this bad traffic at all. It could be blocked, but multicast is much easier on the network than 15 guys using a unicast P2P application to get the same mpeg ripped movie.

So, either these alerts are false positives or not all the alerts are related to multicast traffic. Further research turned up these alerts:

```
08/02-11:01:16.843590  [**] UDP SRC and DST outside network [**]
169.254.145.84:123 -> 207.46.226.34:123
08/02-13:20:37.156088  [**] UDP SRC and DST outside network [**]
192.168.1.22:123 -> 129.6.15.28:123
…
08/04-16:45:01.141117  [**] UDP SRC and DST outside network [**]
192.168.1.101:1058 -> 68.34.76.5:53
08/04-16:38:02.538535  [**] UDP SRC and DST outside network [**]
192.168.1.101:1047 -> 68.34.76.5:53
…
08/05-15:05:46.134384  [**] UDP SRC and DST outside network [**]
192.168.1.22:123 -> 129.6.15.28:123
```

The above alerts are a subset to conserve space; however, most of the alerts appear to be multicast. Marc Kneppers discusses this traffic as multicast in his practical http://www.giac.org/practical/Marc_Kneppers_GCIA.zip. So does Lorraine Weaver here http://www.giac.org/practical/Lorraine_Weaver_GCIA.zip.

spp_http_decode: IIS Unicode Attack Detected
spp_http_decode: CGI Null Byte Attack detected

```
08/01-00:10:57.452228  [**] spp_http_decode: IIS Unicode attack detected [**]
64.86.155.118:2672 -> 130.85.109.87:80
08/01-00:10:59.910726  [**] spp_http_decode: IIS Unicode attack detected [**]
64.86.155.118:2710 -> 130.85.109.87:80
08/01-00:21:07.087017  [**] spp_http_decode: IIS Unicode attack detected [**]
211.91.255.154:51337 -> 130.85.53.84:80
08/01-00:21:08.346841  [**] spp_http_decode: IIS Unicode attack detected [**]
211.91.255.154:51343 -> 130.85.53.84:80

08/01-00:38:06.298032  [**] spp_http_decode: CGI Null Byte attack detected
[**] 66.32.232.141:4065 -> 130.85.70.198:80
08/01-00:38:06.479835  [**] spp_http_decode: CGI Null Byte attack detected
[**] 66.32.232.141:4068 -> 130.85.70.198:80
```

These alerts are generated by the spp_http_decode preprocessor in snort. The IIS Unicode attack means an IIS directory traversal was attempted. It is possible for this to be a false alarm if the customer is from a foreign country that uses a language that requires multi-byte characters. The CGI Null Byte attack means a %00 was found in the http request. This is also subject to false positives when cookies and SSL encryption are involved.

The extreme majority of this traffic is originating from the university's network. Some of the top talkers of this are:

130.85.85.74

```
08/02-09:06:04.218426  [**] Possible trojan server activity [**]
80.62.155.240:2177 -> 130.85.85.74:27374
08/02-09:06:04.218799  [**] Possible trojan server activity [**]
130.85.85.74:27374 -> 80.62.155.240:2177
08/02-09:06:04.819997  [**] Possible trojan server activity [**]
80.62.155.240:2177 -> 130.85.85.74:27374
08/02-09:06:04.820491  [**] Possible trojan server activity [**]
130.85.85.74:27374 -> 80.62.155.240:2177
```

Port 27374 is heavily used by SubSeven v2.1. Here's a CERT reference of W32/Leaves which uses machines with the SubSeven backdoor: http://www.cert.org/incident_notes/IN-2001-07.html. McAfee has a profile article about SubSeven too: http://vil.mcafee.com/dispVirus.asp?virus_k=10566. So, it's possible that this machine is a drone for someone else which explains the http_decode traffic. There were a few entries in the scans files, but they didn't appear to give any additional insight. This address was not listed in the oos files at all.

130.85.153.145

```
08/03-17:31:50.011035  [**] External RPC call [**] 194.98.189.139:3341 ->
130.85.153.145:111
```

This alert means the portmapper port was contacted also. The 194 address was probably attempting to find out what RPC services were available on the server. This took place after the http_decode alerts, so it's not necessarily related. Then again, this same type of traffic could have occurred prior to the logs I was given to process.

130.85.70.48

```
08/01-11:41:25.799792  [**] SMB Name Wildcard [**] 67.38.174.167:137 ->
130.85.70.48:137
```

I wasn't able to find much for this address other than the http decode alerts. This one SMB alert was the only other alert found. It was the recipient of a couple of SYN scans, but there was nothing in the oos files either. Perhaps this box wasn't compromised, and the owner is actively trying to compromise other hosts? This address is listed as ecs020pc-carole.ucs.umbc.edu. in DNS, and looks like it could be a student's PC. I talk more about this SMB alert below.

130.85.91.103

```
08/01-06:25:40.396290  [**] Samba client access [**] 64.81.195.164:39736 ->
130.85.91.103:139

08/05-14:00:07.715736  [**] Incomplete Packet Fragments Discarded [**]
211.234.105.28:0 -> 130.85.91.103:0
08/05-14:00:08.828707  [**] Incomplete Packet Fragments Discarded [**]
211.234.105.28:0 -> 130.85.91.103:0
```

The "Samba client access" alert is telling us that a windows SMB share was connected to, and we probably shouldn't allow this across the internet. DNS reports this belonging to the dsl.speakeasy.net domain. It's quite likely that this external attacker has compromised this machine. He's in the 8/1 alert file a lot for "Exploit x86 NOOP" and a couple of "SMB CD…" alerts, which are all to the tcp 139 port. My snort rules files appear to be slightly different than the universities. The closest I could come to snort rules were:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"EXPLOIT x86 linux s
amba overflow";flags: A+; content:"|eb2f 5feb 4a5e 89fb 893e 89f2|";
reference:bugtraq,1816; reference:cve,CVE-1999-0811; reference:cve,CVE-1999-
0182; classtype:attempted-admin; sid:292; rev:2;)

:alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB CD..";flags:
A+; content:"\\..|2f 00 00 00|"; reference:arachnids,338;
classtype:attempted-recon; sid:534; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB CD...";flags:
A+; content:"\\...|00 00 00|"; reference:arachnids,337; classtype:attempted-
recon; sid:535; rev:1;)
```

The Incomplete Packet Fragments message is an alert from the spp_defrag preprocessor. It's probably UDP packets firing the alerts, but I can't be sure of that based on the information I have. Typically, that's what it is, but it could be a false positive. This article http://www.mcabee.org/lists/snort-users/Nov-01/msg00820.html suggests that the spp_defrag preprocessor should be replaced with frag2, due to problems.

As you can see, there is additional evidence here to suggest that the http_decode attacks are legitimate attacks, and could be the root cause of a lot of other attacks as well. Here's a link that begins a discussion about the CGI Null Byte attack http://archives.neohapsis.com/archives/snort/2000-11/0241.html. Gary Smith talks about these attacks in his practical here http://www.giac.org/practical/Gary_Smith_GCIA.zip.

beetle.ucs

```
08/01-01:11:03.622809  [**] beetle.ucs [**] 65.211.134.134:2189 ->
130.85.70.69:139
08/01-01:11:03.622936  [**] beetle.ucs [**] 130.85.70.69:139 ->
65.211.134.134:2189
08/01-01:11:04.676576  [**] beetle.ucs [**] 130.85.70.69:139 ->
65.211.134.134:218
```

This peeked my curiosity because I had no idea what this was and wanted to learn more about it. After further investigation, this appears to be a custom alert written by the university. They have a server on their network with a DNS name of beetle.ucs.umbc.edu. I still don't fully understand what this is, but this university link references it with regards to burning a CD-R, and Edward Peck mentioned it in his GCIA Practical as relating to CD-R's as well.

The top 3 addresses firing this alert are 80.137.90.34, 203.213.58.38, and 211.232.192.153.

80.137.90.34
DNS: p50895A22.dip.t-dialin.net
Some alerts found:

```
08/05-11:14:43.242446  [**] spp_http_decode: IIS Unicode attack detected [**]
80.137.90.34:2134 -> 130.85.157.11:80
08/05-11:45:21.624113  [**] spp_portscan: End of portscan from 80.137.90.34:
TOTAL time(25s) hosts(166) TCP(175) UDP(0) [**]
08/05-11:46:47.767171  [**] spp_portscan: PORTSCAN DETECTED from 80.137.90.34
(THRESHOLD 12 connections exceeded in 2 seconds) [**]
08/05-11:47:53.617161  [**] spp_portscan: End of portscan from 80.137.90.34:
TOTAL time(31s) hosts(693) TCP(703) UDP(0) [**]
```

The alerts were abbreviated here for space considerations. There were many IIS Unicode alerts as well as different portscan messages. This box did nothing but beetle.ucs until 8/5 and then just exploded with attacks.

203.213.58.38
DNS: syd-ts6-2600-038.tpgi.com.au – Australia?
Some Alerts found:

```
08/04-23:22:46.955510  [**] SMB Name Wildcard [**] 203.213.58.38:137 ->
130.85.56.8:137
08/05-00:53:48.036277  [**] SMB Name Wildcard [**] 203.213.58.38:137 ->
130.85.70.11:137
```

```
08/05-00:53:49.515939  [**] SMB Name Wildcard [**] 203.213.58.38:137 ->
130.85.70.11:137
08/05-00:54:14.676553  [**] SMB Name Wildcard [**] 203.213.58.38:137 ->
130.85.70.19:137
08/05-00:54:16.176122  [**] SMB Name Wildcard [**] 203.213.58.38:137 ->
130.85.70.19:137
```

The alerts were abbreviated. This machine appears to be scanning the entire university. This alert is discussed below in detail.

211.232.192.153
DNS: "not found"
Some Alerts found:

```
08/04-12:17:59.329213  [**] spp_portscan: portscan status from
211.232.192.153: 3 connections across 3 hosts: TCP(3), UDP(0) [**]
08/04-12:18:00.991190  [**] spp_portscan: End of portscan from
211.232.192.153: TOTAL time(84s) hosts(1517) TCP(1558) UDP(0) [**]
```

This machine appears to have limited his activity to portscans.

External FTP to Helpdesk

```
08/01-08:45:57.414452  [**] HelpDesk 130.85.83.197 to External FTP [**]
130.85.83.197:1058 -> 161.69.2.23:21
08/01-09:44:18.815020  [**] HelpDesk 130.85.70.50 to External FTP [**]
130.85.70.50:1191 -> 161.69.2.7:21
08/02-09:34:23.389875  [**] HelpDesk 130.85.70.49 to External FTP [**]
130.85.70.49:1445 -> 161.69.2.7:21
08/02-12:40:39.206928  [**] External FTP to HelpDesk 130.85.70.49 [**]
192.117.104.51:3521 -> 130.85.70.49:21
08/02-12:40:39.230247  [**] External FTP to HelpDesk 130.85.70.50 [**]
192.117.104.51:3522 -> 130.85.70.50:21
```

There seems to be a few custom built snort rules created by the university. There are ftp connections both to and from the helpdesk server. The three IP addresses that had the most hits for these alerts are 213.44.229.72, 80.8.77.137, and 80.11.212.202. All three of these IP addresses have a French domain name and only generated this alert and a lot of port scanning alerts.

Next, I looked for non-ftp traffic from the three helpdesk servers.

```
08/02-15:15:57.717274  [**] SMB Name Wildcard [**] 200.27.201.145:1052 ->
130.85.83.197:137
08/05-18:51:03.290194  [**] Possible trojan server activity [**]
63.196.247.234:3093 -> 130.85.83.197:27374
…
08/05-15:09:49.573043  [**] spp_http_decode: IIS Unicode attack detected [**]
```

```
130.85.70.50:2528 -> 207.200.89.193:80
08/05-15:09:49.573043  [**] spp_http_decode: IIS Unicode attack detected [**]
130.85.70.50:2528 -> 207.200.89.193:80
…
08/02-16:42:14.669405  [**] SMB Name Wildcard [**] 216.228.171.81:137 ->
130.85.70.49:137
08/02-16:42:17.657427  [**] SMB Name Wildcard [**] 216.228.171.81:137 ->
130.85.70.49:137
```

These machines need to be pulled off the network, checked out, and if need be, cleaned up. The .50 box is most certainly compromised as it's initiating traffic that is firing snort alerts. The .197 server may have SubSeven running on it.

SMB Name Wildcard

There's a good note about this here: http://archives.neohapsis.com/archives/snort/2000-01/0222.html and white hats has some information here: http://www.whitehats.com/info/IDS177. You see this mostly when an attacker only has an IP address and is trying to get the NETBIOS name for it. This is usually a pre-attack probe. So, let's see who fired this alert the most, and see if we can catch them doing something else.

The top three talkers of this type of traffic are 216.228.171.81, 63.21.4.50, and 169.254.113.62.

216.228.171.81
DNS: "not found"
Alerts Found:

```
08/02-16:42:32.267620  [**] SMB Name Wildcard [**] 216.228.171.81:137 ->
130.85.70.69:137
08/02-16:42:33.774777  [**] SMB Name Wildcard [**] 216.228.171.81:137 ->
130.85.70.69:137
08/02-16:42:30.060297  [**] beetle.ucs [**] 216.228.171.81:3091 ->
130.85.70.69:445
08/02-16:42:30.060477  [**] beetle.ucs [**] 130.85.70.69:445 ->
216.228.171.81:3091
…
08/01-08:17:49.906202  [**] spp_portscan: PORTSCAN DETECTED from
216.228.171.81 (THRESHOLD 12 connections exceeded in 0 seconds) [**]
08/01-08:17:53.492732  [**] spp_portscan: portscan status from 216.228.171.81:
16 connections across 9 hosts: TCP(16), UDP(0) [**]
08/01-08:17:57.184759  [**] spp_portscan: End of portscan from 216.228.171.81:
TOTAL time(0s) hosts(9) TCP(16) UDP(0) [**]
```

As well as some portscanning, this is one of the people hitting the

beetle.ucs server.

63.21.4.50
DNS: 1Cust50.tnt1.downers-grove.il.da.uu.net
Alerts Found: none

That's weird. With the exception of the SMB Name Wildcards, this guy hasn't been doing anything. I wonder if it's a false positive in this case. Perhaps this guy needs a firewall or his PC is miss-configured.

169.254.113.62
DNS: "not found"
Alerts Found: none

This address didn't have any other alerts either. I'm not sure if that makes it more or less strange.

Top Talkers

Below are tables of the top 10 source IP addresses and the number of hits they recorded broken out by day.

August 1st.

IP Address	# of Hits
130.85.165.24	48,912
130.85.70.207	43,544
63.250.213.12	32,053
130.85.82.2	15,525
130.85.137.7	11,070
205.188.228.17	7,127
64.194.26.225	7,001
24.205.63.140	6,837
216.234.201.25	6,413
130.85.70.180	6,379

August 2nd.

IP Address	# of Hits
130.85.70.200	396,604
130.85.165.24	72,907
130.85.70.207	67,089
130.85.82.2	38,917
130.85.81.27	34,309
130.85.137.7	16,264
130.85.87.44	11,082
217.228.34.247	9,447
3.0.0.99	9,150
130.13.136.86	5,623

August 3rd.

IP Address	# of Hits
130.85.70.200	858,637

130.85.82.2	96,449
130.85.81.37	27,102
130.85.137.7	19,657
194.98.189.139	16,052
202.98.223.86	11,630
3.0.0.99	11,472
80.11.212.202	7,937
130.85.83.150	6,164
130.85.70.207	5,761

August 4th.

IP Address	# of Hits
130.85.84.234	964,424
130.85.70.200	936,537
130.85.83.150	92,240
216.228.171.81	30,098
130.85.70.133	24,918
24.138.61.171	22,146
130.85.87.50	19,873
211.232.192.153	18,358
219.96.171.20	14,237
3.0.0.99	14,007

August 5th.

IP Address	# of Hits
130.85.100.208	1,612,954
130.85.70.200	371,692
130.85.70.207	46,379
80.137.90.34	23,239
161.132.205.100	20,953
130.85.70.133	20,328
67.104.84.142	16,966
66.224.37.26	10,572
3.0.0.99	10,388

209.152.103.69	9,250

Hosts Investigated

Now, we'll take a look at the external addresses from the top 10 talkers to see if anything interesting turns up.

63.250.213.12

This address was mentioned earlier with respect to UDP SRC and DST outside network. He appears to be doing some streaming or some other type of multicast.

Output from ARIN WHOIS

```
OrgName:    Yahoo! Broadcast Services, Inc.
OrgID:      YAHO

NetRange:   63.250.192.0 - 63.250.223.255
CIDR:       63.250.192.0/19
NetName:    NETBLK2-YAHOOBS
NetHandle:  NET-63-250-192-0-1
Parent:     NET-63-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.YAHOO.COM
NameServer: NS2.YAHOO.COM
NameServer: NS3.YAHOO.COM
NameServer: NS4.YAHOO.COM
NameServer: NS5.YAHOO.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-
PORTABLE
RegDate:    1999-11-24
Updated:    2002-03-27

TechHandle: NA258-ARIN
TechName:   Netblock Admin, Netblock
TechPhone:  +1-408-349-7183
    TechEmail:  netblockadmin@yahoo-inc.com
```

DNS Name: dal-qcwm213012.bcst.yahoo.com

211.141.120.18

This address was involved in the compromise of 130.85.100.201, so I definitely wanted to get more information on it.

Output from ARIN WHOIS

```
OrgName:    Asia Pacific Network Information Centre
OrgID:      APNIC

NetRange:   210.0.0.0 - 211.255.255.255
CIDR:       210.0.0.0/7
NetName:    APNIC-CIDR-BLK2
NetHandle:  NET-210-0-0-0-1
Parent:
NetType:    Allocated to APNIC
NameServer: ns1.apnic.net
NameServer: ns3.apnic.net
NameServer: ns.ripe.net
NameServer: rs2.arin.net
NameServer: dns1.telstra.net
Comment:    This IP address range is not registered in
the ARIN database.
            For details, refer to the APNIC Whois
Database via
            WHOIS.APNIC.NET or
http://www.apnic.net/apnic-bin/whois2.pl
            ** IMPORTANT NOTE: APNIC is the Regional
Internet Registry
            for the Asia Pacific region. APNIC does
not operate networks
            using this IP address range and is not
able to investigate
            spam or abuse reports relating to these
addresses. For more
            help, refer to
http://www.apnic.net/info/faq/abuse

RegDate:    1996-07-01
Updated:    2002-09-11

OrgTechHandle: SA90-ARIN
OrgTechName:   System Administrator, System
OrgTechPhone:  +61 7 3858 3100
OrgTechEmail:
```

DNS Name: "not found"

80.62.155.240

I chose this address because it was related to the spp_http_decode alerts mentioned above. Specifically, this address alerted on some Trojan activity against 130.85.85.74, which was one of the top sources for the http_decode alerts.

Output from ARIN WHOIS

```
OrgName:    RIPE Network Coordination Centre
OrgID:      RIPE

NetRange:   80.0.0.0 - 80.255.255.255
CIDR:       80.0.0.0/8
NetName:    80-RIPE
NetHandle:  NET-80-0-0-0-1
Parent:
NetType:    Allocated to RIPE NCC
NameServer: NS.RIPE.NET
NameServer: AUTH62.NS.UU.NET
NameServer: NS3.NIC.FR
NameServer: SUNIC.SUNET.SE
NameServer: MUNNARI.OZ.AU
NameServer: NS.APNIC.NET
NameServer: SVC00.APNIC.NET
Comment:    These addresses have been further assigned
to users in
            the RIPE NCC region. Contact information
can be found in
            the RIPE database at whois.ripe.net

RegDate:
Updated:    2002-09-11

OrgTechHandle: RIPE-NCC-ARIN
OrgTechName:   Reseaux IP European Network Co-
ordination Centre S
OrgTechPhone:  +31 20 535 4444
OrgTechEmail:  nicdb@ripe.net
```

DNS Name: 0x503e9bf0.odnxx4.adsl-dhcp.tele.dk

203.213.58.38

This address is related to some of the beetle.ucs alerts discussed earlier. It was one of the top talkers for the beetle.ucs alert.

Output from ARIN WHOIS

```
OrgName:    Asia Pacific Network Information Centre
OrgID:      APNIC

NetRange:   202.0.0.0 - 203.255.255.255
CIDR:       202.0.0.0/7
NetName:    APNIC-CIDR-BLK
NetHandle:  NET-202-0-0-0-1
Parent:
NetType:    Allocated to APNIC
NameServer: ns1.apnic.net
NameServer: ns3.apnic.net
NameServer: ns.ripe.net
NameServer: rs2.arin.net
NameServer: dns1.telstra.net
Comment:    This IP address range is not registered in
the ARIN database.
            For details, refer to the APNIC Whois
Database via
            WHOIS.APNIC.NET or
http://www.apnic.net/apnic-bin/whois2.pl
            ** IMPORTANT NOTE: APNIC is the Regional
Internet Registry
            for the Asia Pacific region. APNIC does
not operate networks
            using this IP address range and is not
able to investigate
            spam or abuse reports relating to these
addresses. For more
            help, refer to
http://www.apnic.net/info/faq/abuse

RegDate:    1994-04-05
Updated:    2002-09-11

OrgTechHandle: SA90-ARIN
OrgTechName:   System Administrator, System
OrgTechPhone:  +61 7 3858 3100
OrgTechEmail:
```

DNS Name: syd-ts6-2600-038.tpgi.com.au.

216.228.171.81

This address was the source of a lot of SMB Name Wildcard alerts mentioned above.

Output from ARIN WHOIS

```
Bend Cable BENDCABLE (NET-216-228-160-0-1)
                                  216.228.160.0 -
216.228.191.255
bend cable communications BCCI228-DOCSIS (NET-216-228-
168-0-1)
                                  216.228.168.0 -
216.228.172.255
```

DNS Name: didn't resolve

Defensive Recommendations

After analyzing the data provided to me by the university, I am able to make the following conclusions and recommendations. First, the university appears to have fairly adequate security measures in place, but there is always room for improvement.

The security device that could use the most tweaking is the snort sensor, or sensors that generated the alerts provided to me by the university. The current configuration generates an overwhelming amount of alerts. I get the feeling that just like most universities, you want the network to be mostly open; however, you have the snort rules set up like it's a top secret military installation. For example, you have thousands of lines of alerts for things like packets leaving the network with a source address that's either a non routable IP or an IP that isn't a university network. I think it's pretty much standard practice that one of the first things you do from a security standpoint is to block that traffic with egress filters at the router. If you decide to allow it, don't waste resources alerting on it.

Allowing the Server Message Block (SMB) traffic to and from the Internet is another example where you're allowing it, but alerting on all the possible attacks. The traffic is a security nightmare and should be blocked. Again, if you decide to allow it, don't alert on it.

If some time could be spent tweaking the snort rules to make the log file sizes more manageable, it would eliminate a large amount of the "noise". Then, it would be easier for a security administrator to focus on

the real threats.

Speaking of the real threats, the number of alerts pales in comparison to the "noise" I was just talking about; however, they are the dangerous ones to watch out for. There are a lot of alerts of a "possible trojan server activity", port scanning, and pre-attack probes, which could all mean big problems. Since the victims are most likely student computers, it's not necessarily an imminent threat to the university, but it could lead to one. Additionally, the university may feel some responsibility for the security of its student's computers and would feel it prudent to clean up the victim's computers. Perhaps some literature (or some other venue) could be put out on about general computer security to help out the "less savvy" computer users.

Another area of alerts that you might want to take action on is the peer-to-peer (P2P) traffic. This is another way that trojans and viruses end up on student computers. You don't necessarily want to block this traffic (though it would be understandable and OK if you did), so again, maybe some type of education on the potential dangers of retrieving media from unknown sources would be beneficial.

In conclusion, I feel that taking the steps I've outlined above would help clean up the university network considerably. In addition, it would help bring other real threats to the surface that have historically been buried beneath the noise.

References

Snort

Marty Roesch
http://www.snort.org
http://www.sourcefire.com

TCP/IP Illustrated, Volume 1

W. Richard Stevens
Addison-Wesley
ISBN: 0-201-63346-9

WinMX
http://www.winmx.com

University of Washington

Dave Dittrich
http://staff.washington.edu/dittrich/misc/ddos/

CERT Coordination Center
http://www.cert.org

Network World Fusion
http://www.nwfusion.com

Neohapsis
http://www.neohapsis.com

University of Maryland Baltimore County
http://www.umbc.edu/

Edward Peck

GCIA Practical
http://www.giac.org/practical/Edward_Peck_GCIA.doc

Whitehats
http://www.whitehats.com

Toby Miller
http://www.sans.org/y2k/ecn.htm

ARIN
http://www.arin.net/

Dshield
http://www.dshield.org/

eEye Digital Security
http://www.eeye.com/

Appendix A.

```
#!/usr/bin/perl
#
#   vim:tabstop=8:shiftwidth=4:softtabstop=4
#
# cory steers 20020904 - script to parse incidents.org alert, scans,
#                        and oos files
#

use strict;
use Getopt::Long;
use Time::Local;

sub fail;
sub extractArgs;

use vars qw(%args $pair $srcIP $quiet $debug %topSrcHash
%topAlertHash
    $alert $found $file @file $i $j @top10IP @top10Alert @temp);


#+++++++++++++#
# begin code #
#+++++++++++++#

%args = extractArgs();

$quiet = 1 if defined $args{"quiet"};
$debug = 1 if defined $args{"debug"};

# process file once for list of source directories

for(@file) {

    $file = $_;
    open (FILE, $file);

    while(<FILE>) {
      print STDERR "LOG ENTRY: $_\n" if $debug eq 1;
      grep { / (\d\d?\d?\.\d\d?\d?\.\d\d?\d?\.\d\d?\d?)\:/;
          $srcIP = $1;
      } $_;

      if (defined $srcIP) {
          print STDERR "Matched IP Address : $_\n" if $debug eq 1;
          $topSrcHash{${srcIP}}++;

          $srcIP = undef;
      }

      # for some reason some of the log entries aren't
      # seperated by a carraige return
      @temp = split(/\s\[\*\*\]/,$_);
```

```
      # if less than three, it's garbage
      unless (scalar @temp lt 3) {
          shift @temp;
          $alert = shift @temp;
          shift @temp;

          # scans files will return ""
          $alert = undef if $alert eq "";
          # only interested in portscans if it leads to something
          # else
          $alert = undef if $alert =~ m/portscan/i;
          next if ! defined $alert;

          print STDERR qq(Matched alert: $alert\n) if $debug eq 1;
          $topAlertHash{${alert}}++;
      }

      $alert = undef;
    }

    close FILE;
}

# sort the hash by # of hits per IP address (key) to get the top 10

foreach $pair (keys %topSrcHash) {

    print STDERR "working $pair now\n" if $debug eq 1;

    if (scalar @top10IP > 0) {
      undef $found;

      for($i = 0; $i < scalar @top10IP; $i++) {

          if ($topSrcHash{$pair} > $topSrcHash{$top10IP[$i]}) {

            print STDERR "$pair has more hits than $i\n" if $debug eq
1;

            $found = 1;
            @temp = splice @top10IP, $i, ((scalar @top10IP) - $i);
            if (((scalar @top10IP) + (scalar @temp)) >= 10) {
                delete @temp[(scalar @temp - 1)];
            }

            $top10IP[$i] = $pair;
            push @top10IP, @temp;
            $i = scalar @top10IP;
          }
      }

      if ((scalar @top10IP < 10) and ($found eq undef)) { push
@top10IP, $pair; }

    } else { push @top10IP, $pair; }
}

# sort the hash by # of hits per alert (key) to get the top 10
```

```
foreach $pair (keys %topAlertHash) {

    print STDERR "working $pair now\n" if $debug eq 1;

    if (scalar @top10Alert > 0) {
      undef $found;

      for($i = 0; $i < scalar @top10Alert; $i++) {

          if ($topAlertHash{$pair} > $topAlertHash{$top10Alert[$i]})
{

            print STDERR "$pair has more hits than $i\n" if $debug eq
1;

            $found = 1;
            @temp = splice @top10Alert, $i, ((scalar @top10Alert) -
$i);
            if (((scalar @top10Alert) + (scalar @temp)) >= 10) {
                delete @temp[(scalar @temp - 1)];
            }

            $top10Alert[$i] = $pair;
            push @top10Alert, @temp;
            $i = scalar @top10Alert;
          }
      }

      if ((scalar @top10Alert < 10) and ($found eq undef)) { push
@top10Alert, $pair; }

    } else { push @top10Alert, $pair; }
}

# print 10 top with number of hits

print qq(\n\n\tTop 10 "Talkers"\n\n),
qq(   IP\t\t# of Hits\n\n);
map { print qq($_\t$topSrcHash{$_}\n); } @top10IP;

print qq(\n\n\tTop 10 Alerts\n\n),
qq(   Alert\t\t# of Hits\n\n);
map { print qq($_\t$topAlertHash{$_}\n); } @top10Alert;


print qq(\nRun with the "-quiet" flag to get less output\n) unless
$quiet eq 1;

exit;


sub fail {
    if ($_[1] == 1) {
      die $_[0];
    } else {
      warn $_[0];
    }
```

```
}


sub useage {
    print qq(USEAGE: $0 <OPTIONS>\n);
    print qq(\nOPTIONS:\n);
    print qq(\t-file\t\t\t= file to process (can have multiple)\n);
    print qq(\t-quiet or -q\t\t= suppress output\n\n);
    print qq(\t-debug or -d\t\t= provide additional output\n\n);
    print qq(\t-help or -h\t\t= print this help\n\n);
    exit 2;
}

sub extractArgs {
    my %args = ();
    my $rc = 0;

    $rc = GetOptions(\%args,
      "help|h|?",
      "quiet|q",
      "debug|d",
      "file|f:s"  => \@file);

    if (! $rc) {
      fail(qq(\n001 - Error parsing command line flags!\n),0);
      useage();
    }

    useage() if defined $args{"help"};
    useage() unless ((scalar @file) > 0);

    unless ((scalar @file) > 0) {
      fail(qq(\n002 - Must define at least one "file"\n),0);
      useage();
    }

    return (%args);
}
```

Appendix B

```
#!/usr/bin/perl
#
# cory steers 20020910 - script build counts for line graph

use strict;
use Getopt::Long;
use Time::Local;

sub fail;
sub extractArgs;

use vars qw(%args $value $quiet $debug %hourHash
    @file $month $day $hour %monthnam %fixday);


#+++++++++++++#
# begin code #
#+++++++++++++#

%monthnam = ("Jan" => "01", "Feb" => "02", "Mar" => "03", "Apr" => "04",
   "May" => "05", "Jun" => "06", "Jul" => "07", "Aug" => "08",
   "Sep" => "09", "Oct" => 10, "Nov" => 11, "Dec" => 12);

%fixday = (1 => "01", 2 => "02", 3 => "03", 4 => "04", 5 => "05",
   6 => "06", 7 => "07", 8 => "08", 9 => "09");

%args = extractArgs();

$quiet = 1 if defined $args{"quiet"};
$debug = 1 if defined $args{"debug"};

for(@file) {

    open (FILE, $_);

    while(<FILE>) {

          undef $day;

# for alert and oos files
          grep { /\A(\d\d)\/(\d\d)\-(\d\d)\:\d\d\:\d\d\s*/;
                $month = $1; $day = $2; $hour = $3; } $_;

          if (defined $day) {
                $hourHash{${month}."-".${day}."-".${hour}}++;
                next;
          }

# for scans files
          grep { /\A([a-z,A-Z]{3})  ?(\d\d?) (\d\d)\:\d\d\:\d\d\w*/;
                $month = $monthnam{$1}; $day = $2; $hour = $3; } $_;

          if (defined $day) {
                if ($day < 10) { $day = $fixday{$day};
                      $hourHash{${month}."-".${day}."-".${hour}}++;
                }
          }

    }
```

```
    close FILE;
}

foreach $value (sort keys %hourHash) {

    print qq($value,$hourHash{$value}\n);

} # for each

sub fail {
    if ($_[1] == 1) {
   die $_[0];
    } else {
   warn $_[0];
    }
}


sub useage {
    print qq(USEAGE: $0 <OPTIONS>\n);
    print qq(\nOPTIONS:\n);
    print qq(\t-file\t\t\t= file to process (can have multiple)\n);
    print qq(\t-quiet or -q\t\t= suppress output\n\n);
    print qq(\t-debug or -d\t\t= provide additional output\n\n);
    print qq(\t-help or -h\t\t= print this help\n\n);
    exit 2;
}

sub extractArgs {
    my %args = ();
    my $rc = 0;

    $rc = GetOptions(\%args,
   "help|h|?",
   "quiet|q",
   "debug|d",
   "file|f:s"    => \@file);

    if (! $rc) {
   fail(qq(\n001 - Error parsing command line flags!\n),0);
   useage();
    }

    useage() if defined $args{"help"};
    useage() unless((scalar @file) > 0);

    return (%args);
}
```