
0.0 Administrivia

0.1 **Network Info**

One major advantage of working at an eduactional institution is getting fast (FastEthernet or ATM) network feeds. One disadvantage of working at that same education institution can be the lack of a suitable computer budget. Monitoring a 100Mbps network effectively can be a bit of a challenge, especially on a limited budget. To help understand where I'm getting my traces from and as a convenience to others who may be in the same predicament, I shall offer my Top Secret hints on how to be Big Brother On A Budget. After investigating a few possible solutions, it became clear that the homebrew solution was the way to go.

We get a FastEthernet feed on fibre. This is the first problem, since fibre tap devices (splitters, hubs, multiport repeaters and taps) are either nonexistent or impossibly expensive. The first stage in the chain is a fibre-to-copper transceiver. This allows me to put a hub between our feed and our router.

Attached to this hub is the first sniffer, running tcpdump 3.5 and Snort 1.7 (beta version of the week). The sniffer machines are pIII 600's on AOpen AX63pro mainboards. Network cards are the usual EtherExpress Pro100+. Operating system is OpenBSD 2.8, with IPF 3.3.18 configured to block all inbound and outbound packets on the sniffing interface. This does not interfere with sniffing since IPF (the filter) operates independantly from BPF (the network tap).

The router and firewall are both PC/Unix solutions; currently running OpenBSD 2.8, with AltQ 3.0, IPFilter 3.4.15 and Zebra 0.89a. Hardware list includes ASUS P3B-F mainboard, 700MHz Intel Celeron, three Intel EtherExpress Pro100+ NICs, and 256MB of memory. That may be a bit overkill, but it's all commodity hardware and really quite inexpensive.

The firewall is then connected to a series of meshed HP ProCurve 4000M switches. I use the SPAN port (Switch Port ANalyzer) to monitor the link to the firewall. All ethernet frames to and from the firewall get copied to this port which is also connected to a "stealth" interface. This configuration allows me to see what is hitting the firewall (from either side) and what passed through.

Both of the sniffer stations as well as the firewall report to my analyzer box, another pIII 600, this one with 256MB of memory and 280GB of striped UDMA66 disk. The analyzer logs Snort reports to a MySQL 3.23.28b database, for display by ACID 0.9.5b9. Besides keeping a database of interesting things to look for, I have a reasonable quantity of fairly quick disk which allows me to keep full packet dumps for quite a long time. I generate about 750 - 1000 MB of traffic each day, which means that I can have nearly 8 months worth of all packets.

0.2 Log Formats

Here is the same packet (generated by 'synscan' in case you were wondering), displayed a number of different ways by the various tools I use. Each has its strengths and weaknesses. Wherever possible, I chose to not resolve names, since that makes the output easier to parse, and runs a good deal faster since we don't have to do a whole bunch of name lookups. This is a big win if your network connection is down. Throughout this document, the only occurance of RFC1918 addresses is where I have my network replaced with 10.10.8.0/23. This should avoid some confusion regarding the use of these addresses to sometimes represent the destination net, sometimes represent the source net, and sometimes these addresses are actually seen live. In this case any address you see here is what came off the IDS, except for the 10-space addresses I used to sanitize my netblock.

```
=======================================================================
[ tcpdump -lenx ]======================================================
=======================================================================
14:24:34.147653 0:10:5a:9c:91:94 0:2:b3:7:ee:ed 0800 54: 10.10.206.4
+10.10.10.1.21: SF 1262562322:1262562322(0) win 1028
                         4500 0028 9a02 0000 2a06 579d 0a0a 082e
                         0a0a 0801 0015 0015 4b41 2c12 3419 6f64
                         5003 0404 f1b1 0000

=======================================================================
[ snort -dvr ]=========================================================
=======================================================================
01/08-14:24:34.147653 10.10.8.46:21 -> 10.10.8.1:21
TCP TTL:42 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x4B412C12  Ack: 0x34196F64  Win: 0x404  TcpLen: 20

=======================================================================
[ tcpshow -noHostNames ]===============================================
=======================================================================
Packet 1
TIME:   14:24:34.147653
LINK:   00:10:5A:9C:91:94 -> 00:02:B3:07:EE:ED type=IP
  IP:   10.10.8.46 -> 10.10.8.1 hlen=20 TOS=00 dgramlen=40 id=9A02
        MF/DF=0/0 frag=0 TTL=42 proto=TCP cksum=579D
 TCP:   port ftp -> ftp seq=1262562322 ack=0874082148
        hlen=20 (data=0) UAPRSF=000011 wnd=1028 cksum=F1B1 urg=0
DATA:

=======================================================================
[ tcpshow -noHostNames -verbose ]======================================
=======================================================================
Packet 1
        Timestamp:                      14:24:34.147653
        Source Ethernet Address:        00:10:5A:9C:91:94
        Destination Ethernet Address:   00:02:B3:07:EE:ED
        Encapsulated Protocol:          IP
IP Header
        Version:                        4
        Header Length:                  20 bytes
        Service Type:                   0x00
        Datagram Length:                40 bytes
        Identification:                 0x9A02
        Flags:                          MF=off, DF=off
        Fragment Offset:                0
        TTL:                            42
        Encapsulated Protocol:          TCP
        Header Checksum:                0x579D
        Source IP Address:              10.10.8.46
        Destination IP Address:         10.10.8.1
TCP Header
        Source Port:                    21 (ftp)
        Destination Port:               21 (ftp)
        Sequence Number:                1262562322
        Acknowledgement Number:         0874082148
        Header Length:                  20 bytes (data=0)
        Flags:                          URG=off, ACK=off, PSH=off
                                        RST=off, SYN=on,  FIN=on
        Window Advertisement:           1028 bytes
        Checksum:                       0xF1B1
        Urgent Pointer:                 0
TCP Data
```

1.0 Analyze 4 Detects

1.1 **Overflow / x86 NOP Sled (False Alarm)**

0 - Packet Traces

```
--------------------------------------------------------------------
[**]   IDS181 - MISC - Shellcode X86 NOPS [**]
01/12-14:58:16.554459 216.169.70.17:80 -> 10.10.9.88:2294
TCP TTL:255 TOS:0x0 ID:0 IpLen:20 DgmLen:1177
***AP*** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
FF E0 00 10 4A 46 49 46 00 01 02 01 00 48 00 48  ....JFIF.....H.H
00 00 FF ED 02 74 50 68 6F 74 6F 73 68 6F 70 20  .....tPhotoshop
33 2E 30 00 38 42 49 4D 03 ED 00 00 00 00 00 10  3.0.8BIM........
00 48 00 00 00 01 00 01 00 48 00 00 00 01 00 01  .H.......H......
38 42 49 4D 03 F3 00 00 00 00 00 08 00 00 00 00  8BIM............
00 00 00 01 38 42 49 4D 27 10 00 00 00 00 00 0A  ....8BIM'.......
00 01 00 00 00 00 00 00 00 02 38 42 49 4D 03 F5  ..........8BIM..
00 00 00 00 00 48 00 2F 66 66 00 01 00 6C 66 66  .....H./ff...lff
00 06 00 00 00 00 00 01 00 2F 66 66 00 01 00 A1  ........./ff....
99 9A 00 06 00 00 00 00 00 01 00 32 00 00 00 01  ...........2....
00 5A 00 00 00 06 00 00 00 00 00 01 00 35 00 00  .Z..........5..
00 01 00 2D 00 00 00 06 00 00 00 00 00 01 38 42  ...-.........8B
```

```
49 4D 04 00 00 00 00 00 00 02 00 00 38 42 49 4D  IM..........8BIM
04 02 00 00 00 00 00 02 00 00 38 42 49 4D 04 07  ..........8BIM..
00 00 00 00 01 94 00 00 00 80 00 00 00 01 00 00  ................
01 80 00 00 01 80 00 18 00 01 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 38 42 49 4D 04 06  ..........8BIM..
00 00 00 00 00 02 00 04 FF EE 00 0E 41 64 6F 62  ............Adob
65 00 64 00 00 00 00 01 FF DB 00 84 00 06 04 04  e.d.............
04 05 04 06 05 05 06 09 06 05 06 09 0B 08 06 06  ................
08 0B 0C 0A 0A 0B 0A 0A 0C 10 0C 0C 0C 0C 0C 0C  ................
10 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C  ................
0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 01 07 07  ................
07 0D 0C 0D 18 10 10 18 14 0E 0E 0E 14 14 0E 0E  ................
0E 0E 14 11 0C 0C 0C 0C 0C 11 11 0C 0C 0C 0C 0C  ................
0C 11 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C  ................
0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C FF C0  ................
00 11 08 00 07 03 F0 03 01 11 00 02 11 01 03 11  ................
01 FF C4 01 A2 00 00 00 07 01 01 01 01 01 00 00  ................
00 00 00 00 00 00 04 05 03 02 06 01 00 07 08 09  ................
0A 0B 01 00 02 02 03 01 01 01 01 01 00 00 00 00  ................
00 00 00 01 00 02 03 04 05 06 07 08 09 0A 0B 10  ................
00 02 01 03 03 02 04 02 06 07 03 04 02 06 02 73  ...............s
01 02 03 11 04 00 05 21 12 31 41 51 06 13 61 22  .......!.1AQ..a"
71 81 14 32 91 A1 07 15 B1 42 23 C1 52 D1 E1 33  q..2.....B#.R..3
16 62 F0 24 72 82 F1 25 43 34 53 92 A2 B2 63 73  .b.$r..%C4S...cs
C2 35 44 27 93 A3 B3 36 17 54 64 74 C3 D2 E2 08  .5D'...6.Tdt....
26 83 09 0A 18 19 84 94 45 46 A4 B4 56 D3 55 28  &.......EF..V.U(
1A F2 E3 F3 C4 D4 E4 F4 65 75 85 95 A5 B5 C5 D5  ........eu......
E5 F5 66 76 86 96 A6 B6 C6 D6 E6 F6 37 47 57 67  ..fv........7GWg
77 87 97 A7 B7 C7 D7 E7 F7 38 48 58 68 78 88 98  w........8HXhx..
A8 B8 C8 D8 E8 F8 29 39 49 59 69 79 89 99 A9 B9  ......)9IYiy....
C9 D9 E9 F9 2A 3A 4A 5A 6A 7A 8A 9A AA BA CA DA  ....*:JZjz......
EA FA 11 00 02 02 01 02 03 05 05 04 05 06 04 08  ................
03 03 6D 01 00 02 11 03 04 21 12 31 41 05 51 13  ..m......!.1A.Q.
61 22 06 71 81 91 32 A1 B1 F0 14 C1 D1 E1 23 42  a".q..2.......#B
```

```
15 52 62 72 F1 33 24 34 43 82 16 92 53 25 A2 63  .Rbr.3$4C...S%.c
B2 C2 07 73 D2 35 E2 44 83 17 54 93 08 09 0A 34  ...s.5.D..T....4
7C                                               |
```

---

1 - Source of Trace

Production network at a Canadian university.

2 - Detect Generated By ...

Snort 1.7b8, logging to a MySQL 3.23.28 database for display by ACID 0.95b9. Backup analysis capability provided by tcpdump 3.5.2 and tcpshow 1.7.4. I use ACID to find the interesting packets and tcpshow to pretty-print the packet for inclusion into reports. I'm using the January 7, 2001 version of "snortfull.conf" from www.snort.org. The rule which matched this packet was:

```
alert tcp !$HOME_NET any -> $HOME_NET any (msg:
"IDS181 - MISC - Shellcode X86 NOPS"; content:
"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90|"; flags: PA;)
```

3 - Probability of source address spoofing

Slim to none. This is a packet from the "middle" of a tcp stream, which means that the 3-way handshake completed. Granted, there are a number of ways to inject packets into the stream, but in this context, it is nearly certain that the source host is who the packet says it is. Furthermore, the stream continued after this packet, indicating that this packet is legitimate, and the host for which it was destined accepted this packet without much fuss.

4 - Description of attack

This signature is platform dependent. The general idea is that we look for a large number of contiguous identical strings, where the value of a string is the no-op instruction on a given platform. In this case, snort saw a bunch of 0x90 bytes, and decided that it could be a buffer overflow attempt.

5 - Attack Mechanism

Buffer overflow attacks often include large numbers of no-op instructions to help overflow the buffer with data not likely to interfere with the attack and to align the code on a reasonable word boundary. Some processors (Motorola 680X0 for example) do not like executing code which is not aligned on an even address. I'm sure that there are others that demand 4 or 8 byte alignment. A packet containing malicious data is sent to a vulnerable daemon. This will often contain part of a legitimate transaction, and possibly some junk to fill up the buffer, followed by no-ops to align the code in memory. Finally comes the code itself which at its simplest is a bit of setup and a call to execute an arbitrary command (/bin/sh, '/bin/rm -rf /', whoami ...). One signature to look for would be a large number of no-op instructions, whose value will differ based on the target processor. This is, in fact what the above rule looks for, namely 24 x86-compatible no-op instructions.

6 - Correlations

If every packet thought to be an buffer overflow attack were printed out on normal letter-size paper, that pile of paper would probably reach the moon. But just for a moment let's pretend that this is a hostile packet. A few good references to check out include Aleph1's excellent paper "Smashing the stack for fun and profit" in Phrack46. BugTraq archives can be searched at http://www.securityfocus.com/ . This packet happens to be a false alarm. I verified this by checking the firewall's connection logs and by digging through my packet vault and replaying the actual download. Also, this packet happens to be part of an jpeg image created by Adobe Photoshop 3.0 on Macintosh hardware. How can I make those assertions? Clue #1: "JFIF" - This string is used as a signature to mark a JPEG Format Image File. Clue #2: "Photoshop3.0" - JPEG images are allowed to have comments, which often include the image creator. I'd have to check my old mac to be sure, but if memory serves correctly, Photoshop 3 was the first version to support the JPEG image as we know it today. Also, I don't think Photoshop for Windows was available in that version. Clue #3: "8BIM" - This is the application creator for Photoshop on Macintosh. appl/8BIM (Photoshop) appears to have created this JFIF/8BIM (JPEG image file). Given all that, I'd say that it's safe to call this a Mac Photoshop 3 JPEG file.

7 - Evidence of Active Targeting

In the case of false alarms I would say that there isn't really targeting since there isn't a real attack. Targeting, at least for me, implies something more sinister. That said this is certainly not a "little lost packet" or a network scan. It's not an informational broadcast. This is traffic specifically for one from another. This traffic has a purpose. Which makes it just about as targeted as a packet gets.

8 - Severity

Criticality: Linux workstation. You can do nasty stuff with it if you get root, but it's not anything really important. It's not as harmless as MS-DOS though, and not as serious as a DoS on the coffee maker. C=2

Lethality: This was not an attack, so it can't be lethal. L=0

Host Countermeasures: This machine is a well-patched linux machine. It runs no services, and has had many possibly buggy daemons de-installed. H=5

Net Countermeasures: We have a good firewall which only allows FTP access to our official FTP servers, and all traffic must pass through this firewall. We also reject "non-standard" packets, such as things with reserved bits SYN-FIN set. Logging works well. Firewall has not had a thorough audit, but it seems to be doing the job. N=4

(criticality+lethality)-(host+net)=severity
(3 + 0) - (5 + 4) = -6

9 - Defensive Recommendations

Not much beyond a warning to not jump to conclusions. In this case the network performed as it should have; the connection was permitted according to site policy and logged should later analysis become necessary. The logging system functioned as it was designed to do. Delete this alert from database and move to next.

10 - Test Question

The one command that would be most useful in identifying the purpose of this packet is:
a) nslookup
b) strings
c) xv
d) klmgrd (Konshus license manager)

Answer: A. this IP resolves to www.cms.org, and the source port of this packet is port www, which makes it very probable that this really is a web image and not a clever attempt to overflow klmgrd and not get caught.

1.2 A spammer in action

0 - Packets

```
10:04:44.715550 63.38.124.251.2556 > 10.10.8.158.25: S
10:04:45.395204 63.38.124.251.2556 > 10.10.8.158.25: S
10:04:46.080225 63.38.124.251.2556 > 10.10.8.158.25: S
10:04:54.179849 63.38.124.251.2597 > 10.10.8.71.25: S
10:04:54.186432 63.38.124.251.2598 > 10.10.8.72.25: S
10:04:54.895317 63.38.124.251.2597 > 10.10.8.71.25: S
10:04:54.895367 63.38.124.251.2598 > 10.10.8.72.25: S
10:04:55.636915 63.38.124.251.2598 > 10.10.8.72.25: S
10:04:55.647391 63.38.124.251.2597 > 10.10.8.71.25: S
10:04:56.007753 63.38.124.251.2616 > 10.10.8.44.25: S
10:04:56.701015 63.38.124.251.2616 > 10.10.8.44.25: S
10:04:57.385447 63.38.124.251.2616 > 10.10.8.44.25: S
10:33:59.359030 63.38.124.251.1455 > 10.10.8.85.25: S
10:34:00.030285 63.38.124.251.1455 > 10.10.8.85.25: S
10:34:00.732631 63.38.124.251.1455 > 10.10.8.85.25: S
10:36:58.360808 63.38.124.251.2315 > 10.10.8.174.25: S
10:36:59.017112 63.38.124.251.2315 > 10.10.8.174.25: S
10:36:59.723840 63.38.124.251.2315 > 10.10.8.174.25: S
11:08:03.676222 63.38.124.251.3462 > 10.10.8.12.25: S
11:08:04.373171 63.38.124.251.3462 > 10.10.8.12.25: S
11:08:05.068952 63.38.124.251.3462 > 10.10.8.12.25: S
11:14:16.912218 63.38.124.251.1447 > 10.10.8.130.25: S
11:14:17.569698 63.38.124.251.1447 > 10.10.8.130.25: S
11:14:18.248317 63.38.124.251.1447 > 10.10.8.130.25: S
11:48:51.782731 63.38.124.251.3093 > 10.10.8.32.25: S
11:48:52.117656 63.38.124.251.3095 > 10.10.8.74.25: S
11:48:52.481841 63.38.124.251.3093 > 10.10.8.32.25: S
11:48:52.773590 63.38.124.251.3095 > 10.10.8.74.25: S
11:48:53.176248 63.38.124.251.3093 > 10.10.8.32.25: S
11:48:53.469632 63.38.124.251.3095 > 10.10.8.74.25: S
12:03:36.077614 63.38.124.251.3093 > 10.10.8.173.25: S
12:03:36.883468 63.38.124.251.3093 > 10.10.8.173.25: S
12:03:37.681370 63.38.124.251.3093 > 10.10.8.173.25: S
13:12:07.686598 63.38.124.251.4046 > 10.10.8.152.25: S
13:12:08.500399 63.38.124.251.4046 > 10.10.8.152.25: S
13:12:09.301744 63.38.124.251.4046 > 10.10.8.152.25: S
13:46:32.697371 63.38.124.251.2687 > 10.10.8.164.25: S
13:46:33.529838 63.38.124.251.2687 > 10.10.8.164.25: S
13:46:34.086030 63.38.124.251.2692 > 10.10.8.34.25: S
13:46:34.343796 63.38.124.251.2687 > 10.10.8.164.25: S
13:46:34.835802 63.38.124.251.2692 > 10.10.8.34.25: S
13:46:35.457858 63.38.124.251.2703 > 10.10.8.168.25: S
13:46:35.644416 63.38.124.251.2692 > 10.10.8.34.25: S
13:46:36.246270 63.38.124.251.2703 > 10.10.8.168.25: S
13:46:37.040939 63.38.124.251.2703 > 10.10.8.168.25: S
14:13:00.751814 63.38.124.251.2916 > 10.10.8.80.25: S
```

```
14:13:01.295892 63.38.124.251.2920 > 10.10.8.176.25: S
14:13:01.554264 63.38.124.251.2916 > 10.10.8.80.25: S
14:13:02.049395 63.38.124.251.2920 > 10.10.8.176.25: S
14:13:02.351681 63.38.124.251.2916 > 10.10.8.80.25: S
14:13:02.848612 63.38.124.251.2920 > 10.10.8.176.25: S
14:56:15.442562 63.38.124.251.4127 > 10.10.8.153.25: S
14:56:16.274643 63.38.124.251.4127 > 10.10.8.153.25: S
14:56:17.059181 63.38.124.251.4127 > 10.10.8.153.25: S
14:56:17.721415 63.38.124.251.4133 > 10.10.8.129.25: S
14:56:18.550965 63.38.124.251.4137 > 10.10.8.131.25: S
14:56:18.558508 63.38.124.251.4133 > 10.10.8.129.25: S
14:56:19.373244 63.38.124.251.4137 > 10.10.8.131.25: S
14:56:19.373249 63.38.124.251.4133 > 10.10.8.129.25: S
14:56:19.985885 63.38.124.251.4148 > 10.10.8.132.25: S
14:56:20.013741 63.38.124.251.4149 > 10.10.8.133.25: S
14:56:20.027010 63.38.124.251.4150 > 10.10.8.134.25: S
14:56:20.186841 63.38.124.251.4137 > 10.10.8.131.25: S
14:56:20.187248 63.38.124.251.4151 > 10.10.8.135.25: S
14:56:20.271444 63.38.124.251.4152 > 10.10.8.136.25: S
14:56:20.400431 63.38.124.251.4154 > 10.10.8.138.25: S
14:56:20.442496 63.38.124.251.4155 > 10.10.8.137.25: S
14:56:20.757300 63.38.124.251.4148 > 10.10.8.132.25: S
14:56:20.890133 63.38.124.251.4149 > 10.10.8.133.25: S
14:56:20.989740 63.38.124.251.4151 > 10.10.8.135.25: S
14:56:20.989902 63.38.124.251.4150 > 10.10.8.134.25: S
14:56:21.074753 63.38.124.251.4152 > 10.10.8.136.25: S
14:56:21.272186 63.38.124.251.4155 > 10.10.8.137.25: S
14:56:21.611203 63.38.124.251.4148 > 10.10.8.132.25: S
14:56:21.688358 63.38.124.251.4149 > 10.10.8.133.25: S
14:56:21.808456 63.38.124.251.4151 > 10.10.8.135.25: S
14:56:21.815157 63.38.124.251.4150 > 10.10.8.134.25: S
14:56:21.895474 63.38.124.251.4152 > 10.10.8.136.25: S
14:56:22.102237 63.38.124.251.4155 > 10.10.8.137.25: S
14:56:24.608477 63.38.124.251.4177 > 10.10.8.139.25: S
14:56:25.487205 63.38.124.251.4177 > 10.10.8.139.25: S
14:56:26.277560 63.38.124.251.4177 > 10.10.8.139.25: S
14:56:27.794441 63.38.124.251.4195 > 10.10.8.140.25: S
14:56:28.413266 63.38.124.251.4198 > 10.10.8.141.25: S
14:56:28.598670 63.38.124.251.4195 > 10.10.8.140.25: S
14:56:29.307612 63.38.124.251.4198 > 10.10.8.141.25: S
14:56:29.405479 63.38.124.251.4195 > 10.10.8.140.25: S
14:56:30.097338 63.38.124.251.4198 > 10.10.8.141.25: S
14:56:41.460990 63.38.124.251.4261 > 10.10.8.142.25: S
14:56:42.295605 63.38.124.251.4261 > 10.10.8.142.25: S
14:56:43.088776 63.38.124.251.4261 > 10.10.8.142.25: S
15:44:39.043907 63.38.124.251.2988 > 10.10.8.29.25: S
15:44:39.779002 63.38.124.251.2995 > 10.10.8.162.25: S
15:44:39.938644 63.38.124.251.2988 > 10.10.8.29.25: S
15:44:40.676367 63.38.124.251.2995 > 10.10.8.162.25: S
15:44:40.858414 63.38.124.251.2988 > 10.10.8.29.25: S
16:23:05.426888 63.38.124.251.2826 > 10.10.8.154.25: S
16:23:05.594855 63.38.124.251.2827 > 10.10.8.15.25: S
16:23:05.595200 10.10.8.15.25 > 63.38.124.251.2827: SA
16:23:06.253636 63.38.124.251.2826 > 10.10.8.154.25: S
16:23:06.273658 63.38.124.251.2831 > 10.10.8.159.25: S
16:23:06.898189 63.38.124.251.2835 > 10.10.8.160.25: S
16:23:07.192475 63.38.124.251.2831 > 10.10.8.159.25: S
```

```
16:23:07.196102 63.38.124.251.2826 > 10.10.8.154.25: S
16:23:07.316798 63.38.124.251.2840 > 10.10.8.161.25: S
16:23:07.730488 63.38.124.251.2835 > 10.10.8.160.25: S
16:23:08.033837 63.38.124.251.2831 > 10.10.8.159.25: S
16:23:08.141122 63.38.124.251.2840 > 10.10.8.161.25: S
16:23:08.553475 63.38.124.251.2835 > 10.10.8.160.25: S
16:23:08.966677 63.38.124.251.2840 > 10.10.8.161.25: S
16:23:09.870750 63.38.124.251.2856 > 10.10.8.163.25: S
16:23:10.644936 63.38.124.251.2856 > 10.10.8.163.25: S
16:23:11.368169 63.38.124.251.2865 > 10.10.8.165.25: S
16:23:11.441703 63.38.124.251.2856 > 10.10.8.163.25: S
16:23:12.151366 63.38.124.251.2865 > 10.10.8.165.25: S
16:23:12.549095 63.38.124.251.2872 > 10.10.8.166.25: S
16:23:12.907427 63.38.124.251.2875 > 10.10.8.167.25: S
16:23:12.973867 63.38.124.251.2865 > 10.10.8.165.25: S
16:23:13.341915 63.38.124.251.2872 > 10.10.8.166.25: S
16:23:13.644768 63.38.124.251.2875 > 10.10.8.167.25: S
16:23:14.199468 63.38.124.251.2872 > 10.10.8.166.25: S
16:23:14.432115 63.38.124.251.2875 > 10.10.8.167.25: S
16:23:15.417458 63.38.124.251.2895 > 10.10.8.170.25: S
16:23:15.694676 63.38.124.251.2896 > 10.10.8.171.25: S
16:23:16.127011 63.38.124.251.2895 > 10.10.8.170.25: S
16:23:16.432253 63.38.124.251.2896 > 10.10.8.171.25: S
16:23:16.845966 63.38.124.251.2895 > 10.10.8.170.25: S
16:23:17.123984 63.38.124.251.2896 > 10.10.8.171.25: S
16:23:17.343889 63.38.124.251.2906 > 10.10.8.172.25: S
16:23:17.576188 63.38.124.251.2886 > 10.10.8.169.25: S
16:23:18.157174 63.38.124.251.2906 > 10.10.8.172.25: S
16:23:18.368527 63.38.124.251.2886 > 10.10.8.169.25: S
16:23:19.011626 63.38.124.251.2906 > 10.10.8.172.25: S
16:23:19.756446 63.38.124.251.2932 > 10.10.8.175.25: S
16:23:20.566309 63.38.124.251.2932 > 10.10.8.175.25: S
16:23:21.376707 63.38.124.251.2932 > 10.10.8.175.25: S
16:23:26.286295 63.38.124.251.2978 > 10.10.8.177.25: S
16:23:27.084202 63.38.124.251.2978 > 10.10.8.177.25: S
16:23:27.890139 63.38.124.251.2978 > 10.10.8.177.25: S
16:23:32.465555 63.38.124.251.3010 > 10.10.8.178.25: S
16:23:33.373399 63.38.124.251.3010 > 10.10.8.178.25: S
16:23:34.253130 63.38.124.251.3010 > 10.10.8.178.25: S
16:32:31.960406 63.38.124.251.1881 > 10.10.8.145.25: S
16:32:32.677544 63.38.124.251.1881 > 10.10.8.145.25: S
16:32:32.815691 63.38.124.251.1886 > 10.10.8.82.25: S
16:32:33.389635 63.38.124.251.1881 > 10.10.8.145.25: S
16:32:33.586060 63.38.124.251.1886 > 10.10.8.82.25: S
16:32:34.385789 63.38.124.251.1886 > 10.10.8.82.25: S
16:35:18.631288 63.38.124.251.2697 > 10.10.8.27.25: S
16:35:19.364477 63.38.124.251.2697 > 10.10.8.27.25: S
16:35:20.028250 63.38.124.251.2704 > 10.10.8.28.25: S
16:35:20.056055 63.38.124.251.2697 > 10.10.8.27.25: S
16:35:20.763394 63.38.124.251.2704 > 10.10.8.28.25: S
16:35:21.488982 63.38.124.251.2704 > 10.10.8.28.25: S
16:35:25.274105 63.38.124.251.2732 > 10.10.8.30.25: S
16:35:26.083469 63.38.124.251.2732 > 10.10.8.30.25: S
16:35:26.316001 63.38.124.251.2739 > 10.10.8.31.25: S
16:35:26.661623 63.38.124.251.2742 > 10.10.8.33.25: S
16:35:26.706600 63.38.124.251.2743 > 10.10.8.36.25: S
16:35:26.878917 63.38.124.251.2745 > 10.10.8.37.25: S
```

```
16:35:26.914245 63.38.124.251.2732 > 10.10.8.30.25: S
16:35:27.107128 63.38.124.251.2739 > 10.10.8.31.25: S
16:35:27.525931 63.38.124.251.2743 > 10.10.8.36.25: S
16:35:27.525937 63.38.124.251.2742 > 10.10.8.33.25: S
16:35:27.729581 63.38.124.251.2745 > 10.10.8.37.25: S
16:35:27.903257 63.38.124.251.2739 > 10.10.8.31.25: S
16:35:28.320713 63.38.124.251.2743 > 10.10.8.36.25: S
16:35:28.328562 63.38.124.251.2742 > 10.10.8.33.25: S
16:35:28.534366 63.38.124.251.2745 > 10.10.8.37.25: S
16:35:28.966414 63.38.124.251.2760 > 10.10.8.39.25: S
16:35:29.232454 63.38.124.251.2761 > 10.10.8.40.25: S
16:35:29.734872 63.38.124.251.2760 > 10.10.8.39.25: S
16:35:30.018612 63.38.124.251.2761 > 10.10.8.40.25: S
16:35:30.507514 63.38.124.251.2760 > 10.10.8.39.25: S
16:35:30.821204 63.38.124.251.2761 > 10.10.8.40.25: S
16:35:33.693392 63.38.124.251.2787 > 10.10.8.41.25: S
16:35:34.528456 63.38.124.251.2787 > 10.10.8.41.25: S
16:35:35.335208 63.38.124.251.2787 > 10.10.8.41.25: S
16:35:38.444948 63.38.124.251.2809 > 10.10.8.42.25: S
16:35:39.214185 63.38.124.251.2809 > 10.10.8.42.25: S
16:35:40.014991 63.38.124.251.2809 > 10.10.8.42.25: S
16:52:44.505106 63.38.124.251.4095 > 10.10.8.79.25: S
16:52:46.771595 63.38.124.251.4108 > 10.10.8.81.25: S
16:52:47.545954 63.38.124.251.4108 > 10.10.8.81.25: S
16:52:48.068568 63.38.124.251.4116 > 10.10.8.83.25: S
16:52:48.358892 63.38.124.251.4108 > 10.10.8.81.25: S
16:52:48.855676 63.38.124.251.4116 > 10.10.8.83.25: S
16:52:49.659391 63.38.124.251.4116 > 10.10.8.83.25: S
16:52:50.920330 63.38.124.251.4131 > 10.10.8.84.25: S
16:52:50.994346 63.38.124.251.4133 > 10.10.8.86.25: S
16:52:51.139017 63.38.124.251.4135 > 10.10.8.87.25: S
16:52:51.571987 63.38.124.251.4139 > 10.10.8.88.25: S
16:52:51.619724 63.38.124.251.4131 > 10.10.8.84.25: S
16:52:51.727209 63.38.124.251.4133 > 10.10.8.86.25: S
16:52:51.811708 63.38.124.251.4141 > 10.10.8.89.25: S
16:52:51.817312 63.38.124.251.4135 > 10.10.8.87.25: S
16:52:52.338811 63.38.124.251.4139 > 10.10.8.88.25: S
16:52:52.339069 63.38.124.251.4131 > 10.10.8.84.25: S
16:52:52.416664 63.38.124.251.4133 > 10.10.8.86.25: S
16:52:52.531337 63.38.124.251.4135 > 10.10.8.87.25: S
16:52:52.532413 63.38.124.251.4141 > 10.10.8.89.25: S
16:52:53.117187 63.38.124.251.4139 > 10.10.8.88.25: S
16:52:53.337321 63.38.124.251.4141 > 10.10.8.89.25: S
16:52:53.679634 63.38.124.251.4150 > 10.10.8.90.25: S
16:52:54.432160 63.38.124.251.4150 > 10.10.8.90.25: S
16:52:54.813254 63.38.124.251.4157 > 10.10.8.91.25: S
16:52:55.201428 63.38.124.251.4161 > 10.10.8.92.25: S
16:52:55.243823 63.38.124.251.4150 > 10.10.8.90.25: S
16:52:55.243892 63.38.124.251.4162 > 10.10.8.93.25: S
16:52:55.527150 63.38.124.251.4157 > 10.10.8.91.25: S
16:52:55.928514 63.38.124.251.4161 > 10.10.8.92.25: S
16:52:56.023284 63.38.124.251.4162 > 10.10.8.93.25: S
16:52:56.225692 63.38.124.251.4157 > 10.10.8.91.25: S
16:52:56.614750 63.38.124.251.4161 > 10.10.8.92.25: S
16:52:56.716260 63.38.124.251.4162 > 10.10.8.93.25: S
16:52:57.717562 63.38.124.251.4176 > 10.10.8.94.25: S
16:52:58.445344 63.38.124.251.4176 > 10.10.8.94.25: S
```

```
16:52:59.141501 63.38.124.251.4176 > 10.10.8.94.25: S
16:53:05.786735 63.38.124.251.4217 > 10.10.8.95.25: S
16:53:06.460580 63.38.124.251.4221 > 10.10.8.96.25: S
16:53:06.556178 63.38.124.251.4217 > 10.10.8.95.25: S
16:53:07.251125 63.38.124.251.4221 > 10.10.8.96.25: S
16:53:07.352173 63.38.124.251.4217 > 10.10.8.95.25: S
16:53:07.963183 63.38.124.251.4221 > 10.10.8.96.25: S
16:54:59.213260 63.38.124.251.4846 > 10.10.8.155.25: S
16:54:59.999318 63.38.124.251.4846 > 10.10.8.155.25: S
16:55:00.762938 63.38.124.251.4846 > 10.10.8.155.25: S
16:55:02.149746 63.38.124.251.4865 > 10.10.8.156.25: S
16:55:02.862412 63.38.124.251.4865 > 10.10.8.156.25: S
16:55:03.654282 63.38.124.251.4865 > 10.10.8.156.25: S
16:55:03.895258 63.38.124.251.4877 > 10.10.8.157.25: S
16:55:04.687501 63.38.124.251.4877 > 10.10.8.157.25: S
16:55:05.461583 63.38.124.251.4877 > 10.10.8.157.25: S
17:36:29.878006 63.38.124.251.2062 > 10.10.8.143.25: S
17:36:30.533634 63.38.124.251.2066 > 10.10.8.144.25: S
17:36:30.591287 63.38.124.251.2062 > 10.10.8.143.25: S
17:36:31.182383 63.38.124.251.2066 > 10.10.8.144.25: S
17:36:31.256171 63.38.124.251.2072 > 10.10.8.146.25: S
17:36:31.289051 63.38.124.251.2062 > 10.10.8.143.25: S
17:36:31.546820 63.38.124.251.2075 > 10.10.8.147.25: S
17:36:31.752077 63.38.124.251.2077 > 10.10.8.148.25: S
17:36:31.894498 63.38.124.251.2072 > 10.10.8.146.25: S
17:36:31.901536 63.38.124.251.2066 > 10.10.8.144.25: S
17:36:32.188918 63.38.124.251.2075 > 10.10.8.147.25: S
17:36:32.401036 63.38.124.251.2077 > 10.10.8.148.25: S
17:36:32.606623 63.38.124.251.2072 > 10.10.8.146.25: S
17:36:32.900577 63.38.124.251.2075 > 10.10.8.147.25: S
17:36:33.079792 63.38.124.251.2077 > 10.10.8.148.25: S
17:36:34.558785 63.38.124.251.2100 > 10.10.8.149.25: S
17:36:35.196947 63.38.124.251.2100 > 10.10.8.149.25: S
17:36:35.909054 63.38.124.251.2100 > 10.10.8.149.25: S
17:36:37.427303 63.38.124.251.2116 > 10.10.8.150.25: S
17:36:38.123902 63.38.124.251.2116 > 10.10.8.150.25: S
17:36:38.179673 63.38.124.251.2127 > 10.10.8.151.25: S
17:36:38.822733 63.38.124.251.2127 > 10.10.8.151.25: S
17:36:38.822737 63.38.124.251.2116 > 10.10.8.150.25: S
17:36:39.918382 63.38.124.251.2127 > 10.10.8.151.25: S
19:08:23.818857 63.38.124.251.2280 > 10.10.8.1.25: S
19:08:24.522078 63.38.124.251.2280 > 10.10.8.1.25: S
19:08:25.215431 63.38.124.251.2280 > 10.10.8.1.25: S
19:41:57.039697 63.38.124.251.4892 > 10.10.8.13.25: S
19:41:57.256656 63.38.124.251.4894 > 10.10.8.16.25: S
19:41:57.718165 63.38.124.251.4892 > 10.10.8.13.25: S
19:41:57.943291 63.38.124.251.4894 > 10.10.8.16.25: S
19:41:58.445614 63.38.124.251.4892 > 10.10.8.13.25: S
19:41:58.648357 63.38.124.251.4894 > 10.10.8.16.25: S
19:42:04.822932 63.38.124.251.4942 > 10.10.8.17.25: S
19:42:04.823581 10.10.8.17.25 > 63.38.124.251.4942: SA
19:46:07.608044 63.38.124.251.2218 > 10.10.8.43.25: S
19:46:08.318784 63.38.124.251.2218 > 10.10.8.43.25: S
19:46:09.038705 63.38.124.251.2218 > 10.10.8.43.25: S
19:46:29.353452 63.38.124.251.2295 > 10.10.8.45.25: S
19:46:29.399515 63.38.124.251.2296 > 10.10.8.46.25: S
19:46:29.788450 63.38.124.251.2302 > 10.10.8.47.25: S
```

```
19:46:30.086437 63.38.124.251.2296 > 10.10.8.46.25: S
19:46:30.086441 63.38.124.251.2295 > 10.10.8.45.25: S
19:46:30.599587 63.38.124.251.2302 > 10.10.8.47.25: S
19:46:30.767398 63.38.124.251.2295 > 10.10.8.45.25: S
19:46:30.773251 63.38.124.251.2296 > 10.10.8.46.25: S
19:46:31.339661 63.38.124.251.2302 > 10.10.8.47.25: S
19:46:36.183414 63.38.124.251.2332 > 10.10.8.48.25: S
19:46:36.183728 10.10.8.48.25 > 63.38.124.251.2332: SA
19:49:40.973348 63.38.124.251.3080 > 10.10.8.59.25: S
19:49:41.690937 63.38.124.251.3080 > 10.10.8.59.25: S
19:49:42.393056 63.38.124.251.3080 > 10.10.8.59.25: S
19:49:44.099772 63.38.124.251.3099 > 10.10.8.60.25: S
19:49:44.772584 63.38.124.251.3099 > 10.10.8.60.25: S
19:49:45.513760 63.38.124.251.3099 > 10.10.8.60.25: S
19:49:51.561741 63.38.124.251.3132 > 10.10.8.62.25: S
19:49:52.305425 63.38.124.251.3132 > 10.10.8.62.25: S
19:49:53.023899 63.38.124.251.3132 > 10.10.8.62.25: S
19:49:55.854073 63.38.124.251.3152 > 10.10.8.18.25: S
19:49:56.600195 63.38.124.251.3152 > 10.10.8.18.25: S
19:49:57.311607 63.38.124.251.3152 > 10.10.8.18.25: S
19:49:59.432578 63.38.124.251.3170 > 10.10.8.19.25: S
19:50:00.120363 63.38.124.251.3170 > 10.10.8.19.25: S
19:50:00.803371 63.38.124.251.3170 > 10.10.8.19.25: S
19:50:02.212263 63.38.124.251.3185 > 10.10.8.20.25: S
19:50:02.213407 10.10.8.20.25 > 63.38.124.251.3185: SA
19:50:04.326299 63.38.124.251.3195 > 10.10.8.21.25: S
19:50:04.998165 63.38.124.251.3195 > 10.10.8.21.25: S
19:50:05.680725 63.38.124.251.3195 > 10.10.8.21.25: S
19:50:06.242377 63.38.124.251.3203 > 10.10.8.22.25: S
19:50:06.710803 63.38.124.251.3204 > 10.10.8.23.25: S
19:50:06.903685 63.38.124.251.3203 > 10.10.8.22.25: S
19:50:07.394698 63.38.124.251.3207 > 10.10.8.24.25: S
19:50:07.417932 63.38.124.251.3204 > 10.10.8.23.25: S
19:50:07.611131 63.38.124.251.3203 > 10.10.8.22.25: S
19:50:08.131819 63.38.124.251.3207 > 10.10.8.24.25: S
19:50:08.139872 63.38.124.251.3204 > 10.10.8.23.25: S
19:50:08.808806 63.38.124.251.3207 > 10.10.8.24.25: S
19:50:12.385951 63.38.124.251.3233 > 10.10.8.25.25: S
19:50:13.124051 63.38.124.251.3233 > 10.10.8.25.25: S
19:50:13.818205 63.38.124.251.3233 > 10.10.8.25.25: S
19:50:25.342365 63.38.124.251.3276 > 10.10.8.26.25: S
19:50:26.018825 63.38.124.251.3276 > 10.10.8.26.25: S
19:50:26.722198 63.38.124.251.3276 > 10.10.8.26.25: S
19:50:33.198339 63.38.124.251.3317 > 10.10.8.63.25: S
19:50:33.841537 63.38.124.251.3317 > 10.10.8.63.25: S
19:50:34.542317 63.38.124.251.3317 > 10.10.8.63.25: S
19:50:36.364406 63.38.124.251.3333 > 10.10.8.64.25: S
19:50:37.051066 63.38.124.251.3333 > 10.10.8.64.25: S
19:50:37.765563 63.38.124.251.3333 > 10.10.8.64.25: S
19:51:02.646131 63.38.124.251.3466 > 10.10.8.65.25: S
19:51:03.322206 63.38.124.251.3466 > 10.10.8.65.25: S
19:51:03.989397 63.38.124.251.3466 > 10.10.8.65.25: S
19:54:51.161512 63.38.124.251.4537 > 10.10.8.66.25: S
19:54:51.856610 63.38.124.251.4537 > 10.10.8.66.25: S
19:54:52.567594 63.38.124.251.4537 > 10.10.8.66.25: S
19:55:00.676251 63.38.124.251.4588 > 10.10.8.67.25: S
19:55:01.368188 63.38.124.251.4588 > 10.10.8.67.25: S
```

```
19:55:02.067258 63.38.124.251.4588 > 10.10.8.67.25: S
20:28:29.225966 63.38.124.251.2692 > 10.10.8.75.25: S
20:28:29.958051 63.38.124.251.2692 > 10.10.8.75.25: S
20:28:30.635996 63.38.124.251.2692 > 10.10.8.75.25: S
20:28:36.419700 63.38.124.251.2734 > 10.10.8.76.25: S
20:28:37.049630 63.38.124.251.2734 > 10.10.8.76.25: S
20:28:37.762701 63.38.124.251.2734 > 10.10.8.76.25: S
20:28:54.238015 63.38.124.251.2829 > 10.10.8.77.25: S
20:28:54.891411 63.38.124.251.2829 > 10.10.8.77.25: S
20:28:55.604481 63.38.124.251.2829 > 10.10.8.77.25: S
20:29:02.954186 63.38.124.251.2869 > 10.10.8.78.25: S
20:29:03.612168 63.38.124.251.2869 > 10.10.8.78.25: S
20:29:04.310585 63.38.124.251.2869 > 10.10.8.78.25: S

-----------------------------------------------------------------
TIME:   19:46:36.183414
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=48
 TCP:   port 2332 -> smtp seq=4242414174 ack=0000000000
        hlen=28 (data=0) UAPRSF=000010 wnd=8760 cksum=34A3 urg=
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:36.183728 (0.000314)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=48
 TCP:   port smtp -> 2332 seq=2770844848 ack=4242414175
        hlen=28 (data=0) UAPRSF=010010 wnd=17520 cksum=B081 urg
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:36.406081 (0.222353)
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=40
 TCP:   port 2332 -> smtp seq=4242414175 ack=2770844849
        hlen=20 (data=0) UAPRSF=010000 wnd=8760 cksum=FF7D urg=
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:36.696204 (0.290123)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=136
 TCP:   port smtp -> 2332 seq=2770844849 ack=4242414175
        hlen=20 (data=96) UAPRSF=011000 wnd=17520 cksum=550A ur
DATA:   220 my-workstation.my.net ESMTP Sendmail 8.10.1/8.10.1;
        , 17 Jan 2001 19:46:36 -0700 (MST).
-----------------------------------------------------------------
TIME:   19:46:36.952617 (0.256413)
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=61
 TCP:   port 2332 -> smtp seq=4242414175 ack=2770844945
        hlen=20 (data=21) UAPRSF=011000 wnd=8664 cksum=40FB urg
DATA:   HELO partysales.org.
-----------------------------------------------------------------
TIME:   19:46:36.952843 (0.000226)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=40
 TCP:   port smtp -> 2332 seq=2770844945 ack=4242414196
        hlen=20 (data=0) UAPRSF=010000 wnd=17499 cksum=DCE5 urg
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:36.953133 (0.000290)
 TCP:   port smtp -> 2332 seq=2770844945 ack=4242414196
        hlen=20 (data=101) UAPRSF=011000 wnd=17520 cksum=AECF u
DATA:   250 my-workstation.my.net Hello 1Cust251.tnt29.det3.da.
        et [63.38.124.251], pleased to meet you.
```

```
--------------------------------------------------------------
TIME:   19:46:37.245562 (0.292429)
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=80
 TCP:   port 2332 -> smtp seq=4242414196 ack=2770845046
        hlen=20 (data=40) UAPRSF=011000 wnd=8563 cksum=EC4B urg
DATA:   MAIL From: <bndmeovrli@partysales.org>.
--------------------------------------------------------------
TIME:   19:46:37.245789 (0.000227)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=40
 TCP:   port smtp -> 2332 seq=2770845046 ack=4242414236
        hlen=20 (data=0) UAPRSF=010000 wnd=17480 cksum=DC6B urg
DATA:   <No data>
--------------------------------------------------------------
TIME:   19:46:37.496893 (0.251104)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=92
 TCP:   port smtp -> 2332 seq=2770845046 ack=4242414236
        hlen=20 (data=52) UAPRSF=011000 wnd=17520 cksum=79CD ur
DATA:   250 2.1.0 <bndmeovrli@partysales.org>... Sender ok.

--------------------------------------------------------------
TIME:   19:46:37.767933 (0.271040)
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=75
 TCP:   port 2332 -> smtp seq=4242414236 ack=2770845098
        hlen=20 (data=35) UAPRSF=011000 wnd=8511 cksum=4C81 urg
DATA:   RCPT To:<tombanks1944@fnmail.com>.
--------------------------------------------------------------
TIME:   19:46:37.768161 (0.000228)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=40
 TCP:   port smtp -> 2332 seq=2770845098 ack=4242414271
        hlen=20 (data=0) UAPRSF=010000 wnd=17485 cksum=DC0F urg
DATA:   <No data>
--------------------------------------------------------------
TIME:   19:46:37.773536 (0.005375)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=96
 TCP:   port smtp -> 2332 seq=2770845098 ack=4242414271
        hlen=20 (data=56) UAPRSF=011000 wnd=17520 cksum=3719 ur
DATA:   550 5.7.1 <tombanks1944@fnmail.com>... Relaying denied.
--------------------------------------------------------------
TIME:   19:46:38.018447 (0.244911)
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=69
 TCP:   port 2332 -> smtp seq=4242414271 ack=2770845154
        hlen=20 (data=29) UAPRSF=011000 wnd=8455 cksum=C065 urg
DATA:   RCPT To:<retro0823@aol.com>.
--------------------------------------------------------------
TIME:   19:46:38.018672 (0.000225)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=40
 TCP:   port smtp -> 2332 seq=2770845154 ack=4242414300
        hlen=20 (data=0) UAPRSF=010000 wnd=17491 cksum=DBB4 urg
DATA:   <No data>
--------------------------------------------------------------
TIME:   19:46:38.024758 (0.006086)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=90
 TCP:   port smtp -> 2332 seq=2770845154 ack=4242414300
        hlen=20 (data=50) UAPRSF=011000 wnd=17520 cksum=AACB ur
DATA:   550 5.7.1 <retro0823@aol.com>... Relaying denied.
--------------------------------------------------------------
TIME:   19:46:38.277921 (0.253163)
```

```
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=46
 TCP:   port 2332 -> smtp seq=4242414300 ack=2770845204
        hlen=20 (data=6) UAPRSF=011000 wnd=8405 cksum=5A41 urg=
DATA:   RSET.
-----------------------------------------------------------------
TIME:   19:46:38.278156 (0.000235)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=40
 TCP:   port smtp -> 2332 seq=2770845204 ack=4242414306
        hlen=20 (data=0) UAPRSF=010000 wnd=17514 cksum=DB65 urg
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:38.278273 (0.000117)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=63
 TCP:   port smtp -> 2332 seq=2770845204 ack=4242414306
        hlen=20 (data=23) UAPRSF=011000 wnd=17520 cksum=688D ur
DATA:   250 2.0.0 Reset state.
-----------------------------------------------------------------
TIME:   19:46:38.527944 (0.249671)
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=45
 TCP:   port 2332 -> smtp seq=4242414306 ack=2770845227
        hlen=20 (data=5) UAPRSF=011000 wnd=8382 cksum=5A44 urg=
DATA:   QUIT
-----------------------------------------------------------------
TIME:   19:46:38.527961 (0.000017)
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=40
 TCP:   port 2332 -> smtp seq=4242414311 ack=2770845227
        hlen=20 (data=0) UAPRSF=010001 wnd=8382 cksum=FEF4 urg=
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:38.528165 (0.000204)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=40
 TCP:   port smtp -> 2332 seq=2770845227 ack=4242414311
        hlen=20 (data=0) UAPRSF=010000 wnd=17515 cksum=DB48 urg
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:38.528185 (0.000020)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=40
 TCP:   port smtp -> 2332 seq=2770845227 ack=4242414312
        hlen=20 (data=0) UAPRSF=010000 wnd=17515 cksum=DB47 urg
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:38.528384 (0.000199)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=93
 TCP:   port smtp -> 2332 seq=2770845227 ack=4242414312
        hlen=20 (data=53) UAPRSF=011000 wnd=17520 cksum=EE56 ur
DATA:   221 2.0.0 my-workstation.my.net closing connection.
-----------------------------------------------------------------
TIME:   19:46:38.528537 (0.000153)
  IP:   10.10.8.48 -> 63.38.124.251 hlen=20 TOS=00 dgramlen=40
 TCP:   port smtp -> 2332 seq=2770845280 ack=4242414312
        hlen=20 (data=0) UAPRSF=010001 wnd=17520 cksum=DB0C urg
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:38.800740 (0.272203)
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=40
 TCP:   port 2332 -> smtp seq=4242414312 ack=0000000000
        hlen=20 (data=0) UAPRSF=000100 wnd=0 cksum=8312 urg=0
```

```
DATA:   <No data>
-----------------------------------------------------------------
TIME:   19:46:38.806471 (0.005731)
  IP:   63.38.124.251 -> 10.10.8.48 hlen=20 TOS=00 dgramlen=40
 TCP:   port 2332 -> smtp seq=4242414312 ack=4242414312
        hlen=20 (data=0) UAPRSF=000100 wnd=0 cksum=6B4B urg=0
DATA:   <No data>
-----------------------------------------------------------------
```

1 - Source of Trace

Production network at a Canadian university.

2 - Detect Generated By ...

Snort 1.7 logging to a MySQL 3.23.28 database for display by ACID 0.9.5b9. The scan list is a slightly reformatted output from tcpdump, while the more verbose mail session is from tcpshow. The rule that generated this alert was: `alert tcp $HOME_NET 25 -> $EXTERNAL_NET any (msg:"IDS249 - SMTP Relaying Denied" ; flags:AP; content: "5.7.1"; depth:70;)`

3 - Probability of source address spoofing

Essentially none. We see the 3-way handshake complete and then data being exhanged. Except for the fact that this is a filthy spammer, this transaction is about as legitimate as they come.

4 - Description of attack

A connection is made to a mailserver and false source and destination addresses are entered in hope that the mailserver will deliver the message.

5 - Attack Mechanism

Some mailservers are incorrectly configured or are quite old, and will relay mail for anyone. The scum of the earth, who are in the business of carpetbombing the internet with bulk unsolicited email (hereinafter referred to as spammers) search for these vulnerable servers to disguise their identities. Once a server has been located, a message is loaded and sent to thousands if not millions of net users who probably don't want to read the message and who would probably enjoy seeing the spammers run over by a steamroller. The fact that a relay box is used means that it is non-trivial to determine the origin of the spam, and it also passes the bandwidth costs for sending the spam off to the owner of the relay box.

6 - Correlations

See www.mail-abuse.orgfor way more information about unsolicited commercial email and the problems it causes. Mail-abuse.org also offers some suggestions for minimizing the amount of spam your network might be bombarded with.

7 - Evidence of Active Targeting

This attack is in two parts; the first being the scan for SMTP servers, and the second being the actual attempt to relay off a server. The scan is a general one, aimed at my network. The relay was targetted specifically at my workstation since it has a listening SMTP daemon. Notice the speed of and the gaps in the scan. This would indicate that the attacker is probably interleaving probes in hopes of not getting caught too quickly. The rate of the packets indicates that this is probably a dialup connection, and in fact this is one of UUNET's famous dialup modems. UUNET seems to be a favourite place to buy throwaway accounts, as there are are access points everywhere, and broadband connections are still too expensive and hard to obtain, at least in that they generally require some form of identification which is about the last thing in the world that spammers want to give out.

8 - Severity

Criticality: At the very worst, if this attack had succeeded, this attack would have cause a mild denial of service on our mailservers and sysadmins who would spend the next 6 hours digging out from 10000 complaints like we did the last time we got relayed off of. DoS-ing one out of five mailhandlers isn't bad, DoS-ing two out of three admins is; robots like us are hard to come by. C=4

Lethality: At worst, this attack would cause us a great deal of embarrasment and wasted time. Cleanup is a matter of 'rm -f /var/spool/mqueue/* & ; cat /dev/null > /var/mail/root' . L=3

Host Countermeasures: The mailservers are running very-well-maintained versions of Exim. The configuration files have been carefully written so as to prevent relaying and the mail daemon checks incoming messages against RSS, DUL and the RBL. This message was denied before the 'MAIL' command. H=5.

Net Countermeasures: We have a good firewall which only allows SMTP access to our official SMTP servers, and all traffic must pass through this firewall. Firewall has not had a thorough audit, but it seems to be doing the job. N=4

(criticality+lethality)-(host+net)=severity
(4 + 3) - (5 + 4) = -2

Negative severity, which is a good thing. We were never at any risk from this attack, having learned our lessons the hard way.

9 - Defensive Recommendations

The defenses are appropriate. The firewall enforced the site server policy, the mail server enforced the site email policy, and the logging system captured enough data to allow the connection and session to be constructed. The only thing left to do is ensure the analysts have enough coffee handy.

10 - Test Question

Based on the system description above, the most effective way to deal with spam is to:

a)nmap and crash the spammer's network,
b)drag spammers and all users who've ever replied to a "remove" list out to the woods for a lynching,
c)keep your smtp server well patched,
d)subscribe to DUL/RSS/RBL.

Answer: D. While A and B might be fun, they're probably illegal in your part of the world. Keeping patches current is always good, but a well patched daemon is no good if it's enforcing a lousy security policy. The only remaining choice is to have your mailserver check a database of domains (like dialup modem pools or known spamhauses) before accepting messages.

1.3 **t0rn (alias "synscan") scans**

0 - Packet Traces

```
---------------------------------------------------------------
Packet 1
TIME:   05:54:46.696805
LINK:   00:02:B3:07:EE:ED -> 08:00:20:75:03:AA type=IP
  IP:   210.145.55.34 -> 10.10.8.14 hlen=20 TOS=00 dgramlen=40
        MF/DF=0/0 frag=0 TTL=10 proto=TCP cksum=BD8B
 TCP:   port ftp -> ftp seq=1458705109 ack=0898797946
        hlen=20 (data=0) UAPRSF=000011 wnd=1028 cksum=219D urg=
DATA:   <No data>
---------------------------------------------------------------
Packet 2
TIME:   05:54:47.197990 (0.501185)
LINK:   00:02:B3:07:EE:ED -> 00:00:C0:B5:FB:B2 type=IP
  IP:   210.145.55.34 -> 10.10.8.39 hlen=20 TOS=00 dgramlen=40
        MF/DF=0/0 frag=0 TTL=10 proto=TCP cksum=BD72
 TCP:   port ftp -> ftp seq=0082899021 ack=0718383587
        hlen=20 (data=0) UAPRSF=000011 wnd=1028 cksum=8865 urg=
DATA:   <No data>
---------------------------------------------------------------
```

1 - Source of Trace

Production network at a Canadian university.

2 - Detect Generated By ...

Snort 1.7b8, logging to a MySQL 3.23.28 database for display by ACID 0.95b9. Backup analysis capability provided by tcpdump 3.5.2

and tcpshow 1.7.4. I use ACID to find the interesting packets and tcpshow to pretty-print the packet for inclusion into reports. I'm using the January 7, 2001 version of "snortfull.conf" from www.snort.org. The rule which matched this packet was:

```
alert tcp $EXTERNAL_NET any ->
$HOME_NET any (msg:"SCAN-SYN
FIN";flags:SF;)
```

3 - Probability of source address spoofing

Somewhat unlikely. This is a packet from the middle of a tcp stream, which means that the 3-way handshake completed. Granted, a compromised router could be used to inject packets into the stream, but it is nearly certain that the source host is who the packet says it is. Furthermore, the stream continued after this packet, indicating that this packet is legitimate, and the host for which it was destined accepted this packet without much fuss. Finally, this is a recon probe, and the attacker will put the real address of the scanning machine on the outgoing packets so that he gets answers back.

4 - Description of attack

An automated tool is used to scan netblocks for a particular vulnerability and then exploit them. The packets we see here are the recon probes. The actual attack comes later, and bears a striking resemblance to the false alarm buffer overflow.

5 - Attack Mechanism

Once upon a time a tcp segment with both SYN and FIN set was stealthy; that is, it stood a good chance of slipping past firewalls and IDS boxes. This is packet crafting, since IP stacks will generally not try to start and end a connection in the same packet, and the IP ID is supposed to increment. This is not the case with this particular scanner. The scan tries to see if there is a listening FTP server without establishing a connection, thereby avoiding being logged. The actual attack that these packets contain is an attempt to enumerate our FTP servers.

6 - Correlations

There are several "interesting" things to correlate here: The SYN-FIN pair, source and destination port are both tcp/21, and the IP ID is 0x9a02 (39426 decimal). William Stearns mentioned this particular scan in his GCIA practical, but did not mention the significance of the IP ID. Teri Bidwell references this signature, and refers us to www.sans.org/y2k/102800.htm. Teri also has an article at www.sans.org/y2k/111600.htm in which it is asserted that this is a slightly modified "idlescan". Furthermore packets matching this signature are presented in the "Signature Analysis" course notes. It is worth noting that this signature was already getting common in March 2000. As it happens, one tool that will generate this signature is from the t0rn root kit - "t0rnscan". There is an analysis of the t0rn kit at www.sans.org/y2k/t0rn.htm. CERT issued Incident Note IN-2000-10 about this package. Other discussions may be found by searching the "incidents" list at www.insecure.org, specifcally http://lists.insecure.org/incidents/2000/Nov/0175.html

In this case, the attacker was specifically looking for wu-ftpd servers. I can be certain of this because the attacking machine also had wu-ftpd exploits, and several long lists of domains with vulnerable wu-ftpd servers. Furthermore, the scan came from a machine with a vulnerable wu-ftpd server; when contacted, the site administrator was kind enough to give me copies of the rootkit.

7 - Evidence of Active Targeting

This is definately intentional traffic, aimed at my network. It's not one particular host, but scanning a network is clearly active targetting; probes are being sent to the target network, answers are expected.

8 - Severity

Criticality: We don't rely much on FTP servers, it wouldn't be the end of the world to lose one. Unfortunately one of these server is also an email exchanger. C=4

Lethality: This was just recon, the attacker might only get a small bit of network information (FTP daemon version), which we hand out to anyone with an FTP client. Not damaging in itself, but not harmless. L=2

Host Countermeasures: Neither of these 2 machines run wu-ftpd. The FTP daemons are current. TCP wrappers are in place, the daemons are running as "nobody" and serve anonymous-only. Hardened OS with patches. H=5

Net Countermeasures: We have a good firewall which only allows FTP access to our official FTP servers, and all traffic must pass through this firewall. We also reject "non-standard" packets, such as things with reserved bits SYN-FIN set. Logging works well. Firewall has been configured to default deny, but it has not had a thorough audit. It seems to be doing the job, so I'll give it most of the points. N=4

(criticality+lethality)-(host+net)=severity
(4 + 2) - (5 + 4) = -3

This isn't really cause for concern, for now I have the warm fuzzy secure feeling.

9 - Defensive Recommendations

Probably the easiest way to block this sort of nonsense (and a whole bunch of other unwanted traffic) is having a default-deny firewall. Failing that, a bridge that drops illegal flags would be a good start. There are certain protocols that I think should never leave the LAN, including SMB and Portmap (by no means an exhaustive list). If a network-firewall is not permitted (yes, I have seen and heard that, and it wasn't just a nightmare) adding host-based access controls like tcpwrappers would be in order.

10 - Test Question

```
4500 0028 9a02 0000 2a06 579d 0a0a 082e
0a0a 0801 006f 006f 4b41 2c12 3419 6f64
5003 0404 f1b1 0000
```

What's most wrong with this packet?
a)tcp[13:1] & 0x3 = 0x03
b)ip[4:2] = 0x9a02
c)tcp[0:2] = tcp[2:2]
d)tcp[2:2] = 0x006f

Answer: a. This question tests 2 things: familiarity with tcpdump filters and familiarity with tcp/ip headers. In this case both the SYN and FIN flags are set. 'tcp[13:1]' is the tcp flags byte, '& 0x3' says that we are interested in the low 2 bits, and '= 0x03' means that we want to see if those bits are both set. Yes, the checksum is off since I deleted some of my network info. As for the other tests, those may seem a bit threatening based on your knowledge of certain attack tools, but they're by no means as odd as invalid flags.

1.4 **Obvious(?) Host Counting**

0 - Packet Traces

```
---------------------------------------------------------------
Packet 1
TIME:   16:14:12.530970
LINK:   00:E0:8F:0A:F0:00 -> 00:02:B3:07:EE:ED type=IP
  IP:   151.15.109.92 -> 10.10.8.0 hlen=20 TOS=00 dgramlen=47 i
        MF/DF=0/0 frag=0 TTL=112 proto=ICMP cksum=22CF
ICMP:   echo-request cksum=206C
DATA:   .....JC.135333137.pt ..
---------------------------------------------------------------
Packet 2
TIME:   16:14:12.536366 (0.005396)
LINK:   00:E0:8F:0A:F0:00 -> 00:02:B3:07:EE:ED type=IP
  IP:   151.15.109.92 -> 10.10.8.255 hlen=20 TOS=00 dgramlen=47
        MF/DF=0/0 frag=0 TTL=112 proto=ICMP cksum=20D0
ICMP:   echo-request cksum=2865
DATA:   .....JC.-32439037.pt ..
---------------------------------------------------------------
Packet 3
TIME:   16:14:12.540765 (0.004399)
```

```
LINK:   00:E0:8F:0A:F0:00 -> 00:02:B3:07:EE:ED type=IP
  IP:   151.15.109.92 -> 10.10.9.0 hlen=20 TOS=00 dgramlen=47 i
        MF/DF=0/0 frag=0 TTL=112 proto=ICMP cksum=1FCF
ICMP:   echo-request cksum=185B
DATA:   .....JC.135988497.pt ..
-----------------------------------------------------------------
Packet 4
TIME:   16:14:12.548946 (0.008181)
LINK:   00:E0:8F:0A:F0:00 -> 00:02:B3:07:EE:ED type=IP
  IP:   151.15.109.92 -> 10.10.9.255 hlen=20 TOS=00 dgramlen=47
        MF/DF=0/0 frag=0 TTL=112 proto=ICMP cksum=1DD0
ICMP:   echo-request cksum=1E64
DATA:   .....JC.-31783677.pt ..
-----------------------------------------------------------------
```

1 - Source of Trace

Production network at a Canadian university.

2 - Detect Generated By ...

Snort 1.7b8, logging to a MySQL 3.23.28 database for display by ACID 0.95b9. Backup analysis capability provided by tcpdump 3.5.2 and tcpshow 1.7.4. I use ACID to find the interesting packets and tcpshow to pretty-print the packet for inclusion into reports. I'm using the January 7, 2001 version of "snortfull.conf" from www.snort.org. The rule which matched this packet was:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"ICMP-PING Echo Request"; itype: 8; )
```

3 - Probability of source address spoofing

Non-zero, possibly as high as 50%. I say this for several reasons. First, the pings were being sent to my network and broadcast addresses, which suggests that this may be some sort of smurf attempt. Or at least looking for smurf amplifiers. Or maybe it's an attempt to map my network. This would suggest that there is someone 'networkly' close to 151.15.109.92 sending these packets, (possibly even that host) and listening for responses. Most machines don't get too upset if an unsolicited ping response comes in which means that someone could be spoofing without fear of too many people noticing the spoof. Still, the fact that pings we sent to both the broadcast and network addresses suggests that this is reconnaisance and there is someone expecting answers. If that's the case, we must be dealing with someone who has a rather well equipped sniffer machine in a very good place on the network, since these packets are coming in a mere 5ms apart. This makes it sound like the attacker is a lot closer to us than what the packets would indicate - 5ms to generate and send a ping, wait for all the hosts to reply, and then send another. Some days I can't get off the LAN in under 5ms. That suggests to me that the listener is not on the same box as the generator, and might be a compromised router instead.

4 - Description of attack

Pings are being sent to my network and broadcast addresses, probably in the hope of getting a bunch of answers back. This would tell the attacker the IP addresses of all my currently active hosts.

5 - Attack Mechanism

When the router sees a packet bound for 10.10.8.0 or 10.10.9.255, it will rewrite the destination to be 255.255.255.255, and send it to hardware address ff:ff:ff:ff:ff:ff. This will cause all ethernet cards and all IP stacks to see and process the packet. If it's something trivial like a UDP or ICMP echo, then each host will send back a reply with its real IP address. Poof! One packet turns into a whole bunch of shiny new targets.

6 - Correlations

I started to explain these packets to a friend who looked at me and claimed I was playing games with numerology. I won't appeal to the horoscope or tarot here, though those may yet turn to be valuable debugging tools. Looking at a bunch of other probes, I noticed that often the payload can help when trying to determine what sort of attack something is. 3dns and other load-balancers often seem to put the IP address of whatever they're probing in the payload. This was not the case here: none of the bytes making up my network addresses were found in the payload. Further examination of the payload found some things which as yet I cannot explain, but look like they happened for a reason. Of course half a dozen alarms went off in my head when I saw "313337" in the payload, my guess there is that I got the winning output from a random number generator.

There were only 4 packets I saw, addressed to 10.10.{8,9}.{0,255}. After rolling them around for a bit I noticed a few interesting things. The IP ID incremented by 0x100 (256) for each packet sent. Ping 0xd412 went to 10.10.8.0, 0xd512 went to 10.10.8.255, 0xd612 went to 10.10.9.0 and 0xd712 went to 10.10.9.255. There was a correlation of sort between packets sent to the X network versus those sent to the Y network. There was a second correlation between packets sent to the net address (.0) versus those sent to the broadcast (.255). Finally, all 4 of these probes had sort of a prologue and an epilogue. It appeared that there were 8 bytes of data available, which is just the size of 2 IP addresses. And these data bytes changed based on the destination. Below are color-enhanced dumps of the payloads. Looking at the colorings, it appears that 16 bits depend on the destination, and 48 bits are just random. Maybe they are some sort of hash or a sequence number. "Is puzzlement."

```
10.10.8.000 024A4308 31333533 33333133 37007074 2000(
10.10.8.255 024A4308 2D333234 33393033 37007074 2000(
10.10.9.000 024A4308 31333539 38383439 37007074 2000(
10.10.9.255 024A4308 2D333137 38333637 37007074 2000(
```

7 - Evidence of Active Targeting

This could be either a smurf attack or a network scan. If it is a smurf attack, then it is certainly targetted at 151.15.109.92 (or possibly this host's ISP). If it is network mapping then it's still targetted but not at one particular host - 4 probes were sent to me, which blows away the "wrong number" theory. Based on the payload of these probes, I think this is more recon than a smurf attack.

8 - Severity

Criticality: No particular host was contacted, this was sent to my network. I'm assigning this a middle priority, since the attacker could find unimportant things like Mac Powerbooks (which are hardly ever there) to critical machines like the authentication server or the coffee maker. C=3

Lethality: This is not really damaging, but getting your network mapped is a good sign of impending doom. That said it's not as bad as a wrong number, but it certainly is worse than using an ineffective canned script. It shows that the attacker is at least making the effort of staging a good break-in. L=2

Host Countermeasures: We do have a few machines which will still respond to a ping to the broadcast address. For the most part we've turned it off or added host-based packet filters wherever possible, and this seems to not break any useful protocols. C=3

Net Countermeasures: We have a good firewall which only allows FTP access to our official FTP servers, and all traffic must pass through this firewall. We also reject "silly" packets, such as attempts to ping our broadcast addresses. In fact, the firewall has been tuned to make recon sweeps difficult, useless, or at least time-consuming. Logging works well. Firewall has not had a thorough audit, but it seems to be doing the job. N=4

(criticality+lethality)-(host+net)=severity
(3 + 2) - (3 + 4) = -2

This isn't really cause for concern, for now I have the warm fuzzy secure feeling. I suppose our attacker now knows that we have a firewall, but that's one piece of information I don't mind sharing. It's much like the "We Have A Burglar Alarm" stickers: it suggests that maybe the kiddies should go play elsewhere.

9 - Defensive Recommendations

Nothing extra at this time. All systems performed as they should. The firewall and router have been carefully set up to prevent this sort of attack from ever succeeding by both blocking directed broadcasts and blocking echo requests and replies.

10 - Test Question

The simplest way to prevent becoming a smurf amplifier is:
a) name your router either 'gargamel' or 'azrael',
b) set your router to not forward directed broadcasts,
c) set your firewall to not forward echo style protocols

answer: B. On many routers this is a single command which may have been preset for you. Under OpenBSD the command to do this would be 'sysctl -w net.inet.ip.directed-broadcast = 0' which is the default (no directed broadcasts).

2.0 Analyze This

2.1 **Introduction**

This section concerns the analysis and subsequent report of detects found in the 60 or so megabytes of data provided. Since time is scarce, I wrote a number of tools which I used to give me a first-layer order to the data. From there I can dig out the offending packets to back up my analysis if necessary.

2.2 **Statistics**

Below are a few statistics gathered from the logs provided. Besides being interesting for their own sake, these figures serve as a sort of consistency check for my analysis process and tools, as well as a series of pointers to areas that need the most attention.

2.2.1 Number of Porscans seen

24	XMAS
25	SPAU
30	NMAPID
47	FULLXMAS
213	UNKNOWN
235	NULL
425	FIN
465	VECNA
562	NOACK
777	INVALIDACK
11977	UDP
50552	SYNFIN
210926	SYN
276258	**Total**

2.2.2 Top 20 Portscan Targets

Number	**Destination**
204	MY.NET.70.121
229	MY.NET.218.50
230	MY.NET.209.242
284	MY.NET.162.36
296	MY.NET.211.254

302	MY.NET.207.58
329	MY.NET.253.114
334	MY.NET.98.168
341	MY.NET.5.29
349	MY.NET.205.38
356	MY.NET.201.18
360	MY.NET.205.214
518	MY.NET.206.106
596	MY.NET.220.2
670	MY.NET.206.94
695	MY.NET.60.16
921	MY.NET.212.114
970	MY.NET.205.246
991	MY.NET.204.218
1115	MY.NET.215.210

2.2.3 Top 20 Scan Sources

Number	**Source**
4836	199.239.94.98
4855	146.101.147.251
4881	208.214.247.60
4956	209.92.40.32
5496	63.248.55.245
6033	63.198.207.51
6093	216.191.162.145
6400	62.155.244.68
6634	208.61.4.207
6919	211.49.165.9
7001	128.211.237.11
7192	160.78.49.191
8635	64.50.161.162
8763	24.23.151.112
8939	62.96.169.86
9641	62.157.23.237
11717	63.88.175.201

13057	62.252.21.241
20649	66.9.27.254

2.2.4 Top 20 Alert Targets

434	MY.NET.203.118
434	MY.NET.220.190
477	MY.NET.225.58
490	MY.NET.221.246
501	MY.NET.98.181
506	MY.NET.203.206
514	MY.NET.101.192
564	MY.NET.227.190
589	MY.NET.221.146
609	MY.NET.211.178
627	MY.NET.223.254
667	MY.NET.214.74
802	MY.NET.201.174
1280	MY.NET.100.230
1460	MY.NET.218.142
1486	MY.NET.70.255
1639	MY.NET.203.142
3915	MY.NET.206.90
3939	MY.NET.223.98
4813	MY.NET.211.146

2.2.5 Top 20 Attack Sources

667	212.179.44.66
729	212.179.66.2
1085	212.187.21.156
1105	163.10.19.34
1148	24.7.227.215
1212	159.226.91.20
1531	63.167.58.13
1584	143.89.13.3
1591	212.179.72.226

1883	63.193.210.208
2068	211.46.110.81
2338	212.0.107.107
2582	210.101.101.110
3292	195.103.69.159
3295	193.64.114.10
3572	210.113.89.200
3938	212.179.44.115
3950	212.179.79.2
3962	212.179.27.6
6002	212.179.95.5

2.2.6 Top 20 Attack Pairs

365	205.188.153.107-MY.NET.217.214
366	212.179.58.191-MY.NET.207.158
430	212.179.27.6-MY.NET.203.118
475	212.179.24.136-MY.NET.225.58
488	205.188.153.108-MY.NET.221.246
500	212.179.66.2-MY.NET.98.181
505	212.179.50.77-MY.NET.203.206
564	212.179.15.122-MY.NET.227.190
589	212.179.7.58-MY.NET.221.146
609	212.179.72.226-MY.NET.211.178
625	212.179.95.26-MY.NET.223.254
667	212.179.44.66-MY.NET.214.74
794	212.179.95.5-MY.NET.206.90
796	212.179.72.226-MY.NET.201.174
1131	159.226.91.20-MY.NET.100.230
1459	212.179.79.2-MY.NET.218.142
1638	212.179.79.2-MY.NET.203.142
3120	212.179.27.6-MY.NET.206.90
3938	212.179.44.115-MY.NET.223.98
4810	212.179.95.5-MY.NET.211.146

2.2.7 Attack Types

26673	Watchlist 000220 IL-ISDNNET-990517
20991	SYN-FIN scan!
4163	WinGate 1080 Attempt
2893	TCP SMTP Source Port traffic
2001	Attempted Sun RPC high port access
1932	Watchlist 000222 NET-NCFC
1486	Broadcast Ping to subnet 70
1203	Back Orifice
428	SNMP public access
238	Null scan!
209	SMB Name Wildcard
124	Queso fingerprint
88	NMAP TCP ping!
56	connect to 515 from inside
40	SUNRPC highport access!
14	Probable NMAP fingerprint attempt
11	External RPC call
6	Tiny Fragments - Possible Hostile Activity
2	Happy 99 Virus
1	site exec - Possible wu-ftpd exploit - GIAC000623

2.3 **Top 20 Fingerprinting attempts**

991	MY.NET.204.218
670	MY.NET.206.94
518	MY.NET.206.106
360	MY.NET.205.214
341	MY.NET.5.29
329	MY.NET.253.114
302	MY.NET.207.58
126	MY.NET.227.10
93	MY.NET.115.115
73	MY.NET.60.11
64	MY.NET.212.178
53	MY.NET.60.8
51	MY.NET.211.146

49	MY.NET.201.130
45	MY.NET.204.170
42	MY.NET.217.26
38	MY.NET.130.190
37	MY.NET.223.238
37	MY.NET.214.90
37	MY.NET.162.39

Fingerprinting attempts are cause for alarm, since they give an attacker very useful information about the target, allowing the attack to be carefully tuned for maximum success with minimum danger. Fingerprinting is often done with odd or utterly illegal tcp flag combinations. This works because various operating systems will handle these flags in different ways due to the ambiguities of the TCP specification. Between IP characteristics, TCP flags and TCP options it is quite reasonable to expect to be able to remotely determine the target's exact patchlevel. An easy defense against many (but not all) fingerprinting attacks is to block all invalid TCP flags with a bridging filter. The bridge operates transparently, thus it does not require major client reconfiguration, yet it can stop a large number of intelligence-gathering attempts.

Other dangers of portscans include denial of service attacks and remote crashes - older versions of nmap were able to cause some HP/UX machines to reboot, or cause some versions of IRIX inetd to die. This is a known issue with port scans in general - apparently Cisco and others go to a great deal of trouble to ensure that their network scanners do not crash machines. I doubt that attack tools have the same sort of quality standards.

My immediate recommendation would be for the system administrators to acquire a copy of nmap and run it against the entire network to see what sort of information the attackers might have gained. This will also help them prioritize their list of patches and fixes to apply.

2.4 **Synscan / t0rnscan probes**

1	129.93.206.170
1	24.112.150.20
1	24.112.51.119
1	24.200.140.155
1	24.65.121.98
2	129.101.18.16

6	62.96.171.103
51	24.7.227.215
267	139.130.61.206
276	211.46.110.81
1085	212.187.21.156
1105	163.10.19.34
1531	63.167.58.13
1584	143.89.13.3
2338	212.0.107.107
2582	210.101.101.110
3292	195.103.69.159
3295	193.64.114.10
3572	210.113.89.200
20991	**total**

I notice a lot of packets that look like the tool called 'synscan'. There is a repackaged version of this tool in the 't0rn' rootkit called t0rnscan. This tool has a relatively distinct signature: SYN and FIN set, TCP window size is 1028, and the IP ID number is 39426. Obviously there are way more of these characteristics than normal traffic should produce. In any case, SYN+FIN lies somewhere between pathological and malicious, and should be investigated. Especially of interest are scans for tcp/21, tcp/53, and tcp/9704 which are FTP, DNS and rpc.statd respectively, which are historically great places to find root holes.

As I stated earlier, I would suggest filtering invalid flags at the router. This would be a good time to suggest scanning your network for previously undocumented or unauthorized servers. Note that while the tool that generated these alerts sets the source and destination ports equal by default, it is easy to override that behaviour.

2.5 **WinGate Probes**

22	209.212.128.47
23	207.126.106.118
25	216.179.0.37
28	194.84.208.118
41	194.75.152.237
50	168.120.16.250
58	198.139.244.22

89	24.169.61.162
96	207.114.4.46
112	212.72.75.236
126	64.86.5.250
137	204.117.70.5
169	208.194.161.155
179	198.63.2.192
1883	63.193.210.208

These are hosts which have made at least 20 connection attempts. 63.193.210.208 should probably be reported to the appropriate 'abuse@' address, which would be PacBell/SWBell. I must report that I have not had good luck getting timely responses from this ISP.

```
Name: adsl-63-193-210-208.dsl.snfc21.pacbell.net
Address: 63.193.210.208

ADSL BASIC-rback7-snfc21 (NETBLK-SBCIS990913-39)
303 2nd Street Suite 850N
San Francisco, CA 94107
USA

Netname: SBCIS990913-39
Netblock: 63.193.210.0 - 63.193.211.255

Coordinator:
PBI IP Administrator  (PIA2-ORG-ARIN)  ip-admin@PBI.NET
888-212-5411
Fax- 415-442-4999

Record last updated on 13-Sep-1999.
Database last updated on 20-Jan-2001 18:21:20 EDT.
```

tcp/1080 is used for a number of things: test webservers, SOCKS proxies, and a windows package called "wingate". Wingates are often used in an attack to hide the true ip address of the attacking computer, for obvious reasons. A number of societies hold that, if you know someone's True Name, you hold some power over them. Knowing someone's True IP Address aids considerably in gaining some net information about their computer. Proxies are not inherently evil - they do play a very important role in enforcing access controls and policies. As with all network security tools, proxies must be properly installed and configured in order to be effective.

As previously mentioned, I would suggest a service inventory, listing all the services on the network. This is an ongoing process - new servers appear all the time. Also, restricting the networks which can receive incoming connections would be a wise choice.

2.6 **RPC Traffic**

There were quite a number of "false" alarms in the RPC detects, for example, 216.148.218.160:443 was seen sending traffic to an RPC port. As it happens, that IP address resolves to "head.rwc.rhns.redhat.com", thus I classified it as legitimate.

I then filtered out source port 4000 on suspicion of it being icq traffic. These were the bulk of the connections: 97.22% . Most of this traffic was going to 205.188.153.0/24 which belongs to AOL/ICQ. This may not necessarily be a good thing though, depending on site policy regarding personal network use during business hours.

```
America Online, Inc (NETBLK-AOL-DTC)
   22080 Pacific Blvd
   Sterling, VA 20166
   US

   Netname: AOL-DTC
   Netblock: 205.188.0.0 - 205.188.255.255

   Coordinator:
      America Online, Inc.  (AOL-NOC-ARIN)  domains@AOL.NET
      703-265-4670

   Domain System inverse mapping provided by:

   DNS-01.NS.AOL.COM            152.163.159.232
   DNS-02.NS.AOL.COM            205.188.157.232

   Record last updated on 27-Apr-1998.
   Database last updated on 20-Jan-2001 18:21:20 EDT.

www.icq.com             A       205.188.147.56
www.icq.com             A       205.188.147.53
www.icq.com             A       205.188.252.121

   Domain Name: ICQ.NET
   Registrar: AMERICA ONLINE, INC.
   Whois Server: whois.compuserve.com
   Referral URL: domain.compuserve.com
   Name Server: DNS-01.ICQ.NET
   Name Server: DNS-02.ICQ.NET
   Updated Date: 17-nov-1999

Domain Name: ICQ.NET

Registrant:
  ICQ, Inc.
    22000 AOL Way
    Dulles, VA 20166
    US

  Created on..............: Nov 17, 1999
```

```
Expires on..............: Nov 17, 2001
Record Last Updated on..: Nov 17, 1999
Registrar...............: America Online, Inc.
                          http://whois.registrar.aol.com/whois/

Administrative Contact:
  Domain Registration, ICQ
  ICQ, Inc.
  22000 AOL Way
  Dulles, VA 20166
  US
  Email. domains@AOL.NET
  Tel. 703 265 4670

Technical Contact:
  Domain Registration, ICQ
  ICQ, Inc.
  22000 AOL Way
  Dulles, VA 20166
  US
  Email. domains@AOL.NET
  Tel. 703 265 4670

Domain servers:
  dns-01.icq.net
    152.163.159.234
  dns-02.icq.net
    205.188.157.234
```

Of the remaining connections that were not adequately explained by ICQ or https use, approximately 80% of the 53 connections were to port 32771, which seems to be a common place for 'yppasswdd' to run. My explanation for this is that the attacker is attempting to retrieve your NIS password maps for an offline dictionary attack. This will probably give a significant number of userid and password pairs in short order unless you have mechanisms for enforcing strong password selection and are using MD5 rather than old crypt-style passwords. Two ways to defeat this sort of a probe would be to implement a default deny policy or distribute authentication information via more secure methods such as NIS+ or Kerberos.

2.7 **SNMP Probes**

MY.NET.101.92 seems to be some sort of popular snmp device, receiving a total of 428 connection attempts from 412 internal hosts. My guess is that the target is a printer with an snmp agent that people are querying the status on. It would seem the suggestion that the printer be renumbered has been taken, along with the suggestion that SNMP be filtered at the border router since there are no recorded probes from outside. I note that this device does not show up in either the packet traces or the scanlogs.

2.8 **Back Orifice**

It was pleasant to see that nothing seriously indicates BO infestation. The logs show a large number of incoming probes, but no machines with large numbers of connections that one would expect if a machine were infested. I suspect that this might be false alarms due to DNS lookups. I see that happen frequently enough. Naming and shaming the most prolific IP address is in order: 63.46.46.143 or by name: "1Cust143.tnt2.sierra-vista.az.da.uu.net". No packets matching this signature were found in the packet dumps.

2.9 **Directed Broadcast Pings**

24	129.186.67.X
24	193.231.125.X
26	194.102.254.X
29	194.206.208.X
30	208.212.171.X
32	193.231.253.X
35	217.10.201.X
37	194.102.242.X
40	213.154.133.X
41	213.154.134.X
46	193.231.169.X
49	213.154.131.X
50	193.230.162.X
55	193.226.60.X
61	193.231.210.X
62	217.10.206.X
74	194.102.93.X
85	193.230.165.X
98	63.27.120.X
194	193.231.220.X

This is an obvious and easy to defeat recon attempt. It could also be a smurf attempt. Either way, the cure is the same: configure the router not to forward packets to the broadcast address. For an even stronger fix, configure packet filters to not pass incoming echo requests or outgoing echo replies. Echo request include protocols like UDP echo as well as ICMP echo. I feel that this is probably intelligence gathering rather than a smurf attack. I say this because the attacker would want to know if your network will act as a smurf amplifier before spraying your network with pings and getting caught. Multiple responses to a directed broadcast are also an excellent source of network information and should be blocked for privacy reasons.

```
inetnum:      193.231.169.0 - 193.231.169.255
netname:      MEDIASAT
descr:        Media Sat S.A.
country:      RO
admin-c:      MSAT1-RIPE
tech-c:       MSAT1-RIPE
tech-c:       MSAT1-RIPE
rev-srv:      thorin.mediasat.ro 193.231.169.2
rev-srv:      bombur.mediasat.ro 193.231.169.3
status:       ASSIGNED PA
remarks:      object maintained by ro.rnc local registry
notify:       domain-admin@rnc.ro
mnt-by:       AS3233-MNT
changed:      danacorb@sunu.rnc.ro 19980209
changed:      estaicut@rnc.ro 19981123
changed:      peter@rnc.ro 20000525
source:       RIPE

route:        193.231.169.0/24
descr:        Media Sat S.A
origin:       AS8751
mnt-by:       AS3233-MNT
changed:      estaicut@rnc.ro 19980812
source:       RIPE

role:         MEDIASAT Staff
address:      Mediasat SA
address:      Bd. Ferdinand, nr 99
address:      Bucharest, Romania
phone:        +40-1-2050 637
fax-no:       +40-1-2050 655
e-mail:       mnt@mediasat.ro
admin-c:      AC318-RIPE
tech-c:       AC318-RIPE
nic-hdl:      MSAT1-RIPE
remarks:      Role for Mediasat staff
remarks:      Team members are listed below:
remarks:      adc@mediasat.ro, iia@mediasat.ro, budha@mediasat.ro
notify:       mnt@mediasat.ro
changed:      adc@mediasat.ro 19990402
source:       RIPE

inetnum:      194.102.242.0 - 194.102.242.127
netname:      ROCLUBNET
descr:        Soft Trading & Co SRL - Sibiu 2400
descr:        Str Lamark 3
country:      RO
admin-c:      RG3598-RIPE
tech-c:       VF132-RIPE
status:       ASSIGNED PI
remarks:      object maintained by ro.rnc local registry
```

```
notify:      domain-admin@rnc.ro
mnt-by:      AS3233-MNT
changed:     danacorb@rnc.ro 19991112
source:      RIPE

person:      Razvan Gliga
address:     SOFT TRADING & CO SRL
address:     SIBIU
address:     str. Lamark nr. 3
phone:       +40 69 236212
fax-no:      +40 69 236212
e-mail:      razvang@stc.sibnet.ro
nic-hdl:     RG3598-RIPE
remarks:     object maintained by ro.rnc local registry
notify:      domain-admin@rnc.ro
mnt-by:      AS3233-MNT
changed:     danacorb@rnc.ro 19991005
source:      RIPE

person:      Virgil Frum
address:     Telecom Services  S.R.L.
address:     Str. I.C. Bratianu Nr.2 ap.4
address:     Medias, 3125
address:     Romania
phone:       +40-69-811989
fax-no:      +40-69-811989
e-mail:      virgil@telecom.ro
nic-hdl:     VF132-RIPE
remarks:     object maintained by ro.rnc local registry
notify:      domain-admin@roearn.ici.ro
mnt-by:      AS3233-MNT
changed:     danacorb@sunu.rnc.ro 19980219
source:      RIPE

inetnum:     213.154.128.0 - 213.154.130.255
netname:     PCNET
descr:       PCNET Data Network - ISP in Romania
country:     RO
admin-c:     MB51-RIPE
tech-c:      AN160-RIPE
status:      ASSIGNED PA
notify:      alina@pcnet.ro
changed:     hostmaster@ripe.net 20000128
source:      RIPE

person:      Mihai Batraneanu
address:     PC-NET Data Network S.A.
address:     Splaiul Unirii 10, bl B5, sc2, et 1
address:     Bucharest, Romania
phone:       +40-1-330 28 01
fax-no:      +40-1-330 28 42
e-mail:      mihai@pcnet.ro
nic-hdl:     MB51-RIPE
remarks:     object maintained by ro.rnc local registry
notify:      domain-admin@listserv.rnc.ro
mnt-by:      AS3233-MNT
changed:     danacorb@sunu.rnc.ro 19970901
```

```
changed:      ciprian@rnc.ro 19991207
source:       RIPE

person:       Alina-Mihaela Nemes
address:      PC-NET Data Network S.A.
address:      Splaiul Unirii 10, bl B5, sc2, et 1
address:      Bucharest, Romania
phone:        +40-1-330 28 01
fax-no:       +40-1-330 28 42
e-mail:       alina@pcnet.ro
nic-hdl:      AN160-RIPE
remarks:      object maintained by ro.rnc local registry
notify:       domain-admin@listserv.rnc.ro
mnt-by:       AS3233-MNT
changed:      danacorb@sunu.rnc.ro 19970901
changed:      ciprian@rnc.ro 19991207
source:       RIPE
```

2.10 **SMB Probes**

source	dest	connections	percentage
MY.NET.101.160	MY.NET.101.192	86	41.34%
141.157.99.21	MY.NET.6.15	33	15.86%
169.254.184.161	MY.NET	24	11.53%
141.157.98.201	MY.NET.6.15	20	9.61%

Incoming netbios/SMB is generally considered to be a Bad Thing, due to at least four possible unpleasant results:

- successful recon attempt - where's the PDC, etc?
- possible denial of service - flood the printers, DCs, etc.
- possible modification of data - viruses that spread via open shares
- possible loss of confidentiality - pwdump/l0phtcrack, or reading files

I would suggest that these machines be examined to see what exactly is being shared and who is accessing it. Of particular concern is whether some of these attempts are from "partnernets". Perhaps these machines should have some sort of packet filters installed - IPFilter, IPFW, IPChains, NetFilter or ZoneAlarm are just a few of the packages which may be considered for this task.

2.11 **Connect to 515 from inside**

I suspect that this is a legitimate printer being connected to. Most lpd implementations I know of require a connection from a port <1024 (hence lpr and friends being suid root). That condition holds here, which is a point in favour of legitimate printing. What bothers me though is the fact that there is only one host trying to print. Then again, it could be a special print job. That would seem reasonable based on the fact that all but 4 of the detected packets were arrived from one internal source for the target machine within a half hour or so. To me this suggests the following conversation - "A: Hey, can I run off a few pages on your fancy phaser laser printer?" "B: Sure. the IP address is MY.NET.101.142"

2.12 **Miscellaneous Alerts**

There was one attempt to run a WU-ftpd exploit against MY.NET.205.94. This host should be checked for signs of a compromise. Was it running a vulnerable server or had it already been secured? Is it an official FTP server, anyway? There are a number of CVE alerts about this daemon.

Snort detected a few instances of the "happy99" virus. The offending packet did not appear in the packet dumps. Mail logs may be useful, but I doubt this is a virus: generally if one comes in, many go out; or at least that's the pattern I've seen with mail viruses like "snow white" or "melissa". I suspect this is a false alarm. That said, I'd recommend installing desktop firewall software such as ZoneAlarm on the Windows machines. This can help protect against recon attempts like port scans, smb probes and pings, but it can also intercept possibly harmful executable content.

2.13 **Watchlist: ISDNNET**

In the last few months, ISDNnet apparently has not taken any meaningful steps to keep their users in line. An (ab)?normally high fraction of alerts were generated by this ISP. Most of the probes sent were relatively typical: ftp, telnet, mail, gnutella. There were also a large number of probes to napster ports, as well as in the port 4000 - 5000 range. This suggests fishing expeditions, looking for trojans and back doors.

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

person:      Nati Pinko
address:     Bezeq International
address:     40 Hashacham St.
address:     Petach Tikvah  Israel
phone:       +972 3 9257761
e-mail:      hostmaster@isdn.net.il
```

```
nic-hdl:     NP469-RIPE
changed:     registrar@ns.il 19990902
source:      RIPE
```

unique connection types

11	mailserver
12	port 4619
8	port 4922
132	port 4XXX
4	gnutella
31	napster

2.14 **Watchlist: NCFC**

Crist Clark noted an "alarming" amount of telnet activity with NCFC. This seems to have tapered off, with all 123 connections originating from two hosts both of whom were interested in talking to 159.266.41.166.

```
The Computer Network Center Chinese Academy of Sciences (NET-NCFC)
   P.O. Box 2704-10,
   Institute of Computing Technology Chinese Academy of Sciences
   Beijing 100080, China

   Netname: NCFC
   Netblock: 159.226.0.0 - 159.226.255.255

   Coordinator:
      Qian, Haulin  (QH3-ARIN)  hlqian@NS.CNC.AC.CN
      +86 1 2569960

   Domain System inverse mapping provided by:

   NS.CNC.AC.CN                 159.226.1.1
   GINGKO.ICT.AC.CN             159.226.40.1
```

unique connection types

2	ftp
9	https
60	ident
147	mailserver
3	telnet

3.0 Analysis Process

Obviously there's no way I'm going to look at all of these traces by hand. This is the sort of thing that PERL was created for. I thought about using shell and piping things through grep, sort, sed, uniq, cut, tr, awk, and a number of other utilities, but filling up a fairly large /tmp convinced me rather quickly of the error of my ways. I then turned to PERL, and decided that I was willing to make a space-for-time trade, and tossed another 128M of memory at the problem. I strongly recommend making at least 128M of free physical memory available for chewing through the data, and making a mug of coffee while the programs run. Once the alerts and scans had been processed and broken into smaller slices, most of this data got run through grep, 'uniq -c', and sort a few times. The '-c' flag to uniq causes this program to prepend each line with a count of how many times it was seen.

Below is a collection of quickie PERL scripts I used to analyze the Snort data. Most of these demand at least one command line option, and will print out a help message if not given any action flags. The code isn't exactly the most elegant, but it worked on the assignment data and it works well on my production network.

alertcount - This program tallies up the alert files, and spits out various interpretations, including the number and type of alerts generated by a host, or received by a host, or by a given host pair. It can also ignore certain patterns, which is useful if you've already processed that source, destination or alert type. Besides printing a list of all the alerts generated by a given source or destination host, it can generate just a total count of alerts too. No, it will not make Julienne fries.

```
 #!/usr/bin/perl -s

unless (defined($d) ||defined($s) ||defined($q) ||defined($p) ||defined($
        print "you need to specify at least one action flag\n";
        print "\t-d \tprint the destination hosts\n";
        print "\t-s \tprint the source hosts\n";
        print "\t-p \tprint the attacker/target pair\n";
        print "\t-t \tprint the attack types\n";
        print "\t-q \tbe quiet and print the total number of detects\n";
        print "\t-v \tbe verbose and print everything\n";
        print "\t-a \tprocess all (don't ignore portscans)\n";
        print "\t-i=file\tread a list of patterns to skip from \n";
        print "\t-l=n\tthreshold before printing\n";
        exit 1;
        }

if (defined($v) && defined($q)){
        print "the '-q' and '-v' flags are mutually exclusive.\n";
        exit 1;
        }

#the skip list contains case-sensitive patterns, one per line
#of strings, which, if found in the alert, cause processing of
```

```
#that alert to be skipped.
if (defined($i)) {
        open(SKIPLIST,$i) || die "can't open skip list \"$s\" ! ($!)\n";
        while (){
                chomp;
                push(@skiplist,$_)
        }
        close SKIPLIST;
}

while (<>){
        chomp;

        #make sure we have a log line
        unless (/\Q [**] \E/){ next };

#assuming that there are any alerts we're not interested in, we skip them
#here. portscans shouldn't be that interesting, since we have all the
#output from the portscan logger.
        $skipthis=0;
        if (( /spp_portscan/i ) && ( !defined($a) )){ next; }
        foreach $s (@skiplist){
                if ( $_ =~ /$s/ ){ $skipthis=1; }
        }; if ($skipthis) { next; }

        ($timestamp,$desc,$ip)=split(/\Q [**] \E/, $_);
        ($src, $arrow, $dst) = split(/ /, $ip);
        ($s_h,$s_p)=split(/:/,$src);
        ($d_h,$d_p)=split(/:/,$dst);
        $pkey = "${s_h}-${d_h}XXX$desc";
        $skey = "${s_h}XXX$desc";
        $dkey = "${d_h}XXX$desc";

        $atype{$desc} += 1 ;
        $pair{$pkey} += 1 ;
        $asrc{$skey} += 1 ;
        $adst{$dkey} += 1 ;

        $at2{$desc} += 1 ;
        $pr2{"${s_h}-${d_h}"} += 1 ;
        $as2{"${s_h}"} += 1 ;
        $ad2{"${d_h}"} += 1 ;

}

if (((!$q)&&($t))||($v)){
        foreach $key (sort keys(%atype)){
                print "$atype{$key}\t$key\n";
        }
}

if (((!$q)&&($d))||($v)){
        $state = "0xc0ffee";
        foreach $key (sort keys(%adst)){
                ($connection,$crime) = split(/XXX/, $key);
                unless ($connection =~ /$state/){
                        print "\n$connection\n";
                        print ( ("=" x 31 ) . "\n");
                        $state="$connection";
                }
                print "$adst{$key}\t$crime\n";
        }
}

if ((!$q)&&($s))||($v)){
        $state = "0xc0ffee";
        foreach $key (sort keys(%asrc)){
                ($connection,$crime) = split(/XXX/, $key);
                unless ($connection =~ /$state/){
                        print "\n$connection\n";
                        print ( ("=" x 31 ) . "\n");
                        $state="$connection";
                }
                print "$asrc{$key}\t$crime\n";
        }
}

if ((!$q)&&($p))||($v)){
        $state = "0xc0ffee";
        foreach $key (sort keys(%pair)){
                ($connection,$crime) = split(/XXX/, $key);
                unless ($connection =~ /$state/){
                        print "\n$connection\n";
                        print ( ("=" x 31 ) . "\n");
                        $state="$connection";
                }
                print "$pair{$key}\t$crime\n";
        }
}

if (($t)&&($q)){
        foreach $key (sort keys(%at2)){
                print "$at2{$key}\t$key\n";
        }
}

if (($d)&&($q)){
        foreach $key (sort keys(%ad2)){
                print "$ad2{$key}\t$key\n";
        }
}


if (($s)&&($q)){
        foreach $key (sort keys(%as2)){
                print "$as2{$key}\t$key\n";
        }
}
if (($p)&&($q)){
        foreach $key (sort keys(%pr2)){
                print "$pr2{$key}\t$key\n";
        }
}
```

ipsort - this program takes in some files (or stdin), sorts them by IP address and then prints the results to stdout. I suppose I could have stolen the code from shadow, but at the time I needed some way to sort ip addresses *correctly.* This has the advantage that you can specify what field on a line contains the sort key. Under the hood, it just 0-pads the octets in the address and then tries to undo the damage after the sort. Much like old police cars, it's ugly, but it works pretty well. I suppose I should rewrite it to use hex, rather than stripping out 0s.

```
#!/usr/bin/perl -s

while (<>){
        chomp;

        @words=(); @prefix=();
        @words=split;
        if ($f){
                for($n=1; $n<$f; $n++){
                        push(@prefix, shift @words);
                }
                $ip=shift @words;
                @ip=split(/\./,$ip);
                $addr_str=sprintf "%03d.%03d.%03d.%03d", @ip;
        }
        $newline=join(" ", $addr_str, @prefix, @words);
        push(@data,$newline);
}

@data = sort { ($a <=> $b) || ($a cmp $b) } @data;
if ($r) {
        @data = reverse @data;
}

foreach (@data) {
        @prefix=(); @words=(); $ip="";
        @words = split (" ", $_);
        if ($f){
                $ip2 = shift @words;
                        $ip2 =~ s/\.0+/\./g;
                        $ip2 =~ s/^0+//g;
                        $ip2 =~ s/\.\./\.0\./g;
                        $ip2 =~ s/\.$/\.0/g;
                        for($n=1; $n<$f; $n++){
                                push(@prefix, shift @words);
                        }
                }
        print "@prefix $ip2 @words\n";
}
```

scanalyze - This program reads in the portscan log and spits out a more parseable form. The scan is sorted by time, and the timestamps are printed in numeric form (no month names). Arbitrarily I decided that UDP and SYN scans weren't quite as interesting as the other scan types, since they false-positive quite easily, so the program does not count them by default. There is an option flag to make the program count everything.

```
#!/usr/bin/perl -s

#this program can be quite resource intensive. it took about 3.5min CPU
#time running under 'nice -20' on my pIII 550, and used 60MB memory to
#process 19.5MB of data. it must be that sort using all that memory.

#use the '-a' flag to count the UDP and SYN scans too. These are ignored
#by default for fear of false positives. be mindful though that you don't
#skip something important.

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

while (<>){
        chomp;
        #check to make sure this is a real log entry
        unless (/^... .. ..:..:../) { next; }

        #snarf in and split, and prepare a log line
        ($mon, $day, $time, $src, $arrow, $dst, @scantype) = split;
        $mon=$convert{$mon};
        ($s_h,$s_p)=split(/:/,$src);
        ($d_h,$d_p)=split(/:/,$dst);
        $newline=join(" ", ("$mon.$day",$time,$s_h,$d_h,@scantype));

        #append the new line onto the new logfile
        push(@newlog,$newline);
}

#sort the newlog
@newlog = sort {($a <=> $b) || ($a cmp $b) } @newlog;

#uniq the newlog, and print interesting things.
$lastline = "0xc0ffee";
foreach $line (@newlog){
        #nasty kludge to escape the printed "*" characters, and to see if
        #we've printed this line already.
        if ($lastline =~ /\Q$line\E/){
                #no-op.
        } else  {
                #save the line
                $lastline = $line;
                if ($a){
                        print "$line\n";
                } else {
                        #match and skip the "boring" bits
                        unless (($line =~ / UDP/)||($line =~ / SYN /)){
                                print "$line\n";
                        }
                }
        }       #"no-op" comes here...
}
```

scancount - This program takes input from scanalyze and generates reports. The usual src-only, dst-only, scantype-only and src-dst pair reports are supported. When run with the '-l=n' option, the program will not print a key unless that key has been seen at least n times. for example `'-s -l=16'` will only print source hosts that have sent at least 16 probes. a good way to drop some of the noise. Certain broken linuxen will set the R1+R0 TCP flags under normal use, which might trip an alarm. this is a way to get a short list of the really nasty people... and wouldn't you know it, their IP's often seem to come in clusters.

```
#!/usr/bin/perl -s

unless (defined($d) ||defined($s) ||defined($p) ||defined($t) ||defined($
        print "you need to specify at least one action flag\n";
        print "\t-d \tprint the target hosts\n";
        print "\t-s \tprint the attacking hosts\n";
        print "\t-p \tprint the attacker/target pair\n";
        print "\t-t \tprint the attack type\n";
        print "\t-f \twatch for fingerprinting attempts\n";
        print "\t-v \tbe verbose and print everything\n";
        print "\t-l=n\tconnection threshold before printing\n";
        exit 1;
        }

$l = 0 unless defined($l);

while (<>){
        chomp;
        ($date, $time, $src, $dst, $scantype, @scanopts) = split;
        $pkey = "$src-$dst";

        unless (($scantype =~ /SYN/)||($scantype =~ /UDP/)||($scantype =~
                ++$fsrc{$src};
                ++$fdst{$dst};
                ++$fpr{$pkey};
                ++$ftyp{$scantype};
                }
        ++$asrc{$src};
        ++$adst{$dst};
        ++$type{$scantype};
        ++$pair{$pkey};
}

if (($t)||($v)){
        print "\n\nUnique Scan Types\n=================\n\n" if ($v) ;
        foreach $key (sort keys(%type)){
                if(($f)&&($ftyp{$key}>0)){ $fp="\t(fp)"; }else{ $fp=""; }
                print "$type{$key} \t$key $fp\n" unless ($type{$key} < $l
        }
}

if (($d)||($v)){
        print "\n\nUnique Targets\n==============\n\n" if ($v) ;
        foreach $key (sort keys(%adst)){
                if(($f)&&($fdst{$key})){$fp="\t(fp)";}else{$fp="";}
                print "$adst{$key} \t$key $fp\n" unless ($adst{$key} < $l
        }
}
```

```
if (($s)||($v)){
        print "\n\nUnique Attackers\n================\n\n" if ($v) ;
        foreach $key (sort keys(%asrc)){
                if(($f)&&($fsrc{$key})){$fp="\t(fp)";}else{$fp="";}
                print "$asrc{$key} \t$key $fp\n" unless ($asrc{$key} < $l
        }
}

if (($p)||($v)){
        print "\n\nUnique Attacks/Targets\n======================\n\n" if
        foreach $key (sort keys(%pair)){
                if(($f)&&($fpr{$key})){$fp="\t(fp)";}else{$fp="";}
                print "$pair{$key} \t$key $fp\n" unless ($pair{$key} < $l
        }
}
```