Global Information Assurance Certification Paper

Intrusion Detection in Depth

GCIA Practical Assignment Version 3.0
Glenn Larratt

Table of Contents

Assignment 1 - The State of Intrusion Detection

Lcrzo and Lcrzoex

Lcrzo is described by its author as a "Swiss knife for network developpers[sic]". It is primarily a network library dependent on libpcap, that includes modules for packet sniffing and traffic generation, including various sorts of spoofing and network attacks. Lcrzoex is a companion set of example programs that includes a very simple but quite useful front end for various features.

In this paper, I will explore a few of the many aspects of Lcrzo, with an eye toward intrusion detection applications. Using Lcrzoex' example programs , I will demonstrate packet sniffing, and some simple examples of bad behavior.

I used a Red Hat Linux laptop, already installed with libpcap 0.4, tcpdump, and Snort, as a test host for various Lcrzo exercises, and made use of a production Snort host as well for monitoring of Lcrzo's packet generation.

A word about Lcrzo and file formats

Lcrzo can read and write files in pcap format, but most of the examples in Lcrzoex don't - in instances where that format was necessary, I found I needed to use two Lcrzoex example modules to convert between pcap format and Lcrzo's preferred "record format", which is textual.

For the purposes of testing, I used tcpdump to record several packets, and converted them to Lcrzo's record format and back to pcap. I found that Lcrzoex' conversion utility, though it corrupts the pcap header time information (Lcrzo's own formats don't timestamp packets), was robust in reproducing the packet data precisely, as long as they were complete in the first place. This experiment ran into problems initially because of tcpdump's default snaplen.

Lcrzo has multiple display modes; some are more useful than others. Lcrzo modules don't have nearly the protocol decoding capabilities that tcpdump does, but it will decode IP, ICMP, TCP, and UDP headers. One of the more useful modes is "synthetic aspect", which is a bare-bones look similar to tcpdump -n or tcpdump -nq, displayed in the trace below; this mode is available either with or without the MAC address information. Another useful mode is "header in array and data in dump", which does an excellent job of presenting the decodes - it would make a fine teaching tool.

```
ETH_____________________________________________________________________.
| 00:50:0f:a6:10:80 vers 01:00:5e:00:00:05         type : 0x0800          |
|_____________________________________________________________________|
IP______________________________________________________________________.
|version |  ihl    |        tos        |               totlen              |
|___ 4___|___ 5___|_______192________|_____________0054h=   84____________|
|                id                  |xxDfMf          fragoffset         |
|_____________837Dh=33661_____________|0_0_0__________0000h=    0_________|
|       ttl        |    protocol      |          header checksum          |
|_____01h=  1_____|_____59h= 89_____|_________________CEE8h_______________|
|                                source                                   |
|_____________________________MY.NET.5.252_____________________________|
|                              destination                                |
|_____________________________224.0.0.5_____________________________|
02 01 00 40  XX XX FD FC  00 00 00 00  20 F9 00 00     # ...@........ ...
00 00 00 00  00 00 00 00  FF FF FF 00  00 0A 02 01     # ................
00 00 00 28  XX XX 05 FC  XX XX 05 F1  XX XX FD FA     # ...(............
XX XX 05 07  C0 88 90 FE  XX XX FD FB  XX XX FD F8     # ................
```

Lcrzo as a sniffer

Using tcpdump, I took this sample of "background radiation" - OSPF "hellos" from our test router:

```
15:30:00.716089 0:50:f:a6:10:80 1:0:5e:0:0:5 0800 98: MY.NET.5.252 > 224.0.0.5: \
OSPFv2-hello 64: rtrid MY.NET.253.252 backbone [|ospf] [tos 0xc0] [ttl 1]
15:30:10.716005 0:50:f:a6:10:80 1:0:5e:0:0:5 0800 98: MY.NET.5.252 > 224.0.0.5: \
OSPFv2-hello 64: rtrid MY.NET.253.252 backbone [|ospf] [tos 0xc0] [ttl 1]
15:30:20.715913 0:50:f:a6:10:80 1:0:5e:0:0:5 0800 98: MY.NET.5.252 > 224.0.0.5: \
OSPFv2-hello 64: rtrid MY.NET.253.252 backbone [|ospf] [tos 0xc0] [ttl 1]
```

Lcrzoex, if run without arguments, displays a rudimentary menu of choices; "packet sniffing" has its own subsidiary menu, with options for reassembling fragmented IP and reordering of TCP sequences, and output to console or two different file types.

```
00:50:0f:a6:10:80->01:00:5e:00:00:05 - MY.NET.5.252->224.0.0.5 - 64 bytes
00:50:0f:a6:10:80->01:00:5e:00:00:05 - MY.NET.5.252->224.0.0.5 - 64 bytes
00:50:0f:a6:10:80->01:00:5e:00:00:05 - MY.NET.5.252->224.0.0.5 - 64 bytes
```

It's not a replacement for Snort or tcpdump, but it's not intended to be - Lcrzo is a library, which allows the user to graft this capability into his own programs. The data is accurately recorded - I performed a test by sniffing with Lcrzoex, converting to pcap format, and reading the results with tcpdump. In the resulting output, except for the previously noted issue with timestamping, all data was precisely rendered.

Lcrzo for (arguably) bad behavior

I used another Lcrzoex module to send myself forged e-mail, with the resulting headers in my mailbox:

```
Return-Path: <foo@sans.org>
Delivered-To: ME@MAILHOME.MY.NET
Received: from localhost (localhost [127.0.0.1])
        by MAILHOME.MY.NET (Postfix) with ESMTP id 9075C39999
        for <ME@MAILHOME.MY.NET>; Thu,  7 Feb 2002 20:29:24 -0600 (CST)
Received: from MAILGW.MY.NET (MY.NET [MY.NET.5.4])
        by MAILHOME.MY.NET (Postfix) with ESMTP id BA0B239994
        for <ME@MAILHOME.MY.NET>; Thu,  7 Feb 2002 20:29:23 -0600 (CST)
Received: from sans.org (LCRZOHOST.MY.NET [MY.REAL.IP.20])
        by MAILGW.MY.NET (8.9.0/8.9.0) with SMTP id UAA29207
        for <ME@MY.NET>; Thu, 7 Feb 2002 20:29:22 -0600 (CST)
Message-ID: <7FEA7533.79B4A8E0@domain.com>
Date: Thu, 07 Feb 2002 18:29:20 -0800
```

```
From: "bar@sans.org" <bar@sans.org>
Reply-To: "baz@sans.org" <baz@sans.org>
Organization: LCRZO LCRZOEX
X-Mailer: Lcrzoex Mailer
X-Accept-Language: en
MIME-Version: 1.0
To: "undisclosed" <undisclosed@MAILGW.MY.NET>
Subject: TEST
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 8bit
```

Clearly, in the default mode, Lcrzo will add identifying information; however, the package is distributed as source code. With a little tweaking, it could be trivially reconfigured to be much more obscure.

I also used two of the Lcrzoex portscan modules. I was able to successfully scan one of my employer's networks for hosts answering on the ssh port, both in an obvious fashion and using spoofed addresses at both the IP and Ethernet layers. These traces are from the Snort instance on my employer's network backbone (which suffers from some packet loss):

```
20:41:55.320813 MY.REAL.IP.20.20707 > MY.NET.42.24.22: S [tcp sum ok] 744564073:744564073(0) \
win 1500 (ttl 127, id 20707, len 40)
20:41:56.079847 MY.REAL.IP.20.20707 > MY.NET.42.62.22: S [tcp sum ok] 744564073:744564073(0) \
win 1500 (ttl 127, id 20707, len 40)
20:41:56.099929 MY.REAL.IP.20.20707 > MY.NET.42.63.22: S [tcp sum ok] 744564073:744564073(0) \
win 1500 (ttl 127, id 20707, len 40)
        :
20:44:30.453635 A.SPOOF.IP.199.8791 > MY.NET.42.79.22: S [tcp sum ok] 2040951674:2040951674(0) \
win 1500 (ttl 127, id 8791, len 40)
20:44:30.713375 A.SPOOF.IP.199.8791 > MY.NET.42.92.22: S [tcp sum ok] 2040951674:2040951674(0) \
win 1500 (ttl 127, id 8791, len 40)
20:44:30.812984 A.SPOOF.IP.199.8791 > MY.NET.42.97.22: S [tcp sum ok] 2040951674:2040951674(0) \
win 1500 (ttl 127, id 8791, len 40)
```

It's worthy of note that the TTL's, IP ID's, source ports and sequence numbers don't change in the course of one of these scans.

I also tested a scan of a single host on all well-known TCP ports (cf. nmap). The scan took only a few seconds, and managed not to crash inetd on the remote host, which we've known nmap to do.

Lcrzoex has some modules for sniffing traffic and replaying it, with options to selectively change addresses and content; however, I had difficulty getting these to work, and rather than create problems in my employer's network, I have elected not to test them without a closely controlled environment in which to do so.

Other modules include well-known attacks such as SYN flood, session teardown ability similar to tcpkill, and techniques reminiscent of LaBrea, in which the Lcrzoex host will answer every ARP, or send an ACK in response to every SYN.

Implications for intrusion detection

Lcrzo is a powerful tool, much like its brethren Snort and tcpdump, but it's intended as a module for inclusion in software. Lcrzoex modules perform a variety of useful tasks from a very simple front end. Taken together, the two have features that could be useful to intrusion analysts and other security personnel, in such areas as:

- penetration testing / verification of IDS and firewall rules
- development of new IDS and firewall rules, with the playback feature
- training

The documentation is sparse in places, but Laurent Constantin has taken the time to document it, and present user interfaces, in both his native French and in English.

There are some features not currently included in the packages which would add a great deal of value. The ability to natively read and write pcap files is one example; another would be the ability to explicitly manipulate TCP flags

and other packet header values.

References

- "Laurent Constantin's web", the source for Lcrzo and Lcrzoex
- TCPDUMP public repository
- Snort - The Open Source Network IDS
- Nmap -- Free Steach Port Scanner ...
- dsniff (which package includes tcpkill)
- LaBrea - The Tarpit

Assignment 2 - Network Detects

Two sensors of different types were used to accumulate the data in this section, both from the author's company network at a U.S. university.

Sensor "Boris" is the campus network border router, a Cisco 7507 that is configured with a very long border ACL for firewall and IDS functions. Boris syslogs to "Shaun": Sun Ultra 10, Solaris 2.6 . Traces recorded by Boris are filtered for readability, using a locally-developed Perl script "logfilt". Boris commonly misses 95% of traffic because of load.

Sensor "Gus" is a Snort host: Sun Netra X1, Solaris 2.8, Snort 1.8.1. Gus is a two-interface host, with an interface on each of the two Fast Ethernet taps from Boris into the campus core.

Time synchronization among Boris, Shaun, and Gus is by way of NTP, talking to a local Stratum 3 server.

Detect #1 - Spoofed TCP / ICMP Unreachables

from Boris, recorded on Shaun - total of 41485 lines recorded, some with multiple packets:

```
Nov  4 21:44:11 icmp permitted 216.197.146.102 -> MY.NET.77.125 (3/1), 1 packet
Nov  4 21:44:13 icmp permitted 216.197.146.102 -> MY.NET.117.169 (3/1), 1 packet
        :
        :
Nov  6 02:31:58 icmp permitted 216.197.146.102 -> MY.NET.171.123 (3/1), 1 packet
Nov  6 02:31:59 icmp permitted 216.197.146.102 -> MY.NET.31.150 (3/1), 1 packet
```

from Gus - total of 38110847 packets recorded:

```
11/04 22:11:32.886428 216.197.146.102 > MY.NET.169.132: icmp: host \
24.251.166.236 unreachable for MY.NET.169.132.1220 > 24.251.166.236.1169: \
[|tcp] (ttl 101, id 17545, len 48) (ttl 245, id 0, len 56)
        :
        :
11/06 02:31:01.630517 216.197.146.102 > MY.NET.167.167: icmp: host \
24.251.166.236 unreachable for MY.NET.167.167.1147 > 24.251.166.236.1123: \
[|tcp] (ttl 101, id 46156, len 48) (ttl 245, id 0, len 56)
```

1. Source of Trace: the author's network

2. Detect was generated by: Boris. Boris' syslog data (standard Cisco) on Shaun is filtered using our Perl script for readability. Format is as follows:

```
                                                    type/code
        date/time    prot    srcIP        ->     dstIP      vvv   pkt count
```

```
Nov  4 21:44:11 icmp 216.197.146.102 -> MY.NET.77.125 (3/1), 1 packet
```

Data from snort logfiles is generated using tcpdump -nvvr. Format is as before with tcpdump data, except that the encapsulated packet that caused the ICMP message is in fuchsia:

```
11/04 22:11:32.886428 216.197.146.102 > MY.NET.169.132: icmp: host \
24.251.166.236 unreachable for MY.NET.169.132.1220 > 24.251.166.236.1169: \
[|tcp] (ttl 101, id 17545, len 48) (ttl 245, id 0, len 56)
```

This detect is actually more interesting for the source information detail in the ICMP payload than the ICMP itself; the original packets, as far as we can reconstruct, would have looked something like this (no data can be reconstructed regarding timestamps, TCP flags, etc.):

```
11/04 22:11:32.000000   MY.NET.169.132.1220 > 24.251.166.236.1169: [|tcp] \
(ttl 101, id 17545, len 48)
```

3. Probability the source address was spoofed: unlikely for the ICMP, almost certain for stimulus traffic that caused it.

The probability of the ICMP traffic being spoofed as 216.197.146.102 approaches zero. ICMP Unreachables don't generate any response, so there's no motivation for spoofed ICMP Unreachables to be sent.

The probability of the TCP traffic to 24.128.147.68 being spoofed approaches unity: the author's /16 is subnetted at /24, and subnets 117, 169, 171, and 222 are examples of subnets that do not exist in MY.NET. Much of the TCP traffic encapsulated within the ICMP has "source" addresses in such unallocated /24 subnets. **4. Description of the attack:** result of a brute force DoS. Given the volume of traffic (38,000,000+ packets in two days), and the fact that the TCP traffic's target host, 24.251.166.236, is part of a cable modem netblock, the indications of DoS are pretty clear.

5. Attack mechanism: DoS agent simply sends enough traffic to the target host to knock it offline. The distribution of original IP ID and TTL values strongly suggests a DDoS: source and destination ports are randomly and evenly distributed, but every one is in the range 1024-1279. TTL values in the original TCP vary in an odd pattern which suggests multiple loci:

original TTL	92	94	95	96	97	99	100	101
count	1855	15758	1566	610815	108642	142	23364	37348706

IP ID values vary in such a way as to suggest a precisely 16 hosts and crafted packets - or some kind of odd bug in the exploit code, in which specific values wind up in the IP ID field.

original ID	1165	13386	17545	21709	25612	29771	33930	38089
count	2410309	24387n23	2385165	2270841	2376470	2416820	2300451	2401019
original ID	41997	46156	50315	5324	54474	58377	62536	9227
count	2394415	2368421	2392136	2373766	2397676	2398634	2429759	2348351

6. Correlations: I could find no references to this DDoS technique in any of CVE, CERT, or Securityfocus. This may be a function of the criteria being difficult to index ("port 1024-1279", etc.).

7. Evidence of active targeting: 24.251.166.236 is deliberately targeted. It's not clear whether MY.NET is the lone spoof victim.

8. Severity:

Severity = (Criticality + Lethality) - (Host Countermeasures + Network Countermeasures)

For MY.NET:

Criticality = 5 *every* host in our network

Lethality = 0 ICMP 3 is advisory
Host = 4 no host is susceptible to ICMP Unreachables
Network = 3 nIDS immediately alerted of this traffic

(5 + 0) - (4 + 3) = **-2**

For the target host:

Criticality = 4 no way of knowing, but it's probably a single host machine
Lethality = 2 Net effect is that host is offline for the duration
Host = 0 no host-based countermeasures against a DDoS
Network = 1 if his ISP told him about the attack

(4 + 2) - (0 + 1) = **5**

9. Defensive Recommendations: As this is a 'no harm done' scenario for MY.NET, no defensive adjustments need be made. With regard to the DDoS target, it's not clear what can be done to defend against DDoS attacks, although a home broadband user might be in a position to connect by way of more than one POP, and thus not be necessarily tied to one address space or uplink provider.

10. Question: Would blocking ICMP type 3 code 1 packets at the border of MY.NET be an effective defensive strategy against this attack?

A. Yes, it would stop the DDoS
B. Yes, it would prevent the DDoS from spreading to MY.NET
C. No, because that would block traffic that allows TCP/IP to function smoothly.
D. No, because the ICMP is a by-product of the attack, not the attack itself.

The correct answer is D.

Detect #2 - SSH Recon, Buffer Overflow Attempt

This detect is from Gus - a total of 5270 packets were recorded, of which several thousand were direct reconnaissance on ssh, several hundred were indirect recon, i.e. DNS lookups on our name servers, and 1150 were an attack on MY.NET.194.150 .

```
02/02 04:50:26.178295 64.35.105.238.22 > MY.NET.0.144.22: tcp 0
02/02 04:50:26.178626 64.35.105.238.22 > MY.NET.0.145.22: tcp 0
        :
        :
02/02 04:52:22.130892 64.35.105.238.22 > MY.NET.250.239.22: tcp 0
02/02 04:52:22.131237 64.35.105.238.22 > MY.NET.250.240.22: tcp 0
        :
02/02 04:58:12.943799 64.35.105.238.1053 > MY.NET.194.150.22: tcp 0 (DF)
02/02 04:58:12.944101 MY.NET.194.150.22 > 64.35.105.238.1053: tcp 0 (DF)
        :
        :
02/02 05:02:13.842739 64.35.105.238.2335 > MY.NET.194.150.22: tcp 1448 (DF)
02/02 05:02:13.843005 64.35.105.238.2335 > MY.NET.194.150.22: tcp 1448 (DF)
```

1. Source of Trace: the author's network

2. Detect was generated by: Gus. Data herein is the output of tcpdump -nqr . Boris' ruleset of the time was not logging ssh, but it did log to Shaun a similar scan to subnet broadcast addresses and others blocked at the border (format is the same as Shaun's previous data):

```
Jan 31 04:50:28 tcp 64.35.105.238(22) -> MY.NET.0.255(22), 1 packet
Jan 31 04:50:31 tcp 64.35.105.238(22) -> MY.NET.8.40(22), 1 packet
        :
        :
Jan 31 04:52:24 tcp 64.35.105.238(22) -> MY.NET.251.255(22), 1 packet
```

```
Jan 31 04:52:26 tcp 64.35.105.238(22) -> MY.NET.254.255(22), 1 packet
```

3. Probability the source address was spoofed: high. We can see distinct differences by looking more closely at the change from reconnaissance to attack using tcpdump -nvvr:

```
02/02 04:52:22.130559 64.35.105.238.22 > MY.NET.250.238.22: S [tcp sum ok] \
182358395:182358395(0) win 43667 (ttl 114, id 52837, len 40)
02/02 04:52:22.130892 64.35.105.238.22 > MY.NET.250.239.22: S [tcp sum ok] \
182358395:182358395(0) win 43667 (ttl 114, id 52837, len 40)
02/02 04:52:22.131237 64.35.105.238.22 > MY.NET.250.240.22: S [tcp sum ok] \
182358395:182358395(0) win 43667 (ttl 114, id 52837, len 40)
02/02 04:58:12.943799 64.35.105.238.1053 > MY.NET.194.150.22: S [tcp sum ok] \
2373713988:2373713988(0) win 32120  (DF) (ttl 52, id 31773, len 60)
02/02 04:58:13.286637 64.35.105.238.1053 > MY.NET.194.150.22: P [tcp sum ok] \
2373738776:2373740224(1448) ack 466062753 win 32120  (DF) (ttl 52, id 31797, len 1500)
02/02 04:58:13.286764 64.35.105.238.1053 > MY.NET.194.150.22: P [tcp sum ok] \
1448:2896(1448) ack 1 win 32120  \
(DF) (ttl 52, id 31798, len 1500)
```

The recon packets all have an identical initial sequence number, an identical IP ID, an identical and odd window size, and they're using port 22 as source port, which is just bad behavior. The attack packets have so significantly different a value that they're almost certainly a different architecture, IP ID's begin incrementing from a different place, sequence numbers are different, and abruptly ephemeral ports are being used.

4. Description of the attack: buffer overflow attempt against vulnerable ssh.

5. Attack mechanism: we look at the data more closely with tcpdump -xnr:

```
02/02 05:02:13.843005 64.35.105.238.2335 > MY.NET.194.150.22: . \
56472:57920(1448) ack 1 win 32120  (DF)
                         4500 05dc 1729 4000 3406 3d21 4023 69ee
                         xxxx c296 091f 0016 9cd3 4a2e 29b4 e350
                         8010 7d78 84e8 0000 0101 080a 18df e34e
                         088b a714 9090 9090 [hundreds of 0x90's ...

                         ... skipped]   9090 9090 9090 9090 9090
                         9090 9090 9090 9090 9090 9090
```

This is a classic "NOP sled", indicating an attack specific to x86 architecture. Interspersed with these are:

```
02/02 05:01:46.274668 64.35.105.238.2285 > MY.NET.194.150.22: P \
5948:7396(1448) ack 13 win 32120  (DF)
                         4500 05dc 088a 4000 3406 4bc0 4023 69ee
                         xxxx c296 08ed 0016 9b90 ac2f 2835 5ed3
                         8018 7d78 a85d 0000 0101 080a 18df d886
                         088b 9c4c 7350 ffff 0000 3ed1 7350 ffff
                         0000 3ed5 7350 ffff 0000 3ed9 7350 ffff
                         [ each successive 64bits increments by 4]
                              :
                         0000 419d 7350 ffff 0000 41a1
```

I think this data pattern may be intended to overflow onto the stack, causing a function return or call to the NOP sled; alternatively, this may be something that exploits a different architecture altogether. Finally, here we can see, by using tcpdump -Xnvr, MY.NET.194.150 giving up the game:

```
02/02 04:58:13.006135 MY.NET.194.150.22 > 64.35.105.238.1053: P 1:24(23) ack 1 \
win 32120  (DF)
0x0000   4500 004b ca2c 4000 3f06 84ae xxxx c296        E..K.,@.?....*..
0x0010   4023 69ee 0016 041d 1bc7 8c6a 8d7c 0045        @#i........j.|.E
0x0020   8018 7d78 b7ea 0000 0101 080a 088b 48ee        ..}x..........H.
0x0030   18df 851d 5353 482d 312e 3939 2d4f 7065        ....SSH-1.99-Ope
0x0040   6e53 5348 5f32 2e31 2e31 0a                    nSSH_2.1.1.
```

6. Correlations: CIAC: Understanding SSH Exploits;

GIAC: Guy Bruneau GCIA Practical was very instructive about the two-host spoofing behavior.

7. Evidence of Active Targeting: clear, if not initial. The attacker scanned until he found a vulnerable host and then focused all his attention on that single host.

8. Severity:

Severity	=	(Criticality + Lethality) - (Host Countermeasures + Network Countermeasures)
	Criticality = 4	MY.NET.194.150 is a departmental mail server.
	Lethality = 5	Successful buffer overflows can gain root
	Host = 1	Admin reported later that "it's everything a Linux box shouldn't be" - old version, unpatched, etc.
	Network = 2	nIDS notified Networking via e-mail, ssh globally allowed

(4 + 5) - (1 + 2) = **6**

9. Defensive Recommendations: Reinstall the MY.NET.194.150's OS, bring it up to date on patches before placing it back on the network. Configure any instance of ssh for ssh-2 only.

10. Question: We think the following packet, rendered by tcpdump -nxr , contains an ssh version string. What's the correct syntax to show the text?

```
04:58:13.006135 MY.NET.194.150.22 > 64.35.105.238.1053: P 1:24(23) \
ack 1 win 32120  (DF)
                        4500 004b ca2c 4000 3f06 84ae xxxx c296
                        4023 69ee 0016 041d 1bc7 8c6a 8d7c 0045
                        8018 7d78 b7ea 0000 0101 080a 088b 48ee
                        18df 851d 5353 482d 312e 3939 2d4f 7065
                        6e53 5348 5f32 2e31 2e31 0a
```

- A. a.out
- B. snort -A ascii
- C. tcpdump -nar
- D. tcpdump -nXr

The correct answer is D.

Detect #3 - Telnet Host Scan, with Buffer Overflow

```
Feb  2 00:32:13 24.128.147.68:7375 -> MY.NET.5.14:80 SYN ******S*
Feb  2 00:32:13 24.128.147.68:7371 -> MY.NET.5.14:80 FIN *******F
Feb  2 00:32:13 24.128.147.68:7373 -> MY.NET.5.14:80 SYNFIN ******SF
Feb  2 00:32:13 24.128.147.68:7374 -> MY.NET.5.14:80 VECNA ****P***
Feb  2 00:33:03 24.128.147.68:6022 -> MY.NET.5.3:23 SYN ******S*
Feb  2 00:33:03 24.128.147.68:6018 -> MY.NET.5.3:23 FIN *******F
Feb  2 00:33:03 24.128.147.68:6020 -> MY.NET.5.3:23 SYNFIN ******SF
Feb  2 00:33:03 24.128.147.68:6021 -> MY.NET.5.3:23 VECNA ****P***
       :
```

pattern above repeats through MY.NET.5.0/24 as far as MY.NET.5.32, repeating the pattern SYN/FIN/SYNFIN/VECNA exactly almost every time - and *skipping* nonexistent hosts!

1. Source of Trace: the author's network

2. Detect was generated by: Gus. This data is from the portscan.log file, formatted as follows:

```
  date/time           srcIP:srcprt -> dstIP:dstprt   LBL TCPflags
Feb  2 00:32:13 24.128.147.68:7375 -> MY.NET.5.14:80 SYN ******S*
```

3. Probability the source address was spoofed: low, although we need data from the other Snort logs to determine that. The packets are clearly crafted, but we examine data from *both* Snort logs to determine so. We examine some packets that the portscan preprocessor didn't record (using tcpdump -nr):

```
02/02 00:32:37:776052 24.128.147.68.3223 > MY.NET.5.14.23: S 1681155886:1681155886(0) \
win 16384  (DF) [tos 0x10]
02/02 00:32:37.840889 24.128.147.68.3223 > MY.NET.5.14.23: . ack 1455530317 \
win 17376  (DF) [tos 0x10]
02/02 00:32:37.847408 24.128.147.68.3223 > MY.NET.5.14.23: P 0:36(36) ack 1 \
win 17376  (DF) [tos 0x10]
02/02 00:32:42.981255 24.128.147.68.3223 > MY.NET.5.14.23: . ack 2 win 17376 \
 (DF) [tos 0x10]
02/02 00:32:42.987179 24.128.147.68.3223 > MY.NET.5.14.23: F 36:36(0) ack 2 \
win 17376  (DF) [tos 0x10]
```

We only have one side of the conversation, but clearly a handshake or a notable success with blind spoofing was accomplished. The next target, MY.NET.5.3, yielded this pattern:

```
00:33:03.552277 24.128.147.68.6016 > MY.NET.5.3.23: S [tcp sum ok] 1352396425:1352396425(0) \
win 4660 (ttl 238, id 37353, len 40)
00:33:03.564269 24.128.147.68.6017 > MY.NET.5.3.23: S [tcp sum ok] 1352396425:1352396425(0) \
ack 0 win 4660 (ttl 238, id 37354, len 40)
00:33:03.581287 24.128.147.68.6018 > MY.NET.5.3.23: F [tcp sum ok] 1352396425:1352396425(0) \
win 4660 (ttl 238, id 37355, len 40)
00:33:03.621387 24.128.147.68.6020 > MY.NET.5.3.23: SF [tcp sum ok] 1352396425:1352396425(0) \
win 4660 (ttl 238, id 37357, len 40)
00:33:03.647697 24.128.147.68.6021 > MY.NET.5.3.23: P [tcp sum ok] win 4660 \
(ttl 238, id 37358, len 40)
00:33:03.669460 24.128.147.68.6022 > MY.NET.5.3.23: SWE [tcp sum ok] \
1352396425:1352396425(0) win 4660 (ttl 238, id 37359, len 40)
```

MY.NET.5.3 is a Macintosh with nothing listening on the Telnet port, and the attacker's pattern was different when no one was listening - clearly the attacker is achieving the handshake rather than blind spoofing.

We also noted this trace in the logfiles from MY.NET.5.14, indicating successful TCP_wrapper deflection of the first complete 3-way handshake at 00:32:37 :

```
Feb  2 00:32:38 www.MY.NET in.telnetd[10038]: refused connect from 24.128.147.68
```

4. Description of the attack: possible buffer overflow attempt against telnet service, leading to crash or compromise if successful.

5. Attack mechanism: the attack takes place in two phases. The first is a sequence of packets SYN-FIN-SYNFIN-VECNA(****P***); presumably the responses to these stimuli (of which we don't have a trace in this detect) allow the attacker to determine whether the host in question is potentially vulnerable. In the second phase, a three-way handshake is initiated, and telnet option negotiation done. It's not clear what the size of the buffer is, as the number of telnet options presented is not large: WILL AUTHENTICATION, DO ENCRYPT, WILL ENCRYPT, DO SUPPRESS GO AHEAD, WILL TERMINAL TYPE, WILL NAWS, WILL TSPEED, WILL LFLOW, WILL LINEMODE, WILL NEW-ENVIRON, DO STATUS, and WILL XDISPLOC .

6. Correlations:

- Cisco: Security Advisory: Cisco CatOS Telnet Buffer Vulnerability
- CERT: Advisory CA-2001-21 Buffer Overflow in telnetd.

7. Evidence of active targeting: the attacker was walking through MY.NET sequentially - but he began with a WWW query to www.MY.NET (MY.NET.5.14), then telnet queries to other machines on MY.NET subnet 5 - but only those IP's that were in use. He neatly skipped gaps in the sequence at MY.NET.5.21 and .26 .

8. Severity:

Severity = (Criticality + Lethality) - (Host Countermeasures + Network Countermeasures)

Criticality = 5 *every* live host on a critical network
Lethality = 5 crash or compromise
Host = 5 every host on this network uses TCP_wrappers, and deflected the attack
Network = 2 nIDS alerted, but telnet was allowed
(5 + 5) - (5 + 2) = **3**

9. Defensive Recommendations: Telnet needs to be blocked into network MY.NET.5.0/24 - it's supposed to be disabled on all the machines anyway.

10. Question: The invalid TCP flag combinations **U***** and ****P***, and three others, are collectively called 'VECNA' by Snort after the contributor who added detection code for them. Why are they invalid?

- A. the URGENT and PUSH flags should always appear together
- B. The URGENT and PUSH flags should always appear with SYN or FIN
- C. Without SYN, SYN/ACK, ACK, or FIN, the connection is not in a valid state
- D. The absence of SYN or ACK means the connection is being gracefully disconnected

The correct answer is C.

Detect #4 - Flood of UDP DNS

```
02/07 08:07:33.374246 203.126.246.203.775 > MY.NET.5.4.53:  [udp sum ok] 50943 \
PTR? 254.10.MYOTHER.NET.in-addr.arpa. [|domain] (ttl 114, id 24697, len 71)
02/07 08:07:39.817290 203.126.246.203.775 > MY.NET.5.4.53:  [udp sum ok] 50999 \
PTR? 254.10.MYOTHER.NET.in-addr.arpa. [|domain] (ttl 114, id 48505, len 71)
02/07 08:07:54.759223 203.126.246.203.775 > MY.NET.5.4.53:  [udp sum ok] 51027 \
PTR? 254.10.MYOTHER.NET.in-addr.arpa. [|domain] (ttl 114, id 2939, len 71)
        :
```

MYOTHER.NET.0.0/16 is a network allocated to my employer, but not visible outside of MY.NET. MY.NET.5.4 is registered as the authoritative DNS for MYOTHER.NET.0.0/16 , but not answering queries from outside of MY.NET .

Only fifteen different PTR records are ever sought that fall in the MYOTHER.NET.0.0/16 network, and of those, the overwhelming favorite is MYOTHER.NET.10.254; queries for that record approach 2,000 per hour at times, and Gus' logging suffers some level of packet loss. This traffic has been coming at a consistently high level since December 17; Shaun's logs provide corroboration of the high rate of traffic, put no packet payload data, because Boris, as a Cisco router, only logs source IP, target IP, source port, and target port to Shaun.

1. Source of Trace: the author's network

2. Detect was generated by: Gus, based on a rule that records all DNS queries.

3. Probability the source address was spoofed: low, though not zero.

203.126.246.203 is registered as cmsfw.charteredsemi.com.sg ; given that the authoritative nameserver for this domain is csmdns.charteredsemi.com.sg , I would conclude that the "fw" in the hostname is for "firewall". A firewall could reasonably be expected to perform reverse lookups on incoming source IP's.

TTL's on these packets are consistent between 111 and 114, in large groups where the value doesn't change for a long time. A traceroute to csmdns.charteredsemi.com.sg resolves in 13 hops, which seems to verify the topological distance away, assuming an initial TTL of 128. IP ID's increment in a consistent fashion for a busy remote network.

The remote host seems to use a single non-ephemeral source port for its queries for hours at a time; this could be evidence of spoofing, but more likely it's an artifact of the firewall technology in use.

It's extraordinary likely, however, that the stimulus provoking these queries is spoofed traffic from

MYOTHER.NET.10.0/24, as it's a network never allowed access past our border; either it's spoofed traffic, or someone is making use of that address space without authorization.

4. Description of the attack: copious quantities of DNS traffic.

5. Attack mechanism: I surmise that 203.126.246.203 is looking for authoritative DNS to resolve the status of the MYOTHER.NET.0.0/16 hosts in question, since it continues to query repeatly, and would cache the results once found. Presumably the stimulus evoking this response is traffic with source addresses in MYOTHER.NET.0.0/16, whether spoofed or simply unauthorized use. As a test, I created and deployed a zone file for MYOTHER.NET.in-addr.arpa in the DNS on MY.NET.5.4, making it respond to external queries as the authoritative DNS for that zone. The queries abruptly ceased:

```
Feb  7 12:44:51 DNS.MY.NET named[17522]: Ready to answer queries.

12:45:46.591239 203.126.246.203.775 > MY.NET.5.4.53:  63337 PTR? 254.10.MYOTHER.NET.in-addr.arpa. (43)
12:45:55.560151 203.126.246.203.775 > MY.NET.5.4.53:  63344 PTR? 254.10.MYOTHER.NET.in-addr.arpa. (43)
```

This confirmed my surmise, though Gus missed the responses MY.NET.5.4 sent.

6. Correlations: the O'Reilly book ***DNS and BIND***, by Paul Albitz and Cricket Liu, notes the caching behavior of DNS lookups with respect to the standard DNS time-to-live of 24 hours.

7. Evidence of active targeting: based on a review of 48 hours of snort data, during which 203.246.126.203 sent 54,190 packets, every single packet was a PTR request of this sort, to our primary DNS server.

8. Severity:

Severity = (Criticality + Lethality) - (Host Countermeasures + Network Countermeasures)

Criticality = 5 our primary nameserver
Lethality = 0 not an attack per se
Host = 4 worst side effect might be overloading the host
Network = 3 worst side effect might be overloading the network

(5 + 0) - (4 + 3) = **-2**

9. Defensive Recommendations: Depending on organizational priorities, one should either (a) monitor traffic indicating unauthorized use of its networks, for purposes of early warning; or (b) configure empty authoritative zone files for external queries, to allow DNS' caching behavior to reduce traffic.

10. Question: If the pattern of packets below was observed, (this trace is *not* the same!), what different inference could be drawn?

```
02/07 08:07:33.374246 203.126.246.203.775 > MY.NET.5.4.53:  [udp sum ok] 50943 \
PTR? 254.10.MYOTHER.NET.in-addr.arpa. [|domain] (ttl 114, id 24697, len 71)
02/07 08:07:39.817290 203.126.246.203.775 > MY.NET.5.4.53:  [udp sum ok] 50999 \
PTR? 253.10.MYOTHER.NET.in-addr.arpa. [|domain] (ttl 114, id 48505, len 71)
02/07 08:07:54.759223 203.126.246.203.775 > MY.NET.5.4.53:  [udp sum ok] 51027 \
PTR? 252.10.MYOTHER.NET.in-addr.arpa. [|domain] (ttl 114, id 2939, len 71)
        :
```

A. This is a DNS attack
B. This is reconnaissance
C. The DNS requests are spoofed
D. The packets are crafted

The correct answer is B.

Detect #5 - "ICMP Traffic Help" request from incidents.org Intrusions Mail list

N.B. I did not read any of the mailing list responses to this query before completing this analysis.

- ***Date*: Wed, 06 Feb 2002 04:33:24 +0800**
- ***From*: " Ray Nichols"**
- ***Subject*: ICMP Traffic Help**

```
Can someone help me understand these alerts? Is this an attempted DDOS? THE
first part of the message looks like an ICMP packet directed at me, but the
**ORIGINAL DATAGRAM DUMP looks like an SMTP response. I don't get it.

ICMP_UNRCH_PACKET_FILT
Red# more ICMP_UNRCH_PACKET_FILT
[**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**]
02/05-15:07:53.061214 216.54.219.98 -> x.x.x.2
ICMP TTL:242 TOS:0x0 ID:1090 IpLen:20 DgmLen:56
Type:3  Code:13  DESTINATION UNREACHABLE: PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
x.x.x.2:25 -> 152.50.232.134:2049
TCP TTL:239 TOS:0x0 ID:63363 IpLen:20 DgmLen:44
Seq: 0xA32ECF4F
** END OF DUMP
45 00 00 2C F7 83 40 00 EF 06 E0 BD xx xx xx 02  E..,..@.........
98 32 E8 86 00 19 08 01 A3 2E CF 4F              .2.........O

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

[**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**]
02/05-15:07:55.957378 216.54.219.98 -> x.x.x.2
ICMP TTL:242 TOS:0x0 ID:1091 IpLen:20 DgmLen:56
Type:3  Code:13  DESTINATION UNREACHABLE: PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
x.x.x.2:25 -> 152.50.232.134:2049
TCP TTL:239 TOS:0x0 ID:63364 IpLen:20 DgmLen:40
Seq: 0xA32ECF50
** END OF DUMP
45 00 00 28 F7 84 40 00 EF 06 E0 C0 xx xx xx 02  E..(..@.........
98 32 E8 86 00 19 08 01 A3 2E CF 50              .2.........P

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

[packet truncated]

1. Source of Trace: incidents.org mailing list.

2. Detect was generated by: Snort. The use of the delimiter "[**]", and the layout of the IP and ICMP field values, make this look like Snort data from the console or syslog, with Snort alerting in "full" mode, the default.

3. Probability the source address was spoofed: small, but not impossible. TTL's are consistent in both the ICMP packets and the "ORIGINAL DATAGRAM DUMP" packets contained therein. IP ID's appear to vary normally.

There are exactly two TCP sequence numbers in the DUMP packets, leading one to believe that these are retries - although it's odd that there are more than three of each, and that the two sequence numbers are interleaved.

4. Description of the attack: ICMP notification, probably that a router ACL has blocked the TCP traffic in the "ORIGINAL DATAGRAM DUMP"s.

5. Attack mechanism: the "ORIGINAL DATAGRAM DUMP"s look like an SMTP connection blocked through the unfortunate happenstance of having chosen port 2049 as its ephemeral source port. Since NFS is offered using 2049 for the server, many networks block traffic across their border to port 2049. One can surmise that router 216.54.219.98 is on the path between Ray Nichols' mailserver and remote host 152.50.232.134, and a traceroute to the latter seems to confirm it.

6. Correlations: note Detect #1 above regarding ICMP payload. An example of port 2049 being blocked is in the SecurityFocus FOCUS-LINUX archive.

7. Evidence of active targeting: only a handful of packets are here, so it's difficult to say. This looks like two separate sequences of TCP retries - since ICMP doesn't encapsulate the stimulus packet as far as the TCP flags, we can't tell whether they were set SYN/ACK, as they should have been.

8. Severity:

Severity	=	(Criticality + Lethality) - (Host Countermeasures + Network Countermeasures)
	Criticality = 5	presumably Internet-accessible mail server
	Lethality = 1	with some ephemeral ports blocked, SMTP and other services may suffer occasional outages
	Host = 3	unknown
	Network = 4	no specifics, but if port 2049 is blocked, networking/security personnel are somewhat aggressive

(5 + 1) - (3 + 4) = **-1**

9. Defensive Recommendations: "Communication Administratively Prohibited" messages can be used by an attacker to map one's ACL's - the maintainers of 216.54.219.80 might consider turning off such notifications.

The rule that caused this traffic appears to have blocked a legitimate TCP connection because of a conflict between the use of port 2049 as a service port and its use as an ephemeral port. The desired result could be accomplished by only blocking initial SYN packets to 2049, but allowing SYN/ACK and ACK; for example, on Cisco routers,

```
ip access-list 101 permit tcp {external net} {internal net} eq 2049 established
ip access-list 101 deny   tcp {external net} {internal net} eq 2049
```

will do this.

One should obfuscate addresses in packet payload, as well as headers, before disseminating it.

10. Question: We note from the trace above that TCP packets were blocked. What TCP flags were set in the stimulus packets that caused these ICMP notifications?

A. We can't tell, because ICMP doesn't encapsulate enough of the stimulus packet
B. We can't tell because ICMP and TCP are incompatible protocols
C. ACK and RST
D. SYN and ACK

The correct answer is A.

Assignment 3 - "Analyze This"

Table of Contents

Gentlemen,

I am more than happy to assist you with your security needs. I have received the data you sent, containing Snort logs of alerts, portscans, and out of specification traffic, as well as the pointer to previous security reports you have received.

I do wish at the outset to make my intentions fully clear, and thus your expectations similarly so. My report to you will be a high-level security analysis, rather than an in-depth security audit. To effectively use this information, your security personnel will need to expand on this information, since:

- I do not have a copy of your campus Security Policy, nor even knowledge that one exists, against which to audit;
- I will be analyzing your logs in a vacuum, as I have no knowledge of your network topology nor your sensor configuration (particularly your Snort ruleset) and placement;
- numerous compromises and infections noted here can be expected to spread before your personnel can correct them;
- you have provided me excerpts, rather than complete raw data from your sensor. I cannot fault your practice of obfuscating your network address space in potentially sensitive situations. Working from these excerpts, however, will tend to create situations where I simply do not have the data to be as complete as we both would like. Intrusion analysis invariably takes place with a paucity of hard data; it would be to your advantage to provide all data possible to any future analyst, so as to get the most accurate and useful results.

The use of Snort, while an excellent choice, raises another issue which must be clear at the outset. The standard Snort ruleset includes reference pointers for its alerts, and nearly 65% of these references point to the site www.whitehats.com (by way of the code word "arachnids"). Unfortunately, that site has been offline, for reasons of which I am unaware, for many weeks now. I was able to find substitute references in most cases, but there are three specific Snort alerts for which I could find no other reference material. I have noted these three instances in the report.

That said, there is a wealth of data from just the one Snort sensor. It is my hope that you find this information useful - please do not hesitate to contact me with any questions you may have.

Executive Summary

- Alert, Scan, and OOS (Out-Of-Specification traffic) logfiles from a single instance of Snort, covering traffic in MY.NET.0.0/16, for the period 9-22 January 2002, were submitted.
- Significant quantities of traffic were observed to and from networks MY.NET.150.0/23, MY.NET.152.0/23, MY.NET.88.0/24, and MY.NET.5.0/24 . Smaller quantities were observed relating to MY.NET.6.0/24 and MY.NET.149.0/24, and minuscule amounts from other subnets.
- 68 nominally different Snort alerts were noted, with counts ranging from over 100,000 down to 1; those alerts or logical groups of alerts with counts over 10 were analyzed in detail, using both numerical and graphical methods, and a number of probable compromises were noted.
- Lists of top talkers and top targets were collated from the Scan logs, and those nodes' traffic analyzed in depth; the results were cross-referenced with the Alert analyses above, and additional compromises and network issues noted.
- OOS files were analyzed: some interesting results were noted, but little was gleaned in the way of important security implications.
- Specific recommendations are included on a per-alert or per-node basis in the body of this report
- Overall recommendations:
    - Remove security exposures - use non-default community names, assign services to servers based on consistency of access rules, readdress managed infrastructure onto private networks using VLAN technology, protect WWW resources with smart caches, deploy and use antivirus software, shut down unauthorized servers, verify currency of patches and security configurations, configure MY.NET against spoofing.
    - Address issues by way of security policy - discontinue the use of unencrypted protocols and filesystem access across the network border, create or amend policies regarding administrative access across the border, block commonly-abused and not supported traffic at the border, develop response and recovery procedures for network attacks.
    - Address security infrastructure issues - more and larger IDS hosts, update software and ruleset, adjust ruleset for local environment as needed, remove outdated Watchlist and internal troubleshooting rules, provide auditors and consultants with proper data to review.
    - Address network infrastructure issues - determine status of RFC1918 proscribed networks, troubleshoot core network problems.
    - Address bandwidth utilization - examine MY.NET's policies regarding the use of "chat" programs, filesharing utilities such as Napster, Gnutella, KaZaA, etc., and other high-bandwidth entertainment usage, and adjust border filters as appropriate according to policy.

Terms and Abbreviations

- "eph" indicates "ephemeral ports", i.e. ports above 1023 used as source ports in TCP connections. In my usage herein, it implies such use of these ports, as opposed to use relating to certain specific applications (e.g. 6000 for X11 traffic, or 1214 for KaZaA); it also indicates likelihood of multiple ephemeral ports being used.
- "(Napster)" is used as a shorthand to mean any of the Napster ports, including but not limited to 2222,

3333, 4444, 5555, 6666, 6699, 7777, 8875, 8888, and 9999.

- "noise level" indicates a consistent level in time distribution; if a "noise level" is noted, one can expect no "surges"[q.v.] or larger variations.
- "peak", "surge", and "SPIKE"[sic], as used to describe time distribution of alert or host records, all imply a value that visibly stands out from background traffic. Time analysis is based on events per hour; a "surge" is a value up to about 100, a "peak" up to about 1,000, and a "SPIKE" multiple thousands. An example of a SPIKE is shown below, in a graph of the alert entitled connect to port 515 from inside.
- "step-decay", as used to describe time distribution, indicates a repeating pattern of high initial surges/peaks/SPIKES, each followed by a tapering off back to background traffic level. An example is shown below, in a graph of the alert ICMP Router Selection.
- "qty" - quantity.
- "widely varying" - nonstandard use of (particularly TCP ports); I use this term when nodes are using both the "well-known" and the ephemeral port range, which nonstandard practice is inherently suspicious.

Explanation of Analysis Process

The data provided was analyzed in the following environment:

Analysis host 1, "san-gabriel", is a Sun Ultra 5 running Solaris 2.8, equipped with

- gunzip 1.2.4 GNU Awk 3.0.2
- Perl 5.004_04
- Netscape Communicator 4.74 for UNIX
- GNU Emacs 19.30.1
- xv 3.10a
- openssh 2.3.0p1
- Snort 1.8.1 and a standard 1.8.1-vintage ruleset

Analysis host 2, "broccoli", is a Dell Dimsion XPS D333 running Windows 95, equipped with

- Netscape Communicator 4.75 for Windows
- Microsoft Office 97 (Word and Excel)
- F-Secure SSH Tunnel & Terminal 2.0
- SSH(c) Secure Shell 2.2.0

Steps taken:

1. on san-gabriel, placed copies of original data files in a subdirectory away from working copies, and set them to read-only to prevent accidental erasure or modification. Filtered "spp_portscan" entries from Alert logs, as they are duplication of data in Scan logs and are formatted differently than true alerts.
2. created ordered table of alerts, using commandline awk to isolate alert text and Perl script to collate.
3. wrote ad hoc script "alerttable.pl" to collate time, host, and port statistics regarding alerts, generating *.results and *.time.dist files. Added resulst to table.
4. after working with awk to reformat time.dist files, transferred them to broccoli and loaded them into Excel. Created charts to visually inspect time distribution of alerts; cut-and-pasted two by way of Microsoft Windows Paint accessory to bitmap format. Recorded time distribution in table.
5. transferred bitmaps to san-gabriel, used xv to crop and save to GIF format.
6. performed analysis on each alert, creating several groups of related alerts. Much general WWW research performed at sites:
    - www.google.com
    - www.ciac.org
    - www.mitre.org

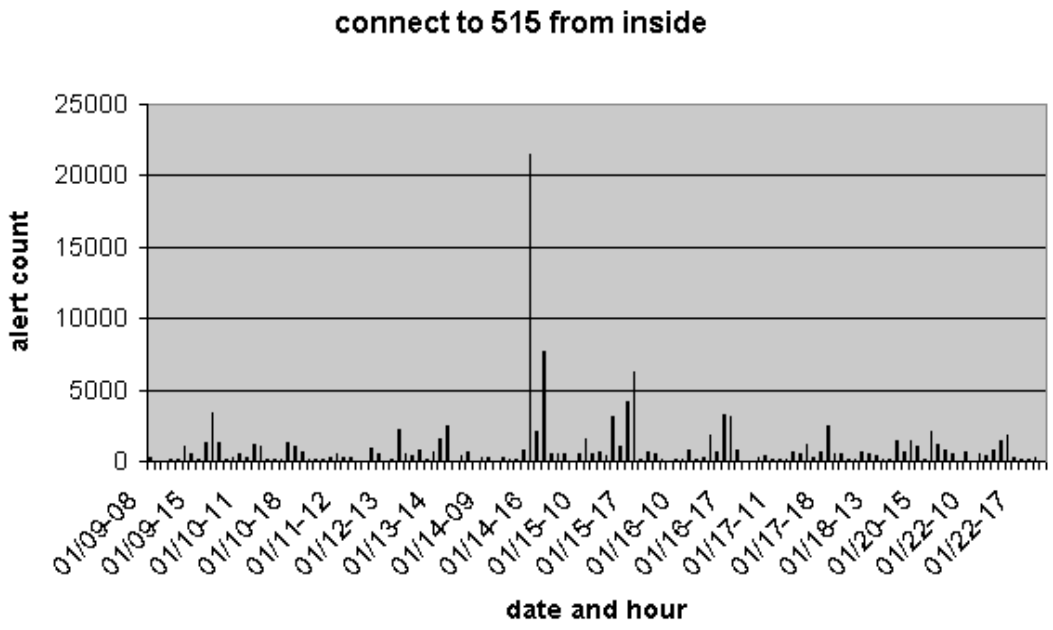
- www.sans.org
- www.securityfocus.com
- www.faqs.org
- www.insecure.org

and others too numerous to mention.

7. color-coded alerts for immediacy of recommended action.
8. created ordered lists of "source" and "target" nodes, for each permutation of source file (Alert log or Scan log) and host locus relative to MY.NET (internal or external).
9. processed top ten nodes in each list much as alerts above, with ad hoc script "toptentable.pl". Included statistic generation and visual determination of time distribution as before.
10. performed analysis of individual nodes, cross-referencing with alert analyses as necessary.
11. attempted unsuccessfully to establish precise relationship of OOS data to either Alert or Scan logs; analyzed OOS data by hand.

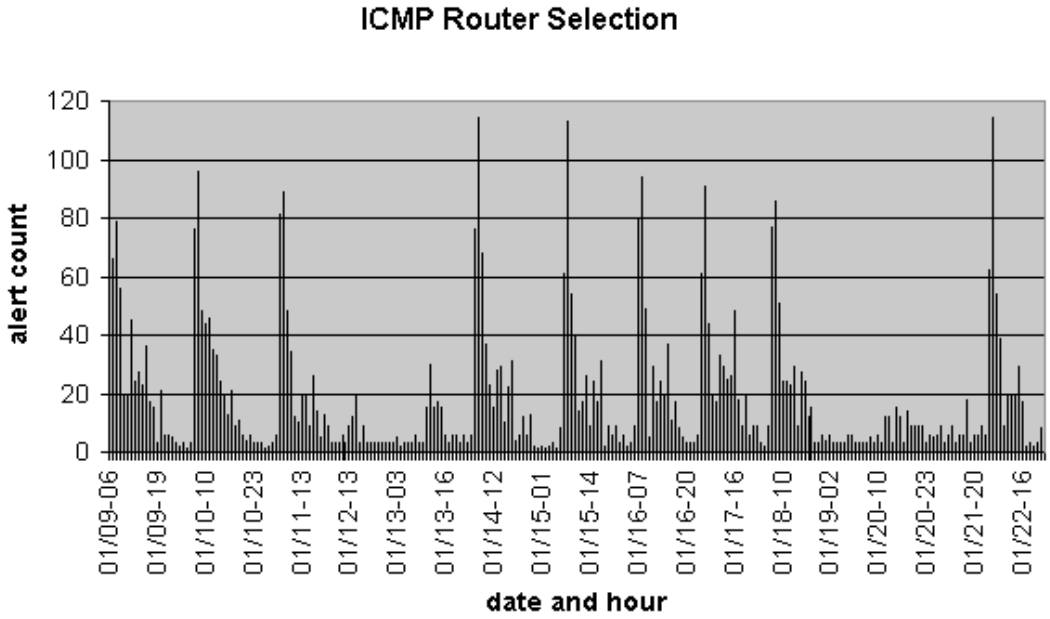
---

example of a SPIKE



---

example of step-decay



List of Files Processed

Upon unpacking your data, I found the following files - please double-check the sizes and checksums, that we may both be assured no data loss or corruption occurred.

gzip'd size	filename	raw size	md5 checksum
1,537,642	alert.020109.gz	19,245,604	fa0f2aa1380f68e3c541a8bb7ca4a417
1,547,686	alert.020110.gz	17,926,616	d9de610616dfb5c90270e8a959c2ce83
1,198,174	alert.020111.gz	14,181,717	9d4516f8a4b73349f28ba47acc88823a
660,989	alert.020112.gz	7,560,032	1ac511601d7b8c65d5ad417fdbed86f9
709,464	alert.020113.gz	8,593,576	83b7dbe66b1da2c0ff5769ed137b70ba
1,530,572	alert.020114.gz	18,724,853	85e0ec097d220406c9abe4f6e65049e1
1,565,097	alert.020115.gz	18,224,331	29787d7d2d08f136fdf10d3a56c24e0c
1,457,925	alert.020116.gz	17,016,421	01ba674cbfce9060266863b04047ccfe
1,910,592	alert.020117.gz	21,028,993	95d9493acd5319e0d932a6737d70909e
1,165,771	alert.020118.gz	14,636,230	d282d8395b6a16cf4fec89e7338617e2
417,285	alert.020119.gz	4,994,257	5f556db08219b5828336425d5ff84f29
810,800	alert.020120.gz	10,014,976	6aed55d7e9ed43f3d4e95d67c4076146
430,238	alert.020121.gz	5,324,733	f66b1b312eb6c645bd9f3cfeede746bd
1,204,035	alert.020122.gz	14,079,663	a256b6ef503cd6fdd3c6478dc8683d9e
792	oos_Jan.9.2002.gz	3,120	b7ce33fae8eb6b19d2951df1f1fe71b4
4,518	oos_Jan.10.2002.gz	92,308	8d9fa2d68d706d0cd4cb99b028c3925d
536	oos_Jan.11.2002.gz	1,369	ab3190c05636ae76b2b068d90d933f65
968	oos_Jan.12.2002.gz	8,297	1dac011d31449df8ce3f47190f299dd9
214	oos_Jan.13.2002.gz	505	2d9d14b46c1d47e2eaa5e68a45c23ea1
3,573	oos_Jan.14.2002.gz	65,889	ddf8db8fb83ccc14349c615ef89309ae
1,336	oos_Jan.15.2002.gz	6,211	285977ef1b1b028258ef03b83ad1ea5b
631	oos_Jan.16.2002.gz	1,989	760a8a8f3053f690f78e0c92619fb94d
670	oos_Jan.17.2002.gz	1,819	afbc8bbef22edf90573a58294440662b
779	oos_Jan.18.2002.gz	7,265	01ff74274214c95bf4896615b04b53a9
214	oos_Jan.19.2002.gz	505	b458d07688a6e0ce645d5a015b0b0059
4,182	oos_Jan.20.2002.gz	80,840	c41b4c0b0a1c244ceb3211ebfeeed483
1,147	oos_Jan.21.2002.gz	3,414	5dbd6d848dee98e5f3180e32132bdc8c
351	oos_Jan.22.2002.gz	780	944b20e75aeca2919db2e49e51d0ad14
2,461,458	scans.020109.gz	22,694,662	73921c26ea53743fe13a35c2222ce3df
2,552,395	scans.020110.gz	23,418,768	a1105e3bcda758719abd73218cb92878
1,797,359	scans.020111.gz	17,212,084	746d8bae64e69da8cebcead5fd8a0b22
859,670	scans.020112.gz	8,458,081	e544e2b0bc645df9ee55d52a5ae13606
1,036,060	scans.020113.gz	10,000,718	68a824ac3071ff55b6a23cda113be205
1,853,558	scans.020114.gz	17,465,551	24543e4e276b90e64ecf80b19e18bd84

```
  2,547,273 scans.020115.gz    22,581,404 f0f3e7078621f3ff0d3a7a19c1afe3a5
  2,261,896 scans.020116.gz    19,988,353 2d4262d9197ce4010fb3fafd44aa98c4
  3,287,311 scans.020117.gz    28,241,770 e38ea916c10f43f65d89d1177acaf98f
  1,628,331 scans.020118.gz    15,832,334 a3b3b1510bb3f256b000a068685eebe3
    582,135 scans.020119.gz     7,054,621 91bd64fb18597c413b4c7ad7acf5d22e
  1,175,150 scans.020120.gz    11,742,946 9b333912a1cc42be5cd35cdc9927c511
    575,201 scans.020121.gz     6,735,101 33c038a67832e86b550d1e8f81e1ae01
  1,838,481 scans.020122.gz    17,456,573 7b5a1b5db81a3d3df3eb71259f2222ef
```

Summary of Alerts

Alerts are presented by descending alert count, with statistics on host address, ports, and time distribution, which was determined by using Microsoft Excel to chart hourly alert counts and visually inspecting the results; see the examples above. (Note - not every alert presented had enough hits for time distribution to be significant).

The leftmost column is an immediacy of action indicator based on analysis results, keyed as follows:

* RED	probable **compromise**, requiring immediate securing action
* Orange	a security practice that should be corrected as soon as possible
* Yellow	a change recommendation for the long term
	no key color indicates a range of action recommendations, from policy revisions to no recommendations at all
?	question mark indicates an alert I could not evaluate fairly

It should be noted that the determination of "source" and "target" host, while often readily apparent on a per-packet basis, is less clear when applied to an ongoing flow of traffic between hosts, especially if the sensors used experience packet loss, or, as it appears in this case, the sensors only see one side of the flow.

Table I. Alerts by frequency from Alert logs

	Alert Count	Alert Description	"Source" Hosts		"Target" Hosts		"Source" Ports	"Target" Ports	Time Distribution
			Int	Ext	Int	Ext			
	119059	connect to 515 from inside	118	0	2	0	eph	515	large qty during business hours; HUGE SPIKE on 14 Jan
*	92095	spp_http_decode: IIS Unicode attack detected	117	38	10	832	eph	80, 8080	SPIKES 9, 11, 13, 16, 19 Jan
*	86273	ICMP traceroute	7	0	4	0	n/a	n/a	steady avg 275/hr; ceases 0715 20 Jan
*	63691	SNMP public access	24	0	144	0	eph	161	uniform background level; erratic peaks
									SPIKES on

*	36061	MISC Large UDP Packet	0	24	20	0	eph + 0	eph + 0	14, 17, 20 Jan
	26958	Watchlist 000220 IL-ISDNNET-990517	0	58	16	0	eph + 80, 1214	eph + 1214	SPIKES 14-16, 19, 20 Jan
	15672	INFO MSN IM Chat data	53	43	49	42	large qty 1863; eph	large qty 1863; eph	erratic peaks; SPIKE on 12 Jan
*	11614	spp_http_decode: CGI Null Byte attack detected	22	1	2	27	eph	80	two SPIKES on 9 Jan
*	9158	High port 65535 udp - possible Red Worm - traffic	67	73	151	0	large qty 65535; eph + 0, 255	large qty 65535; eph + 0, 255, 256	erratic peaks; loose tie-in with business day
	4596	ICMP Router Selection	149	0	0	1	n/a	n/a	daily step-decay each business day
*	3704	ICMP Fragment Reassembly Time Exceeded	44	0	14	69	n/a	n/a	SPIKE on 13 Jan, peak on 14 Jan
	3233	SMB Name Wildcard	89	0	100	0	137, 1883	137	varies with business day
*	2837	FTP DoS ftpd globbing	0	23	16	0	eph	21	SPIKE on 17 Jan
?	1837	ICMP Echo Request L3retriever Ping	31	0	18	0	n/a	n/a	varies with business day
	1822	INFO Inbound GNUTella Connect request	0	1195	4	0	eph	6346	peaks on 13, 17 Jan
	1453	Null scan!	1	152	17	0	widely varying; include 0, 1, 23, 194, 1214	widely varying; include 0, 1214	single peak on 22 Jan
	1327	ICMP Echo Request Windows	20	0	17	26	n/a	n/a	peaks on 10, 20 Jan
	921	SYN-FIN scan!	0	7	421	0	eph + 22, 80	eph + 22, 80	three peaks
	636	WEB-MISC Attempt to execute cmd	0	38	8	0	eph	80	erratic peaks
*	619	SCAN Proxy attempt	0	39	386	0	large qty 8080; eph	1080, 8080	single peak on 9 Jan
*	605	ICMP Destination Unreachable (Communication	1	0	1	0	n/a	n/a	noise level up to 10/hr

		Administratively Prohibited)							up to 10/hr
*	462	ICMP Echo Request Nmap or HPING2	8	0	3	11	n/a	n/a	peak on 20 Jan
	409	ICMP Echo Request CyberKit 2.2 Windows	10	0	0	11	n/a	n/a	peak on 15 Jan
	396	INFO Outbound GNUTella Connect request	3	0	0	116	eph	1080, 6346, 6347, 6373	peak on 17 Jan
	382	INFO FTP anonymous FTP	0	33	22	0	eph	21	erratic surges
	256	INFO Inbound GNUTella Connect accept	4	0	0	213	6346	eph	surge on 17 Jan
	249	MISC traceroute	0	8	4	0	eph	80, 1214	single peak on 11 Jan
	213	INFO - ICQ Access	2	0	0	19	eph	80	surges on 9, 11 Jan
	196	ICMP Echo Request BSDtype	4	0	9	0	n/a	n/a	surges on 12, 17 Jan
	182	WEB-IIS view source via translate header	0	8	3	0	eph	80	isolated surges
	170	INFO Possible IRC Access	18	0	0	16	eph	6667-6669, 7000	surge on 19 Jan
*	162	TCP SRC and DST outside network	0	19	0	21	eph + 80, 139, 1214	eph + 139	erratic
	149	ICMP Destination Unreachable (Host Unreachable)	2	0	55	0	n/a	n/a	surge on 22 Jan
	118	NMAP TCP ping!	0	22	8	0	80, 1755	1214, 4403, 9620	noise level up to 4/hr
	110	IDS552/web-iis_IIS ISAPI Overflow ida nosize	0	99	10	0	eph	80	noise level up to 4/hr
*	109	EXPLOIT NTPDX buffer overflow	0	26	14	0	eph + 0, 123	123	surges on 9, 20 Jan
	105	EXPLOIT x86 NOOP	0	20	23	0	eph + 20, 80	eph + 20	surge on 10 Jan
*	95	Possible trojan server activity	11	4	11	4	eph + 1214, 27374	eph + 1214, 27374	erratic surges
	82	Incomplete Packet Fragments Discarded	0	7	4	0	0 (n/a?)	0 (n/a?)	surge on 22 Jan
	74	ICMP Destination Unreachable (Protocol Unreachable)	4	0	1	5	n/a	n/a	surge on 22 Jan
	70	Watchlist 000222 NET-NCFC	0	4	4	0	80, 1664	eph + 1214	surge on 9 Jan, little

		NOPC						1214	else
	68	WEB-IIS _vti_inf access	0	24	2	0	eph	80	
	63	SCAN Synscan Portscan ID 19104	0	62	7	0	eph	1214, 9876	
	62	WEB-MISC compaq nsight directory traversal	0	18	16	0	80	2301	
	61	WEB-FRONTPAGE _vti_rpc access	0	21	2	0	eph	80	
	47	EXPLOIT x86 setuid 0	0	23	18	0	eph + 20, 80, 82, 84	eph + 1214	
	45	WEB-MISC 403 Forbidden	4	0	0	19	80	eph	
	41	INFO Napster Client Data	5	5	3	12	(Napster) + eph	(Napster) + eph	
*	39	INFO - Possible Squid Scan	0	5	15	0	eph	3128	
	37	WEB-MISC http directory traversal	0	2	2	0	eph	80	surge on 18 Jan
*	36	Back Orifice	7	0	22	0	eph + 14	31337	
*	30	Attempted Sun RPC high port access	5	1	18	0	widely varying	32771	
	29	WEB-COLDFUSION administrator access	0	2	1	0	eph	80	surges 9,10 Jan
	25	SCAN FIN	0	11	4	0	eph + 137	eph + 53, 137, 1214, 7001	
*	22	Port 55850 tcp - Possible myserver activity - ref. 010313-1	2	7	2	7	1214, 2313, 55850	1214, 2313, 55850	
	21	EXPLOIT x86 setgid 0	0	18	10	0	eph + 20, 80	eph	
	17	Virus - Possible MyRomeo Worm	0	1	1	0	110	eph	surge on 17 Jan
	16	INFO Outbound GNUTella Connect accept	0	9	4	0	6346	eph	
	14	Queso fingerprint	0	6	4	0	high	1214, 6346, 25197	
	12	x86 NOOP - unicode BUFFER OVERFLOW ATTACK	0	3	3	0	80, 1054	eph + 20	
	9	EXPLOIT x86 stealth noop	0	7	6	0	eph + 80	eph + 20	
	6	WEB-IIS Unauthorized IP Access Attempt	1	0	0	1	80	eph	
*	5	SUNRPC highport access!	0	1	1	0	80	32771	

	3	Probable NMAP fingerprint attempt	0	2	2	0	high	high
	1	Virus - Possible Simbiosis Worm	0	1	1	0	110	eph
*	1	Port 55850 udp - Possible myserver activity - ref. 010313-1	1	0	1	0	32780	55850
	1	MISC Large ICMP Packet	0	1	1	0	n/a	n/a
*	1	High port 65535 tcp - possible Red Worm - traffic	0	1	1	0	65535	65535

Analysis of Alerts

Analysis is presented on the traffic for each alert by name. Where alerts are closely related, they are grouped together and analyzed as a body; when this sort of grouping occurs, a hyperlink is inserted to preserve the ordering of the alerts by frequency.

Any alert or logical group of alerts that had 10 or fewer hits in the two-week analysis period is not presented here.

1. connect to 515 from inside

Port 515 is the IANA Registered Port for the "printer" service, most commonly implemented with an lpd daemon. All of the traffic triggering this alert is between internal MY.NET hosts, and some 99.99% (all but eight packets) is destined for host MY.NET.150.198 , which host's only other significant alerts are SNMP Public Access traffic.

99.5% of this traffic comes from network range MY.NET.152.0/23 . The huge flow of traffic on January 14th, does not seem to indicate anything other than heavy printing use; hosts MY.NET.153.164 and MY.NET.153.146 together account for over 90% of the traffic, but these hosts triggered no other alerts at all.

My reference Snort ruleset has specific port 515 exploit rules, but no rule like this one - it appears that this rule is intended to catch internal hosts looking outside, which behavior could be an indication of compromised internal hosts.

Conclusions:

- MY.NET.150.198 is a server running lpd, serving network range MY.NET.152.0/23 and possibly others.
- This is a local rule intended either for informational logging only, or to catch internal machines running port 515 exploits across the border.
- January 14th or 15th may have been a project due date for users of MY.NET.153.146 and MY.NET.153.164. Alternatively, these two hosts may be misconfigured or broken.

Recommendations:

- A widely-available print server and an NMS host, which contains community names and crucial network information, are not a good combination in a single host; rules of access for the two servers are not particularly compatible. Separate these functions.
- Logging in this way creates a high level in your Snort instance, which could interfere with IDS function; either a "log" rule with reporting based on tcpdump or snort replays on packet logs, or a

"tag" rule would probably improve your IDS signal-to-noise ratio. Alternatively, tweaking the ruleset to alert only on cross-border traffic might meet your needs better. Suggestions:

```
# log to snort-xxxx@xxxx files, but not alert facility
log tcp $HOME_NET any -> $HOME_NET 515
alert tcp $HOME_NET any -> any 515 (msg:"connect to 515 from inside";)

# log initiation of lpr sessions, rather than all such traffic
alert tcp $HOME_NET any <> any 515 (flags: S+; \
    tag:session,5,packets; msg:"connect to 515 from inside";

# remember to run snort with -o ! pass intra-campus traffic
pass tcp $HOME_NET any -> $HOME_NET 515
alert tcp $HOME_NET any -> any 515 (msg:"connect to 515 from inside";)

# alert only on internal hosts looking for lpd outside
alert tcp $HOME_NET any -> !$HOME_NET 515 (msg:"connect to 515 from inside";)
```

References:

- SecurityFocus: [Bugtraq ID 1712] an example vulnerability on port 515
- CVE: the same vulnerability

2. IIS Unicode attack detected

Unicode is a unifying standard for encoding character data, intended to resolve conflicts between characters sets (such as ASCII), and reduce corruption of data. It is supported by most WWW browsers and clients.

Unicode encoding, however, has come to the fore as a means for evasion of both network and host-based intrusion detection. By using Unicode to encode portions of a URL sent to an IIS server, an attacker can avoid triggering IDS alert rules and also bypass IIS' onboard controls, thus allowing an attacker access to data on the IIS server that he ought not to have.

For example, the request:

```
GET /scripts/..\../winnt/system32/cmd.exe?/c+dir
```

would be denied out of hand by an IIS server; however, by using the Unicode encoding %5c instead of the backslash:

```
GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir
```

one finds that an IIS server will honor the request. This particular example would allow an attacker a directory listing of *any directory* on the server accessible to the account under which IIS is run.

This technique is used most commonly by the Nimda worm and its variants, and it indicates there is a good deal of Nimda or related exploit traffic in your network.

Conclusions:

- Nimda infections or related compromises are widespread in networks MY.NET.150.0/23, MY.NET.152.0/23, and MY.NET.88.0/24. MY.NET.16.42 also appears affected.
- Numerous hosts external to your network are sending this traffic as well, as they have been since Nimda first appeared on 18 September 2001 - it seems unavoidable that Nimda and CodeRed variants are now a permanent part of the noise level on the Internet.

Recommendations:

- A massive disinfection effort is called for on the affected internal networks. 117 hosts appear to be sources of this traffic in varying amounts.
- Consider amending your policy regarding WWW servers, such that you can assert control of who can run a server. Out-of-the-box installations of IIS account for much of the Nimda presence on the Internet.
- Consider deploying a transparent WWW cache that includes filtering capability. Unicode decoding is computationally taxing for most firewalls, but a cache is typically optimized for that feature.

References:

- CVE: Unicode encoding possibly allows arbitrary commands
- CIAC: Nimda (The W32.nimda worm)

3. ICMP traceroute

A traceroute of any sort is an attempt to learn your network topology. Of the packets observed that triggered this alert, 99.7% (all but twenty-eight of them), were from MY.NET.5.202 to MY.NET.5.1 ; if your network is subnetted at /24, as many Class B sized networks are, this makes no sense at all. The traffic ran at a uniformly high level throughout most of the analyzed time period, and abruptly ceased before 8:00 am on January 20th.

Conclusions:

- Something is clearly awry with MY.NET.5.202, although it seems likely that it is something broken or a network test gone awry.

Recommendations:

- Examine MY.NET.5.202 for signs of misconfiguration, failure, or compromise. Determine whether a user or maintainer of that system was running some sort of test or experiment.

References:

- Analysis of an ICMP traceroute by Roland Gerlach

4. SNMP public access

Simple Network Management Protocol, or SNMP, is a lightweight protocol intended for monitoring and updating networked hosts, particularly network infrastructure and networked printers. SNMP access is authorized by use of textual "community names" embedded in an SNMP request packet; "public" is an extraordinarily common default community name for read-only access.

Using the default community name is a bad security practice, as it gives anyone who can transmit SNMP to your managed infrastructure free access.

Such traffic could be legitimate NMS (network monitoring, e.g. by Spectrum, HP Openview, MRTG, etc.), or printer management, but the number of hosts sending this traffic, and the random distribution about your network, argues against this.

MY.NET.70.177 is sending far and away most of the requests. MY.NET.150.198 is clearly a print server, as noted here, but it is not clear what the other hosts are.

The good news here is that none of this traffic crosses your border.

Conclusions:

- Host MY.NET.70.177 appears to be a network monitoring station.

- Numerous devices are being managed by way of SNMP with the default community name. It is not clear whether all of this is authorized use or not.
- SNMP is blocked at your border (as it should be).

Recommendations:

- Change the community names on the target devices at once.
- Consider a move to later versions of SNMP that do not send community names in clear text.
- If the devices managed via SNMP are network infrastructure, I strongly recommend you consider retooling your network using switched VLAN technology, and addressing your infrastructure on a network inaccessible to your user community and the Internet at large.

References:

- RFC1067 A Simple Network Management Protocol

5. MISC Large UDP Packet
67. MISC Large ICMP Packet

Large packets are part of the method of several different attacks. The traffic patterns here, though at first face seemingly chaotic, resolve into patterns resembling downloads upon analysis:

Packet Count	Time Duration	"Source"	"Target"
Top 5			
2900	00:05:10	64.124.157.58:56704 (domain: akamaitechnologies.com)	MY.NET.153.45:2599
2647	00:04:41	64.124.157.48:10216 (domain: akamaitechnologies.com)	MY.NET.153.45:1558
1686	00:03:49	218.2.4.101:4272 (unregistered host in China)	MY.NET.153.142:1500
1539	00:03:16	211.172.232.21:2106 (unregistered host in Korea)	MY.NET.153.144:1992
1507	00:03:53	211.233.45.41:1057 (unregistered host in Korea)	MY.NET.153.106:3535
suspicious			
1154	n/a	multiple external on port 0	multiple internal
36	n/a	6 hosts in Korea, 1 in United Kingdom, "well-known" ports	8 hosts in MY.NET.153.0/24, random ports

I have seen this pattern in my own university network, mostly from dormitory, public lab, and dialup pool networks. It seems likely that the bulk of this traffic consists of filesharing transactions, similar to those of Gnutella or KaZaA, but UDP-based and port-agile. The top four external sources of this traffic are content caching hosts; the next three are unregistered hosts in Korea.

The UDP port zero traffic is troubling. The bulk of it, some 92%, comes from Korean networks within 211.233.0.0/16 - in fact, from the same network as the three hosts mentioned above. The port 0 traffic represents 13% of the total traffic from the Korean networks, and on a per-host basis sometimes exceeds 50%.

The "well-known" port (ports below 1024) usage represents 0.1% of total traffic regarding this rule, and seems insignificant.

Conclusions:

- A great deal of file downloading appears to be underway by the users of MY.NET.150.0/23, MY.NET.152.0/23, and host MY.NET.88.132.
- A nontrivial amount of traffic on UDP port 0 is crossing your border.
- Something is causing malformed traffic from 211.233.0.0/16.

Recommendations:

- If your policy prohibits the use of filesharing software, then you have several policy violators using your network. However, as the technology being used is port-agile, it is not clear what can be done technologically to stop it.
- block UDP source and target port 0 at your border.

References:

- CVE: an example of susceptibility to port 0 traffic

6. Watchlist 000220 IL-ISDNNET-990517
41. Watchlist 000222 NET-NCFC

No rule of the form "Watchlist..." exists in my reference instance of Snort. It appears that these rules are logging traffic from specific networks. "IL-ISDNNET-990517" is registered at whois.ripe.net, as follows:

```
inetnum:      212.179.0.0 - 212.179.255.255
netname:      IL-ISDNNET-990517
descr:        PROVIDER
country:      IL
admin-c:      NP469-RIPE
tech-c:       TP1233-RIPE
tech-c:       ZV140-RIPE
tech-c:       ES4966-RIPE
status:       ALLOCATED PA
mnt-by:       RIPE-NCC-HM-MNT
changed:      hostmaster@ripe.net 19990517
changed:      hostmaster@ripe.net 20000406
changed:      hostmaster@ripe.net 20010402
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

person:       Tomer Peer
address:      Bezeq International
address:      40 Hashakham St.
address:      Petakh Tiqwah  Israel
phone:        +972 3 9257761
e-mail:       hostmaster@isdn.net.il
nic-hdl:      TP1233-RIPE
changed:      registrar@ns.il 19991113
source:       RIPE

person:       Zehavit Vigder
address:      bezeq-international
```

```
address:        40 hashacham
address:        petach tikva 49170 Israel
phone:          +972 52 770145
fax-no:         +972 9 8940763
e-mail:         hostmaster@bezeqint.net
nic-hdl:        ZV140-RIPE
changed:        zehavitv@bezeqint.net 20000528
source:         RIPE

person:         Eran Shchori
address:        BEZEQ INTERNATIONAL
address:        40 Hashacham Street
address:        Petach-Tikva 49170 Israel
phone:          +972 3 9257710
fax-no:         +972 3 9257726
e-mail:         hostmaster@bezeqint.net
nic-hdl:        ES4966-RIPE
changed:        registrar@ns.il 20000309
source:         RIPE
```

and NET-NCFC at whois.arin.net as:

```
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
   CN

   Netname: NCFC
   Netblock: 159.226.0.0 - 159.226.255.255

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

Specifically, traffic was logged from multiple sources in 212.179.0.0/17 and 159.226.0.0/16 to multiple destinations in networks MY.NET.150.0/24, MY.NET.152.0/23, and MY.NET.5.0/24, as well as host MY.NET.88.162.

Virtually all of it appears to be KaZaA (port 1214) filesharing traffic in both directions, or WWW browsing by MY.NET hosts of nine remote hosts (none of which appear in the DNS as WWW servers; this may be a filesharing technology again, masquerading as HTTP - I would need to see raw packet logs to confirm this).

Conclusions:

- A local rule was configured to watch traffic from 212.179.0.0/17 and 159.226.0.0/16, presumably following some bad behavior from those networks. It is not clear how long ago this may have been, or whether this monitoring is still appropriate.
- None of the current traffic appears to be a security threat per se, although filesharing software may not be in accord with your policies.

Recommendations:

- Consider repealing these rules, or amending them to log-only rules.
- If the KaZaA traffic is a problem, do *not* block port 1214; the software is designed to notice if it cannot communicate on its default port at all, and change ports. A much more effective strategy is to rate-limit it to an arbitrarily low value.

7. INFO MSN IM Chat data

"IM" is "Instant Messenger", a development pioneered by AOL, and now distributed as MSN

Messenger. Users that are running Messenger, as the name implies, can send real-time text messages back and forth between their computers without the overhead of e-mail.

All of the traffic triggering this rule is between external network 64.4.12.128/26 and internal networks MY.NET.150.0/24, MY.NET.152.0/23, and MY.NET.88.0/24. The external network is registered to "MS HOTMAIL"; presumably Messenger is relayed through the external network, rather than point-to-point, for efficiency.

My reference Snort ruleset classifies this traffic as "not-suspicious".

Conclusions:

- A large amount of Messaging is going on among your users.

Recommendations:

- If the use of Messenger is against your policies, a block on port 1863 in both directions at your border would stop it .

8. CGI Null Byte attack detected

The mechanism is different, but, like the Unicode attack noted earlier, this is a means for an attacker to have arbitrary access to a WWW server.

Virtually all of this traffic is flowing from MY.NET hosts to external hosts. This is a strong indication of compromise or malicious behavior by the internal hosts.

Conclusions:

- Numerous hosts in networks MY.NET.88.0/24, MY.NET.150.0/24, and MY.NET.152.0/23 appear to be under malicious control or broken.

Recommendations:

- Audit and secure the affected internal networks. 22 hosts appear at present to be sources of this traffic in varying amounts.

References:

- CVE: an example of a CGI Null byte attack
- An analysis of a CGI Null attack by Roy Naldo (will require unzipping)

9. High port 65535 udp - possible Red Worm - traffic
68. High port 65535 tcp - possible Red Worm - traffic

The "Red Worm" (not to be confused with "Code Red") is a Linux-specific infection, typified by a backdoor listening on port 65535 (which otherwise should not be used). Traffic seen from this port strongly indicates a compromised host.

Such traffic is widespread and high in volume in network MY.NET.6.0/24; networks MY.NET.5.0/24, MY.NET.88.0/24, MY.NET.149.0/24, and MY.NET.152.0/23 are showing such traffic, albeit in smaller volume, as is host MY.NET.60.43. Numerous external hosts, naturally, exhibit this behavior as well.

Conclusions:

- Numerous hosts in MY.NET, as noted above, may be infected with the Red Worm.

Recommendations:

- A massive disinfection effort is necessary for the 150 internal hosts affected.

- Port 65535 should be blocked at the border.

References:

- F-Secure: Description of Red Worm / Adore
- SANS: Adore Worm analysis

10. ICMP Router Selection

The single target address 224.0.0.2 is the reserved multicast "all routers" address. Although the rule generating this alert seems to have been retired from my reference Snort ruleset, I have found references to the rule:

```
alert icmp any any -> any any (msg:"ICMP Router Selection"; itype: 10; icode: 0;)
```

This traffic all appears to be coming from networks MY.NET.88.0/24, MY.NET.150.0/23, MY.NET.152.0/23, and one host MY.NET.5.141 . As such traffic is an attempt to find a navigable path outside the subnet, it is possible that it is indicative of a misconfiguration. However, no reference I can find indicates this is hostile traffic.

Conclusions:

- Numerous hosts on the networks above are dynamically searching for routes out of those networks. It is not clear whether this is a problem to be addressed or a deliberate design choice.

Recommendations: None

References:

- NLANR: Ping Examples including one to 224.0.0.2 / all-routers.
- RFC1256 ICMP Router Discovery Messages

11. ICMP Fragment Reassembly Time Exceeded

This alert records ICMP type 11, code 1 packets, which are notifications from a host that packets received by that host are fragmented and not being completed before the timeout for reassembly. This rule is not present in my reference Snort ruleset, leading me to conclude that it is been removed as informational and not necessary (cf. Router Selection).

81% of this traffic is destined for trans-Pacific hosts, i.e. most of it seems to indicate problems with fragmentation of traffic from those hosts to your network. All of the sources are in networks MY.NET.6.0/24, MY.NET.88.0/24, MY.NET.150.0/23, and MY.NET.152.0/23 .

A small quantum of traffic seems to indicate an internal problem with traffic between networks MY.NET.6.0/24 and MY.NET.152.0/23 .

Conclusions:

- A small level of fragmentation and packet loss is occurring between your internal networks, as noted above.

- Some fragmentation and packet loss are occurring on trans-Pacific traffic to your network.

Recommendation:

- Troubleshoot the network path between MY.NET.5.0/24 and MY.NET.152.0/23 .

12. SMB Name Wildcard

This alert is triggered by Windows machines polling each other for names and type of resources to

share; the "wildcard" aspect indicates that the machine initiating the poll wants to know all of them (as the command "nbtstat -a").

All traffic triggering this alert appears to have both source and target within MY.NET, which is as it should be; as a general principle, I recommend blocking SMB traffic at the network border.

92% of this traffic is destined for hosts on network MY.NET.5.0/24, with 37% flowing to host MY.NET.5.4 alone. Sources vary widely across MY.NET . It should be noted that this is UDP 137 traffic, and the sensor appears to be catching both sides of the numerous conversations, except in the case of MY.NET.5.4:

```
01/09-00:00:59.256778  [**] SMB Name Wildcard [**] MY.NET.5.7:137 -> MY.NET.5.87:137
01/09-00:00:59.257013  [**] SMB Name Wildcard [**] MY.NET.5.87:137 -> MY.NET.5.7:137
01/09-00:01:06.055526  [**] SMB Name Wildcard [**] MY.NET.5.7:137 -> MY.NET.5.87:137
01/09-00:01:06.055871  [**] SMB Name Wildcard [**] MY.NET.5.87:137 -> MY.NET.5.7:137

01/09-07:04:29.601436  [**] SMB Name Wildcard [**] MY.NET.150.127:137 -> MY.NET.151.190:137
01/09-07:04:29.601529  [**] SMB Name Wildcard [**] MY.NET.150.127:137 -> MY.NET.151.190:137
01/09-07:04:29.601906  [**] SMB Name Wildcard [**] MY.NET.151.190:137 -> MY.NET.150.127:137
01/09-07:04:29.602171  [**] SMB Name Wildcard [**] MY.NET.151.190:137 -> MY.NET.150.127:137

01/09-09:36:03.732325  [**] SMB Name Wildcard [**] MY.NET.153.158:137 -> MY.NET.5.4:137
01/09-09:36:05.239264  [**] SMB Name Wildcard [**] MY.NET.153.158:137 -> MY.NET.5.4:137
01/09-09:36:06.759318  [**] SMB Name Wildcard [**] MY.NET.153.158:137 -> MY.NET.5.4:137
```

Such traffic, while it could be indicative of Nimda infection or other bad behavior if the traffic pattern was different, is part of normal Windows networking. This rule is not present in my reference Snort ruleset.

Conclusions:

- There is a fair amount of Windows networking present in MY.NET .
- SMB traffic is blocked at the border.
- MY.NET.5.0/24 is a core network providing centralized Windows resources.
- The Snort sensor may be topologically located where it can see traffic to much of MY.NET.5.0/24, but not that to MY.NET.5.4, suggesting that the sensor is somewhere on network MY.NET.5.0/24 itself.

Recommendations: None

13. FTP DoS ftpd globbing

FTP is one of the least secure protocols that can be run. Even without the numerous vulnerabilities associated with various ftp servers, it transmits userids, passwords, and all data in cleartext and in chunks of moderate size (cf. telnet, where a malicious listener would need to reassemble a session to get a password - with ftp, one packet gives the game away).

Many ftp servers have vulnerabilities associated with globbing, that is, matching against directory paths and filenames with wildcards. The packets triggering this alert probably constitute attempts to exploit these vulnerabilities. All traffic logged here crossed the border from outside into MY.NET , flowing exclusively to networks MY.NET.150.0/23 and MY.NET.152.0/23 .

A host receiving these packets is not of necessity compromised; however, particularly if the packets are in large quantity, as is the case with MY.NET.153.141 (which received 46% of all traffic triggering this alert), it is worth looking for other signs of compromise. There is a strong correlation between the target hosts for this alert and those exhibiting the Unicode attack and Red Worm alerts.

Conclusions:

- A number of hosts in MY.NET.150.0/23 and MY.NET.152/23 were the objects of attempted compromise.
- MY.NET.153.141's influx of this traffic is a strong suggestion of compromise.

Recommendations:

- An audit and cleanup is in order for the 16 hosts on these internal networks.
- Serious thought should be given to blocking ftp ports 20 and 21 at the border. HTTP is a reasonable substitute where unauthenticated file distribution is the intent, although uploads are a problem. Where authentication is necessary, encrypting methods such as sftp or scp should be used.

References:

- DShield: Discussion of ftp's level of exposure
- CERT: Multiple vulnerabilities [one of which is globbing] in WU-FTPD
- SecurityPortal: Solaris ftpd glob vulnerability

14. ICMP Echo Request L3retriever Ping

The only reference to this alert I have been able to find points to www.whitehats.com , which as I have noted is currently unavailable.

The traffic pattern is entirely made up of internal hosts, and roughly correlates with the hosts triggering the SMB Name Wildcard alert.

My reference Snort rule for this alert gives no information why this type of ping might be a problem.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP L3retriever Ping"; \
   content: "ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI"; itype: 8; icode: 0; \
   depth: 32; reference:arachnids,311; classtype:attempted-recon; \
   sid:466; rev:1;)
```

Conclusions:

- Although I can provide no reference to support this view, the traffic pattern suggests that this is normal and related to Windows networking

Recommendations:

- Until such time as further reference information becomes available, traffic triggering this alert should be evaluated on a continuous basis for patterns suggesting malicious intent.

15. INFO Inbound GNUTella Connect request
24. INFO Outbound GNUTella Connect request
26. INFO Inbound GNUTella Connect accept
58. INFO Outbound GNUTella Connect accept

Gnutella is an open source filesharing program which, by default, uses port 6346.

The hosts MY.NET.152.247, MY.NET.153.153, MY.NET.153.203, and MY.NET.153.210 all appear to be functioning as Gnutella nodes, and communicating with some number of external nodes.

Interestingly enough, of 1195 external hosts making a total of 1822 connection requests, only nine hosts and sixteen requests were accepted. By comparison, 396 requests were made outbound to 116 external hosts, and 256 acceptances were transmitted from 213 external hosts. It seems anomalous that more hosts answered than were queried, but I do not know the internal details of the function of a Gnutella network.

Conclusions:

- You are successfully rate-limiting Gnutella connection attempts from outside of your border.
- Despite this, users persist in running the software on four internal nodes.

Recommendations:

- As noted above, the method of rate-limiting vs. blocking, which I have concluded is in use here, also works for KaZaA, which seems to be much more the software of choice in your network. If you have policy support to control Gnutella this way, it would seem to apply to KaZaA as well. If you are not rate-limiting Gnutella, you might consider it before it becomes more popular among your users

16. Null scan!

A Null scan is when a TCP packet is sent with none of the TCP flags set, which condition is invalid. The intention is reconnaissance which might be missed by an IDS, but this technique does not fool Snort.

In 60% of the instances noted, the Null scan packets were fewer than 15 in number of the entire two-week analysis period. It may be of interest that three of the source addresses were in 192.168.0.0/16; without any documentation of MY.NET's topology, or packet logs from which to examine MAC addresses, I cannot determine if this is appropriate or not.

The single exception was a 617-packet, 21-minute slew of Null scan packets from 217.80.164.95, directed entirely at host MY.NET.88.162; of these packets, 576 of them were crafted or corrupted with source and target port 0, and the remainder appear to have had random port values. This host is a DSL connection in Germany, pD950A45F.dip.t-dialin.net, and it has been my experience with this ISP that they handle abuse reports by sending registered mail to the customer whose node was the source, and doing little else.

Conclusions:

- Traffic from 217.80.164.95 is badly broken, either at the source or somewhere along the path to MY.NET .
- Traffic from 192.168.0.0/16 is present in MY.NET .

Recommendations:

- Consider blocking or closely monitoring further traffic from 217.80.164.95, or notifying Deutsche Telekom of this issue, if such notification is supported by your policies.
- It is not clear what might be the result of the traffic sent to MY.NET.88.162; a security audit might be in order.
- Review whether the 192.168.0.0/16 traffic is legitimate. Add it to your ingress filters if not, and $HOME_NET in your Snort configuration if so.

References:

- Charles Hutson points out that Null scans can be done with nmap, and are especially effective with the -D(ecoy) option set. Fortunately, the user of 217.80.164.95 did not know this, and probably is not reading these papers.

17. ICMP Echo Request Windows

This rule triggers on nothing more nor less than Windows machines sending pings to other hosts.

Over 94% of this traffic had its source in the three nodes MY.NET.169.222, MY.NET.5.87, and

MY.NET.150.247 . 81% went to the two nodes MY.NET.5.96 and MY.NET.70.219 .

This rule appears to be informational only, and thus is not found in my reference Snort ruleset.

Conclusions:

- MY.NET.169.222 performed 11 minutes of "ping -t" (equivalent to UNIX "ping -s") against MY.NET.150.6 ; MY.NET.5.87 similarly transmitted to MY.NET.70.219 for about 11 minutes. MY.NET.150.247 performed much testing of connectivity to 2 MY.NET hosts and 18 external ones.
- No such pings were recorded from external nodes.

Recommendations: None

18. SYN-FIN scan!

A SYN-FIN scan is a TCP traffic pattern in which both the SYN and FIN bits are set in the TCP packet header. This is a technique designed to confuse IDS systems, but Snort detects it out of hand.

Three hosts performed host scans on port 22 (ssh), uniformly across MY.NET . The hosts were 130.161.249.59, 68.52.151.18, and 195.42.223.134, and this was the only ssh traffic across the border logged during the two-week analysis period.

Two scans of port 80 (http) of much smaller magnitude were logged as well.

Conclusions:

- The three hosts performing the ssh scan were looking for vulnerable hosts in MY.NET ; the SYN-FIN aspect of the scan only confirms the crafting of the packets, which was made obvious anyway by the use of source port 22.
- The same is true of the two hosts performing the http scan.
- As the distribution of traffic from the scanning hosts was uniform across MY.NET, there is no indication that the perpetrators found any hosts to exploit.

Recommendations:

- If such notification is supported by your policies, you might notify the appropriate contacts of the traffic coming from their hosts 130.161.249.59, 68.52.141.18, and 195.42.223.134.
- It seems likely that these three were not the only ssh scans during this period, as my experience of a university network has been several ssh scans per day for the last several months. Your Snort ruleset should be modified to catch initial ssh connections (and others that might be exploited), and your logs should be regularly reviewed for widespread scans. I use the rule:

```
alert tcp any any <> any 22 (flags:S+; sid:1940005; rev:2; \
   tag:session,15,packets; msg:"SSH connection";)
```

References:

- Guy Bruneau does an excellent analysis of a SYN-FIN scan, demonstrating with traces from the wild how SYN-FIN is used in vulnerability scans.

19. WEB-MISC Attempt to execute cmd

The rule triggered here is:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 \
```

```
(msg:"WEB-MISC Attempt to execute cmd"; flags: A+; \
content:"cmd.exe"; nocase;)
```

This style of attempt is characteristic of the Nimda worm.

The sources of these attacks are all external, and they are directed at a total of eight MY.NET nodes, seven of them on MY.NET.5.0/24 . A disproportionate amount (76%) of this traffic seems targeted at MY.NET.5.96, MY.NET.150.83, and MY.NET.5.95; there are no logs here to show whether these attempts were successful (Such alerts immediately followed by the target host initiating a TFTP request would indicate Nimda-susceptibility, at the very least).

Conclusions:

- MY.NET.5.0/24 and MY.NET.150.83 are popular targets for WWW-based attacks.
- There is no other data to support it, but the obvious active targeting raises concern that some of these internal hosts might be responding to these attacks.

Recommendations:

- A cursory security check of all target hosts is appropriate.

20. SCAN Proxy attempt
49. INFO - Possible Squid Scan

These alerts are indicative that the source is looking for a WWW ls -l proxy to use, presumably to obfuscate his real source from the WWW servers he intends to browse or attack.

Over half of this traffic had its source in Italy, from host 217.57.19.30. This provider, both in my personal experience and based on discussions with other security professionals, turns a deaf ear to abuse complaints. Of the remainder, a significant amount seems to be coming from hosts in network 216.152.64.0/24.

Examing the target addresses, one finds that over 120 such requests, from about twenty-five different hosts, were directed at MY.NET.151.79 .

Conclusions:

- MY.NET.151.79 may well be running one or more proxy servers.

Recommendations:

- If MY.NET.151.79 is not an authorized proxy server, take steps to shut it down and audit as necessary.
- If you are not providing a proxy server for use by Internet users, block ports 1080, 3128, and 8080 at the border.
- Complaints to the Italian provider are probably not worth the trouble, but you might contact WebMaster, Incorporated regarding 216.152.64.0/24.

21. ICMP Destination Unreachable (Communication Administratively Prohibited)

The labeling here is fairly self-evident; the source host, which is a router or a firewall, is notifying the target host that the traffic he is sending is not allowed, and thus is blocked by an access control list.

About ten times an hour for the entire two-week analysis period, node MY.NET.150.1 apparently notified node MY.NET.150.24 that the traffic he had sent was not permitted.

Without the packet logs, I cannot determine what MY.NET.150.24 was trying to do.

Conclusions:

- MY.NET.150.24 is trying to do something that is not allowed, and just does not understand "no". This may be something broken, or something administratively blocked of which a user has not yet learned. MY.NET.150.1 is likely the default gateway for MY.NET.150.24, so the user is confined to his LAN for the purposes of whatever traffic caused these ICMP responses.

Recommendations:

- Examine the packet logs for the traffic that MY.NET.150.24 was sending (which will be partially replicated in the ICMP packets). Troubleshoot as necessary.

22. ICMP Echo Request Nmap or HPING2

Nmap and hping2 are powerful tools that can allow network analysis by both attackers and defenders.

Generally speaking, if traffic from either is crossing your border, it may be a bad thing. If traffic from either has its source in a machine that is not being used for system administration (and sometimes even if it is), it may also be a bad thing.

No hosts external to MY.NET appear to be sending such traffic. However, a variety of hosts in MY.NET.150.0/23 are sending such traffic to MY.NET.153.220 in high volume, and in small quantity to MY.NET.1.3 and MY.NET.150.197 .

Host MY.NET.153.149 has sent nearly half of this traffic, all of it to a variety of external hosts; he paid a great deal of attention (well over half his traffic) to external host 150.26.230.138 .

Conclusions:

- A group of nodes on MY.NET.150.0/23 may be running one or the other of these tools.
- MY.NET.153.149 appears to be running one or the other of these tools, and targeting external hosts. It may be compromised, especially as it is triggering the Red Worm alert as well.

Recommendations:

- Check the legitimacy of the activity from MY.NET.150.0/23 .
- Audit MY.NET.153.149 with all dispatch.

References:

- SANS Reading Room: Erik J. Kamerling on using HPing2 for anonymous reconnissance
- Joseph Rach has performed an exhaustive analysis of an Nmap scan.

23. ICMP Echo Request CyberKit 2.2 Windows

CyberKit is a Windows program allowing the use of utilities like ping, traceroute, whois, finger, et.al., from the Windows GUI interface. It also includes a PortScanner client that appears to function much like Nmap.

All of the traffic triggering this alert came from internal MY.NET hosts and targeted external hosts: 95% of the packets targeted two Yahoo DNS servers in network 204.71.200.0/24 , and the remainder went to a handful of servers in Yahoo networks in 216.136.128.0/17 .

All of the sources were in networks MY.NET.88.0/24, MY.NET.150.0/23, and MY.NET.153.0/24 . 82% had its source in node MY.NET.151.105, which had the equivalent of a "ping -s" running against the two Yahoo DNS servers, generating the peak observed on January 15th.

Conclusions:

- No data is present to indicate that this was outside of normal ICMP Echo Request usage; the targeting was not widespread enough for this to be reconnaissance.

Recommendations: None

24. INFO Outbound GNUTella Connect Request

25. INFO FTP anonymous FTP

Anonymous ftp is an older means of distributing files, and occasionally soliciting uploads. HTTP is an appropriate replacement, as noted earlier.

All traffic observed was from external sources, and targeted servers in networks MY.NET.5.0/24, MY.NET.88.0/24, MY.NET.150.0/23 and MY.NET.153.0/24, with fairly even distribution.

Conclusions:

- A fair amount of anonymous ftp is being served by MY.NET hosts.

Recommendations:

- As noted earlier, I would consider a policy block ftp at the border, as secure methods of accomplishing the same tasks are available.

26. INFO Inbound GNUTella Connect accept

27. MISC traceroute

A traceroute is a tool for learning network topology. However, the packets triggering this rule do not look like traceroutes, as they are uniformly targeted at ports 1214 and 80, KaZaA and WWW respectively (if they are TCP), to four hosts on network MY.NET.150.0/24 .

It has already been shown that much KaZaA is flowing around MY.NET ; it is not clear why it triggered this rule. It may be that the TTL's were low in value from the handful of source nodes, especially 192.168.0.2 . As noted earlier, I do not know whether that traffic crossed the border, or it is a legitimate internal network; as it is proscribed by RFC1918, it is not clear whether this traffic should be present or not.

Conclusions:

- This rule may not be working properly.
- Traffic with a source address of 192.168.0.2 is present in MY.NET .

Recommendations:

- Review the packet logs to determine whether this rule is functioning correctly.
- Review whether the 192.168.0.2 traffic is legitimate; add it to your ingress filters if not, and to $HOME_NET in your Snort configuration if so.

References:

- freesoft.org: TraceRoute from "Connected: An Internet Encyclopedia

28. INFO - ICQ Access

ICQ is a chat system, similar to Instant Messenger, provided through America Online's networks (judging from the external targets to which it is being sent).

There are exactly two sources being noted by this sensor, MY.NET.5.239 and MY.NET.151.79 .

My reference Snort ruleset classifies this traffic as "not-suspicious".

Conclusions:

- A small amount of ICQ chatting is going on among your users.

Recommendations:

- If the use of ICQ is against your policies, enforcement may be difficult. ICQ uses TCP port 80, requiring you to deploy a WWW proxy or similar filtering device with content inspection.

29. ICMP Echo Request BSDtype

Here we have another ICMP Echo request rule, with no illuminating information (at least until www.whitehats.com returns).

Distribution of this traffic tells little. MY.NET.6.7 sent over half the traffic, most of that to MY.NET.5.74 . MY.NET.5.38 sent traffic exclusively to MY.NET.5.11 . No other sources or targets were noted more than six times. All sources and targets were internal to MY.NET; no BSDtype pings crossed the border.

Conclusions:

- MY.NET.6.7, MY.NET.5.38, MY.NET.5.32, and MY.NET.5.200 may be BSD hosts.
- No data is present to indicate this traffic is anything out of the ordinary; distribution is not sufficient for this to be reconnaissance.

Recommendations: none.

30. WEB-IIS view source via translate header

This alert triggers on the exploit code for an IIS vulnerability that causes the server to send the source code, rather than its output, from an ASA or ASP page-generating installation.

This vulnerability was new to me. I am very grateful to Murray Goldschmidt for the analysis he provided.

The traffic in question came from a variety of external sources, but 97% of it targeted MY.NET.5.96 .

Conclusions:

- MY.NET.5.96 was targeted by multiple nodes for the source code of its page-generating scripts.

Recommendations:

- Verify that patches on MY.NET.5.96 are up-to-date.

31. INFO Possible IRC Access

IRC is another chat program. Although my reference Snort ruleset classifies this traffic as "not-suspicious", there are distributed DDoS and other abuse tools that use IRC as a control mechanism.

The traffic triggering this alert was fairly evenly distributed among MY.NET.88.0/24, MY.NET.150.0/23, and MY.NET.152.0/23 . The leading target was host 207.68.167.253, an unregistered msn.com host.

Conclusions:

- A small amount of IRC chatting is going on among your users.
- A disproportionate amount is targeted at an unregistered msn.com host.

Recommendations:

- If IRC is against your policies, a block on ports 6666-7000 in both directions at the border should take care of much of it. IRC clients are user-configurable for port, however.
- Examine the traffic to 207.68.167.253 more closely in the packet logs.

References:

- mIRC is a windows-based IRC client; its homepage has much useful information and links.

32. TCP SRC and DST outside network

All of the traffic matching this rule fits three categories:

1. 169.254.0.0/16 source addresses, which indicate traffic from hosts unable to successfully negotiate a DHCP lease;
2. 192.168.0.0/16 source addresses, about which there is not enough information to make a determination;
3. 172.128.0.0/10 source addresses, which are registered to aol.com .

   Conclusions:

   - Either hosts that should not be are trying to get DHCP leases, or a DHCP server or servers in MY.NET are not functioning correctly.
   - Either ingress or other filtering is not working correctly, or some 192.168.0.0/16 networks are part of MY.NET .
   - Some hosts in MY.NET are spoofing aol.com source addresses.
   - MY.NET is not configured to be proof against spoofed traffic.

   Recommendations:

   - Verify the correct performance of any MY.NET DHCP servers.
   - Review whether the 192.168.0.0/16 traffic is legitimate, and add it to your ingress filters if not, and $HOME_NET in your Snort configuration if so.
   - Examine the packet logs to determine the source of the aol.com traffic. Hypotheses: residential network users who dial into AOL while connected to the campus network; users who connect to a campus dialup pool, while connected to a broadband service.
   - Spoof-proof MY.NET, as far as possible. On leaf networks especially, no traffic should be exiting a network unless it has a source address to which the route passes to or through that network. If you use Cisco routers, a single per-interface configuration command, "ip verify unicast reverse-path", will do this.

   33. ICMP Destination Unreachable (Host Unreachable)

   The labeling here is fairly self-evident; a router is telling various hosts that their packets cannot find the host to which it is destined.

   All but one packet had its source in host MY.NET.150.1; all target addresses are in MY.NET.88.0/24, MY.NET.150.0/23, and MY.NET.152.0/23 .

   Conclusions:

- ICMP is correctly advising some MY.NET hosts that their traffic cannot find its target host.

Recommendations: none

34. NMAP TCP ping!
64. Probable NMAP fingerprint attempt

Nmap uses a TCP packet that has both the ACK flag set and an acknowledgement number of zero, which should never happen. Such traffic is not guaranteed to be nmap in action, but a large quantity from a small number of source hosts should be taken as a strong indicator.

Nmap also crafts packets with abnormal combinations of flags, in order to perform OS detection.

There are further oddities in the traffic triggering this rule. Virtually all (97%) has a "Source" port of 80 and a "Target" port of 1214; given the use of those two ports, WWW and KaZaA services respectively, it seems wrong for them to be on opposite ends of the same connections with such consistency.

External host 193.144.127.9, registered to rediris.es in Spain, is the source of 65% of this traffic, with the remainder scattered among 21 other hosts. MY.NET.150.133 received 81% of this traffic.

Conclusions:

- HOST MY.NET.150.133 was probably probed with Nmap, by multiple external hosts.

Recommendations:

- A security audit of MY.NET.150.133 is in order.

35. IDS552/web-iis_IIS ISAPI Overflow ida nosize

Although Code Red has been somewhat eclipsed by Nimda, it still is and likely will remain a part of the background noise of the Internet.

This alert is most commonly triggered by instances of Code Red looking to infect new hosts.

A large variety of external hosts scanned MY.NET; the targets were host MY.NET.150.83 and nine hosts on MY.NET.5.0/24 . No such activity was recorded with a source inside MY.NET .

Conclusions:

- Code Red is still active.
- No MY.NET hosts appear to have Code Red infections.

Recommendations:

- Make sure policies and procedures are in place such that no IIS servers are deployed unpatched, ever. This background noise will infect machines that are not patched.

References:

- CIAC: The Code Red Worm

36. EXPLOIT NTPDX buffer overflow

The rule triggered by this traffic is:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 123 \
   (msg:"EXPLOIT ntpdx overflow attempt"; dsize: >128; \
```

```
reference:arachnids,492; classtype:attempted-admin; sid:312; rev:1;)
```

This vulnerability was new to me. I am very grateful to Miika Turkia for this analysis and exploit source code.

All sources of this traffic were external, and the targets were all on networks MY.NET.88.0/24, MY.NET.150.0/23, and MY.NET.150.0/24 . Every single targeted host except MY.NET.88.183 and MY.NET.150.145 also triggered the Red Worm alert; the former was subjected to the ftpd globbing attack, and the latter seems to be a compromised Windows machine, triggering the two spp_http_decode rules.

Conclusions:

- Every target host has a reasonable probability of being compromised.

Recommendations:

- A disinfection effort is in order for all 16 target hosts.

References:

- SecurityFocus: [BugTraq] ntpd =< 4.0.99k remote buffer overflow

37. EXPLOIT x86 NOOP
46. EXPLOIT x86 setuid 0
56. EXPLOIT x86 setgid 0
60. x86 NOOP - unicode BUFFER OVERFLOW ATTACK
61. EXPLOIT x86 stealth noop

I am indebted to David Oborn's excellent analysis of the x86 NOOP rule and exploit.

This is a prime example of the need for complete packet logs. As Mr. Oborn's analysis shows, a block of character 0x90 (a "NOOP sled") can be part of an exploit attempt, but without examining the entire packet contents, I cannot be certain what exactly this was.

This is even more true of the other rules, whose patterns of binary machine language code are much shorter and thus even more statistically prone to false positives.

The external source addresses range from ISP's to educational institutions all over the globe, including one host that appears to be a WWW server for American Express. All target addresses are in networks MY.NET.88.9/24, MY.NET.150.0/24 and MY.NET.152.0/23 . "Source" port 80 dominates, with "target" port 20 a distant second in traffic volume.

Conclusions:

- Crafted exploit traffic may have come across the MY.NET border.

Recommendations:

- An examination of the complete packet logs is necessary to confirm or deny the accuracy of this detect.

38. Possible trojan server activity

This rule is triggering exclusively on the use of port 27374, one of the three most popular ports on which Trojans listen. with

A little over 20% of the traffic triggering this rule is also using port 1214, which leads to the possible explanation of 27374 having been chosen as a random ephemeral port. None of the other ports used, however, are recognizeable as a particular service, so there is a good chance that the 11 MY.NET hosts triggering this rule are in fact compromised.

Conclusions:

- A number of MY.NET hosts may be under the control of Trojan programs.

Recommendations:

- An audit and disinfection effort is in order for hosts MY.NET.150.133, MY.NET.150.220, MY.NET.185.28, and eight hosts on network MY.NET.5.0/24 .
- The loss of a single ephemeral port is a good tradeoff. Block port 27374 in both directions at the border.

References:

- SANS Reading Room: SubSeven 2.2: New Flavor of an Old Favorite

39. Incomplete Packet Fragments Discarded

This message is actually not from an alert rule, but the defrag preprocessor module of Snort. As an optimization for high speed and low overhead, the fragment reassembly preprocessor will throw away any fragmented packet that is not at least half full when the last fragment arrives.

Dragos Ruiu, the author of this module, noted in the snort-users mailing list that:

```
"This can be caused by:
- transmission errors
- broken stacks
- and fragmentation attacks"
```

One might expect a correlation between these alerts and the Fragment Reassembly Time Exceeded alert, but such correlation does not exist - though some of the source hosts are trans-Pacific in each case, there is no overlap whatsoever, and this alert is also triggered by a few hosts in Europe and one in Canada. This rule is *not* triggered, as the other is, by any internal MY.NET traffic.

These alerts might be indicative of fragmentation attacks, particularly by 211.172.232.21 upon MY.NET.153.144, which session accounts for for 84% of the traffic and times out 69 incomplete packets in less than three seconds.

Conclusions:

- Host 211.172.232.21 may have perpetrated a fragmentation attack against MY.NET.153.144.
- Although my reference instance of Snort has the source code to this module, it is not clear to me whether ICMP Time Exceeded packets cause this module to silently discard fragmented traffic. That would be a possible explanation for the seeming discrepancy between TimEx packets and those triggering this message.

Recommendations:

- Host MY.NET.153.144 merits a security check, following the attentions of the remote host.

40. ICMP Destination Unreachable (Protocol Unreachable)

The labeling here is fairly self-evident; the source host is notifying the target host that he does not speak the protocol being sent. A trivial example seen in my network is an SNMP-manageable UPS that does not speak TCP.

This alert was triggered by four MY.NET sources, sending to one MY.NET target MY.NET.115.133, and five external hosts, varying from broadband ISP's to the Army National Guard.

Without the packet logs, I cannot determine what protocols were being sent but not honored. It is not unlikely that protocol scans were perpetrated, but I have no way to confirm or deny.

Conclusions:

- Several external hosts, and host MY.NET.115.133, may have perpetrated protocol scans against four other MY.NET hosts.

Recommendations:

- A security and functionality check of MY.NET.115.133 is in order.

41. Watchlist 000222 NET-NCFC

42. WEB-IIS _vti_inf access
45. WEB-FRONTPAGE _vti_rpc access

These are IIS vulnerabilities, specific to FrontPage. I had difficulty finding amplifying information about these attacks.

All sources triggering this alert were external hosts, and all sent sparse numbers of packets. MY.NET.5.96 and MY.NET.150.83 were the only targets.

Conclusions:

- The two MY.NET hosts were repeatedly scanned for the FrontPage vulnerabilities.

Recommendations:

- A security check of the two affected hosts is in order.
- A review of patchlevels on FrontPage-equipped servers is in order.

43. SCAN Synscan Portscan ID 19104

This alert is triggered by the rule:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any \
   (msg:"SCAN Synscan Portscan ID 19104"; id: 19104; \
   flags: S; reference: arachnids,521;)
```

With whitehats.com down, I cannot find any reference that explains why the IP ID of the packet would matter on the SYN.

A cursory examination of my own Snort instance turned up about the same frequency of packets triggering this alert (in fact, one of the the packets came from my Snort host itself!). Statistically, there are only 64k values for the IP ID, so you would expect a particular one to appear from time to time.

Oddly, all traffic triggering this rule appears to have come from outside of MY.NET, to 7 hosts on MY.NET.88.0/24, MY.NET.150.0/24, and MY.NET.152.0/24 ; 80% was targeted at MY.NET.150.133 . Only one source, however, sent even two packets, so the 52 hits on MY.NET.150.133 came from 51 separate external hosts (or were spoofed). There was no apparent significance to the time distribution.

My reference Snort ruleset contains no trace of this rule. I would venture to say it has been

retired.

Conclusions:

- MY.NET.150.133 may have been a target for this type of scan.

Recommendations:

- A security check on MY.NET.150.133 would be appropriate.

44. WEB-MISC compaq nsight directory traversal

This is a vulnerability in various Compaq management software packages for their hardware. A WWW server listens on port 2301 for WWW traffic, but the WWW server is trivially vulnerable to directory traversal.

All of the observed traffic had a "source" port of 80 and a "target" port of 2301, thus triggering the rule. My reference Snort ruleset has the tag 'content: "../";' but it is revision 2 of the rule, so it is possible that your rule simply triggers on the use of port 2301.

Conclusions:

- This could be exploit attempts, or a false positive. I cannot be certain without the packet logs.

Recommendations:

- A security check of the 16 MY.NET hosts would be a appropriate.
- Any vulnerable Compaq platforms should be patches from the file SSRT0612U_im_upd06991.tar at www.service.digital.com/patches .

References:

- SecurityFocus: [BugTraq ID 282]: Compaq Management Agents Web File Access Vulnerability

45. WEB-FRONTPAGE _vti_rpc access

46. EXPLOIT x86 setuid 0

47. WEB-MISC 403 Forbidden
62. WEB-IIS Unauthorized IP Access Attempt

These rules note when a web server serves up a notice to a user browsing it, that he is forbidden the page he has requested.

This rule can augment, but not substitute for, good WWW logs, although it has the advantage of not being on the WWW server should it be compromised.

Most of the traffic noted here was from external WWW servers to four nodes on MY.NET.5.0/24 that were forbidden pages they sought. Four external nodes attempted a total of six forbidden requests of MY.NET.5.92 .

Conclusions:

- Your Snort logging is complementing your WWW server logging.

Recommendations: None

48. INFO Napster Client Data

Napster is arguably the original distributed filesharing technology. It has fallen out of favor because of fierce competition in its market and the company's legal troubles.

A variety of external hosts and a variety of MY.NET.150.0/23 and MY.NET.152.0/23 hosts appear to be making sparse use of Napster.

Conclusions:

- Napster still has a user community - Napster traffic is crossing MY.NET's border in both directions.

Recommendations:

- If your policies support blocking Napster, it takes blocking all of the ports specified above, and it may also to block their allocated IP address space.

49. INFO - Possible Squid Scan

50. WEB-MISC http directory traversal

This is a generic rule to guard MY.NET's internal WWW servers, by my reference Snort ruleset - it only triggers on target address $HTTP_SERVERS . The only such traffic was targeted at MY.NET.5.95 and MY.NET.5.96 , from two external hosts.

Conclusions:

- Two external hosts appear to be trying directory traversal exploits on MY.NET's two WWW servers, using embedded "../" sequences.

Recommendations:

- Verify up-to-date patching on MY.NET's WWW servers.

51. Back Orifice

Back Orifice, written by the Cult of the Dead Cow, is a remote control program hawked by cDc as a "remote administration tool". That discussion is far beyond the scope of this report.

Back Orifice listens on port 31337 (or "eleet" in the jargon used), for remote control traffic.

It should be noted that Back Orifice 2000 is the current version, and is port-agile. Newer versions of the Snort ruleset include a preprocessor for proper detection.

Seven hosts on network MY.NET.6.0/24 appear to be sending BO traffic to 22 hosts on network MY.NET.152.0/23 .

Conclusions:

- The hosts on MY.NET.152.0/23 may be under the remote control of the hosts on MY.NET.6.0/24; the latter hosts may be otherwise under unfriendly control.

Recommendations:

- An disinfection effort is in order on network MY.NET.6.0/24 *first*, then MY.NET.152.0/23 .
- Block port 31337 in both directions at the border.

52. Attempted Sun RPC high port access
63. SUNRPC highport access!

Sun RPC commonly has a portmapper listening on port 111, and RPC services of various sorts listening on ports from 32771-34000. These services are often exploitable.

4 hosts on MY.NET.6.0/24 and one external host triggered this alert, sending traffic on port 32771 to 18 different hosts in MY.NET.152.0/23 , and to MY.NET.88.136. Packet craft was evident, as many of the source ports were "well-known" (i.e. less than 1024) ports that are not commonly used.

Conclusions:

- Numerous MY.NET.152.0/23 hosts were reachable from on and off-campus on the RPC high ports (or at least port 32771).

Recommendations:

- An examination of all MY.NET hosts involved is in order, to determine whether the use was legitimate or exploit.
- Block ports 32771-34000 at the border.

References:

- CVE: one of the many RPC service vulnerabilities

53. WEB-COLDFUSION administrator access

A self-explanatory rule, triggered when administrator access is attempted to a Cold Fusion WWW server.

Two hosts were the source of this traffic, one an @Home broadband connection, the other one wiredforlife5.spyral.net . All of it was targeted at MY.NET.5.92 .

Conclusions:

- Either crack attempts are being made on MY.NET.5.92, or its administrator or administrators are performing their tasks in an unsecure manner.

Recommendations:

- Verify whether the two connections in question were in fact legitimate administrators.
- Establish guidelines for remote administration tasks. Suggestion: under no circumstances should any task requiring password authentication, particularly administrative tasks, be performed through an unencrypted network connection.

References:

54. SCAN FIN

A FIN scan is one in which only the FIN TCP flag is set, an invalid combination. Presumably this is intended as a way of evading an IDS, but Snort catches it by name.

A total of 11 unique external hosts perpetrated FIN scans against MY.NET.88.162, MY.NET.150.133, MY.NET.150.145, and MY.NET.152.46 , on a variety of destination ports.

Conclusions:

- FIN scans were successfully perpetrated across the MY.NET border.
- without complete packet data, it is not clear whether the scans succeeded in any exploit behavior, or even accomplished successful reconnaissance, if that was the aim.

Recommendations:

- A security check of the four MY.NET targets would be a appropriate.
- It may be possible to block FIN scans at the MY.NET border, but the side effects are not known to me even if it is, so I hesitate to recommend it as a technique.

References:

- insecure.org: Remote OS detection via TCP/IP Stack Fingerprinting by Fyodor

55. Port 55850 tcp - Possible myserver activity - ref. 010313-1
66. Port 55850 udp - Possible myserver activity - ref. 010313-1

MyServer is a DDoS agent under some variant or variants of UNIX, that dates from summer 2000. Port 55850 traffic is the remote control back door.

Over half of the traffic thus logged appears to be KaZaA traffic with 55850 as the ephemeral port. However, much of the remainder is not to known ports, and thus is suspicious.

Hosts MY.NET.150.133, MY.NET.150.136, and MY.NET.153.207 are affected .

Conclusions:

- The three MY.NET hosts may be compromised.

Recommendations:

- A security audit of these three machines would be appropriate.
- Block port 55850 in both directions at the border.

56. EXPLOIT x86 setgid 0

57. Virus - Possible MyRomeo Worm
65. Virus - Possible Simbiosis Worm

These alerts trigger not on the SMTP delivery of infected mail, but on the POP3 (TCP port 110) download of same.

From the external source, it appears the user is (or users area) logging into the YahooMail WWW-based mail front end. From a security standpoint, this is a bad practice, both because the user effective surrenders custody of his password to YahooMail, and because POP3 traffic is unencrypted.

Conclusions:

- MY.NET users are using POP3 to download mail across the border.
- At least one user has received an e-mail virus through this channel.

Recommendations:

- As noted with respect to WWW server administration, consideration should be given to prohibiting unencrypted password transmissions across the border.
- If the user can be identified from the logs on mail server MY.NET.88.165, he should be proactively contacted for disinfection of his machine or machines.

References:

- F-Secure: Description of BleBla AKA Romeo

58. INFO Outbound GNUTella Connect accept

59. Queso fingerprint

Queso is an OS fingerprinting tool which operates exlusively by way of packets with crafted TCP flags and acknowledgement numbers, according to its Documentation.txt file. As with Nmap, it is not something you want coming across your border from parties unknown.

Although in small quantity, Queso is coming across the MY.NET border from parties unknown, and mostly on other continents. Four MY.NET hosts are targets: MY.NET.88.162, MY.NET.150.133, MY.NET.152.180, and MY.NET.150.133.

Conclusions:

- Various parties unknown may be OS fingerprinting MY.NET hosts, although some of the traffic may be simply corrupted.

Recommendations:

- Security and patchlevel checks would be appropriate on the affected hosts, as successful reconnaissance appears to have occurred.

60. x86 NOOP - unicode BUFFER OVERFLOW ATTACK

61. EXPLOIT x86 stealth noop

62. WEB-IIS Unauthorized IP Access Attempt

63. SUNRPC highport access!

64. Probable NMAP fingerprint attempt

65. Virus - Possible Simbiosis Worm

66. Port 55850 udp - Possible myserver activity - ref. 010313-1

67. MISC Large ICMP Packet

68. High port 65535 tcp - possible Red Worm - traffic

Top Talkers and Targets

Nodes are presented sorted by descending hit count, with statistics; time distribution was by Microsoft Excel charting and visual inspection, as above.

The color indicator is keyed as follows:

* RED	probable COMPROMISE of this node, requiring immediate securing action
* Orange	a node engaging in a poor security practice, which practice should be corrected as soon as possible
* Yellow	a node engaging in a poor security practice which need be addressed at the policy level, or a node whose traffic pattern is just as likely to be network problems rather than malicious intent or practice
	no key color indicates that the node simply is passing a great deal of traffic

Table II. Top Ten "Source" Addresses, from Alert logs

Within MY.NET								
	IP	Talked To			Packets	Port Distribution		Time Distribution
		Internal	External	Total		"Source"	"Target"	
*	MY.NET.70.177	30	0	30	31585	4 eph + 137	137, 161	uniform background level; daily SPIKES
*	MY.NET.153.164	3	10	13	24686	eph	80, 515, 65535	SPIKE on 14 Jan
*	MY.NET.153.196	2	5	7	12656	eph	80, 515, 6699, 65535, (ICMP)	SPIKES on 15-16 Jan
*	MY.NET.153.113	1	67	68	11154	eph	80, 515, 1863, (ICMP)	erratic bursts
*	MY.NET.153.119	1	47	48	10830	eph	80, 515, 1863, 8080, (ICMP)	weekday afternoon peaks
*	MY.NET.88.183	3	9	12	9437	eph + 137	80, 137, (ICMP)	SPIKES on 9, 18 Jan
*	MY.NET.153.124	2	53	55	8915	eph	80, 515, (ICMP)	SPIKE on 18 Jan
*	MY.NET.153.146	2	29	31	8490	eph + 137	80, 137, 515	SPIKE on 14 Jan
*	MY.NET.153.117	1	86	87	8187	eph	80, 515, 8080, (ICMP)	SPIKES on 11, 16 Jan
*	MY.NET.150.198	101	0	101	7856	eph; huge qty 1025	161	uniform background level; daily SPIKES
External hosts								
	212.179.35.118	5	0	5	13175	80, 1214	eph	SPIKES on 14, 15, 20 Jan
	63.250.213.99	2	0	2	9565	0, eph	0, 123, eph	SPIKE on 17 Jan
	212.179.48.2	3	0	3	7859	eph	1214, 9620	SPIKE on 18 Jan
	64.124.157.48	1	0	1	3620	eph	eph? large qty 1558, 3317	SPIKE on 20 Jan
	212.179.35.8	1	0	1	3259	80	eph	SPIKE on 17 Jan
	64.124.157.58	1	0	1	2900	56704	2599	SPIKE on 20 Jan

	63.250.213.98	1	0	1	2724	eph? (include 7000, 7001)	eph? (include 7000, 7001)	SPIKE on 17 Jan
*	211.233.70.161	1	0	1	2391	widely varying; large qty 0, 1618, 2135	widely varying; large qty 0, 1450, 1518	SPIKE on 14 Jan
*	211.233.45.39	1	0	1	1870	widely varying; large qty 0, 4441	widely varying; large qty 0. 1583	SPIKE on 14 Jan
*	211.233.45.41	1	0	1	1770	widely varying; large qty 0, 1057	widely varying; large qty 0, 3535	SPIKE on 9 Jan

Table III. Top Ten "Target" Addresses, from Alert logs

Within MY.NET								
	IP	Talked To			Packets	Port Distribution		Time Distribution
		Internal	External	Total		"Source"	"Target"	
*	MY.NET.150.198	117	4	121	119055	eph	22, 515, 8080	SPIKE on 14 Jan
*	MY.NET.152.109	4	0	4	14272	eph	161	mostly uniform
	MY.NET.151.63	1	9	10	12302	eph + 0, 22, 7000, 7001, (ICMP)	eph + 0, 22, 7000, 7001, (ICMP)	SPIKE on 17 Jan
*	MY.NET.153.164	4	10	14	8452	large qty 1214; eph + 1, 22, 8080	large qty 3793; 22, 1214, 8080	SPIKE on 14 Jan
*	MY.NET.153.211	3	10	13	8083	eph + 22, 80, 8080	large qty 1214; eph + 0, 22, 8080	SPIKE on 18 Jan
*	MY.NET.153.45	0	33	33	7186	eph + 22, 32, 69 123, 8080	eph + 0, 22, 69, 8080	SPIKE on 20 Jan
*	MY.NET.5.96	7	72	79	6603	large qty 1064-1070, (ICMP); eph + 22, 80, 137,	large qty 80, 161, (ICMP); 22, 137, 8080	uniform background level; SPIKE on 20 Jan
*	MY.NET.153.219	8	16	24	5530	eph + 22	21, 22, 161,	SURGE overnight

							1080	14-15 Jan
*	MY.NET.5.127	2	6	8	5188	large qty 1064-1070; 22, 80, 1619, 8080, (ICMP)	large qty 161; 22, 80, 5632, 8080, (ICMP)	uniform background level; daily SPIKES
*	MY.NET.5.128	1	5	6	5157	large qty 1064-1070; 22, 80, 1163, 8080	large qty 161; 22, 80, 8080	uniform background level; daily SPIKES
External hosts								
	211.115.213.202	8	0	8	9080	eph	80	multiple peaks
	211.32.116.112	4	0	4	8482	eph	80	SPIKES on 9, 16 Jan
	209.10.239.135	1	0	1	6050	eph	80	SPIKE on 9 Jan
	211.32.117.26	12	0	12	5074	eph	80	erratic peaks
	207.200.86.66	3	0	3	4726	eph	80	SPIKES on 10, 18 Jan
	216.241.219.14	1	0	1	4718	eph	80	SPIKE on 9 Jan
	224.0.0.2	149	0	149	4596	n/a	n/a	varies with business day
	211.115.213.207	5	0	5	2198	eph	80	several peaks
	211.32.117.227	13	0	13	2003	eph	80, 8080	erratic relation to business day
	64.12.184.141	15	0	15	1886	eph	80	erratic peaks

Table IV. Top Ten "Source" Addresses, from Scan logs

Within MY.NET								
	IP	**Talked To**			**Packets**	**Port Distribution**		**Time Distribution**
		Internal	**External**	**Total**		**"Source"**	**"Target"**	
*	MY.NET.60.43	164	0	164	1046708	large qty 0, 123, 7000; widely varying	eph + 0, 7001	Consistently high, except on weekend nights
						large qty 0, 256, 7000,	large qty 0, 7000, 7001	erratic

*	MY.NET.6.49	99	0	99	179734	7001, 8224, 65535; widely varying	7001, 8224, 65535; widely varying	SPIKES; huge SPIKE on 17 Jan
*	MY.NET.6.45	162	0	162	176371	large qty 0, 123, 7000; widely varying	large qty 0, 7001; widely varying	varies with calendar day; SPIKES on 15, 16, 20 Jan
*	MY.NET.6.52	112	0	112	165288	large qty 0, 256, 7000, 7001, 65535; widely varying	large qty 0. 7000, 7001, 65535; widely varying	erratic SPIKES; huge SPIKE on 17 Jan
*	MY.NET.6.50	99	0	99	145877	large qty 0, 7000, 7001; widely varying	large qty 0, 7000, 7001; widely varying	erratic SPIKES; huge SPIKES on 16, 17 Jan
*	MY.NET.6.48	92	0	92	101994	large qty 0, 7000, 7001; widely varying	large qty 0, 7000, 7001, 8224, 65535; widely varying	erratic SPIKES
	MY.NET.150.143	16	12634	12650	58900	large qty 2688, 8765; eph + 20, 68	large qty 53, 80, 1214, 1900, 6257, 6699; eph + various well-known	Consistently high until 14 Jan, noise thereafter
*	MY.NET.6.60	125	0	125	55592	large qty 0, 256, 7000, 7001, 65535; widely varying	large qty 0, 7000, 7001, 65535; widely varying	varies with business day; SPIKES on 15, 16, 17, 22 Jan
*	MY.NET.6.53	114	0	114	42712	large qty 0, 256, 7000, 7001; widely varying	large qty 0, 7000, 7001, 65535; widely varying	varies with business day; SPIKES on 16, 17, 22 Jan
							large qty 53,	

*	MY.NET.153.196	24	14558	14582	36281	large qty 6257, 7001; eph + 68	806257, 6699; eph + various well-known	SPIKES on 15, 22 Jan
External hosts								
	205.188.228.65	14	0	14	52911	eph	large qty 6970; 6972-7100	erratic SPIKES amid high calendar day usage
	205.188.228.33	14	0	14	47449	eph	large qty 6970; 6972-7138	erratic SPIKES amid high calendar day usage
	205.188.228.1	14	0	14	46121	eph	large qty 6970; 6972-71330	erratic SPIKES amid high calendar day usage
	205.188.228.17	14	0	14	45469	eph	large qty 6970; 6972-7134	erratic SPIKES amid high calendar day usage
*	216.106.172.147	6	0	6	6342	large qty 0, 7000; widely varying	large qty 0, 7001; widely varying	peaks on 17, 20 Jan
*	216.106.172.148	6	0	6	5443	large qty 0, 7000; widely varying	large qty 0, 7001; widely varying	peaks on 17, 18, 20 Jan
*	216.106.173.148	7	0	7	4267	large qty 0, 7000; widely varying	large qty 0, 7001; widely varying	peaks on 17, 18 Jan
*	216.106.173.149	5	0	5	4215	large qty 0, 7000; widely varying	large qty 0, 7000; widely varying	peaks on 17, 18 Jan
*	216.106.172.149	4	0	4	3944	large qty 0, 7000, 34571; widely varying	large qty 0, 1522, 7001; widely varying	peaks on 17, 18, 20 Jan
*	216.106.173.147	5	0	5	3811	large qty 0, 7000, 34571; widely varying	large qty 0, 1522, 7001; widely varying	peaks on 17, 20 Jan

Table V. Top Ten "Target" Addresses, from Scan logs

Within MY.NET								
	IP	Talked To			Packets	Port Distribution		Time Distribution
		Internal	External	Total		"Source"	"Target"	
	MY.NET.1.3	382	0	382	91445	eph + 123	53, 123	varies (roughly) with calendar day
	MY.NET.1.4	325	0	325	67287	eph + 123, 137	53, 123	varies (roughly) with calendar day
*	MY.NET.6.45	162	0	162	50334	large qty 7001; eph	123, 7000	varies with calendar day; SPIKES on 15, 16, 20 Jan
*	MY.NET.60.43	162	0	162	47647	large qty 1030, 7001; eph	123, 7000	Consistently high, except on weekend nights
*	MY.NET.153.163	13	25	38	35874	large qty 0, 123, 7000; widely varying	large qty 0, 7000, 7001, 8224; widely varying	SPIKE on 17 Jan
*	MY.NET.153.172	14	31	45	31966	large qty 0, 123, 7000; widely varying	large qty 0, 7001; widely varying	small SPIKES on 10, 18 Jan
*	MY.NET.153.173	11	28	39	32757	large qty 0, 123, 7000; widely varying	large qty 0, 7001; widely varying	SPIKE on 14 Jan
*	MY.NET.153.204	11	27	38	31603	large qty 0, 123, 7000; widely varying	large qty 0, 7001; widely varying	erratic small SPIKES
*	MY.NET.152.178	9	22	31	34094	large qty 0, 7000, 7001, 65535; widely varying	large qty 0, 7000, 7001, 65535; widely varying	SPIKE on 17 Jan
						large qty 0, 123,	large qty 0, 7001;	SPIKE on

*	MY.NET.153.142	12	28	40	29456	7000; widely varying	0, 7001; widely varying	SPIKE on 17 Jan
External hosts								
	205.188.228.65	15	0	15	28484	large qty 6970; eph	large qty 554; eph + 80	erratic SPIKES amid high calendar day usage
	205.188.228.33	15	0	15	23528	large qty 6970; eph	large qty 554; eph + 80	several SPIKES amid high calendar day usage
	205.188.228.17	15	0	15	23433	large qty 6970; eph	large qty 554; eph + 80	erratic SPIKES amid high calendar day usage
	205.188.228.1	15	0	15	26426	large qty 6970; eph	large qty 554: eph	erratic SPIKES amid high calendar day usage
	66.28.14.131	10	0	10	9736	large qty 6970, 6972; eph	large qty 554; eph	primarily weekday afternoons, evenings
	131.118.254.39	273	0	273	8468	eph	80, 443	varies with calendar day
	131.118.254.38	264	0	264	6835	eph	80, 443	varies with calendar day
	64.224.194.150	10	0	10	6148	large qty 6970; eph	eph + 80, 554	primarily weekday afternoons, evenings
	131.118.254.40	251	0	251	4063	eph	80	varies with calendar day
	136.160.130.177	3	0	3	3648	large qty 6970; eph	eph + 554, 8080	primarily weekday afternoons, evenings

---

Analysis of the Various Top-Ten Nodes

1. **Compromised or questionable internal hosts**

   It should be noted that, by the time you receive and act on this report, compromises and infections noted here will undoubtedly spread. It is a good practice, when reinstalling and disinfecting a host, to review security on hosts related to the one being cleaned.

- The following nodes appear almost certainly compromised, based on traffic patterns containing some combination (in some cases all three) of Unicode attacks, Red Worm traffic, and CGI Null attacks.

  - **MY.NET.88.183**
  - **MY.NET.153.45**
  - **MY.NET.153.113**
  - **MY.NET.153.117**
  - **MY.NET.153.119**
  - **MY.NET.153.124**
  - **MY.NET.153.146**
  - **MY.NET.153.164**
  - **MY.NET.153.196**
  - **MY.NET.153.211**

- There is a large presence of AFS traffic, whether a commercial version or not, in the internal campus network, and several external nodes are sending AFS traffic as well. I am unaware of the relative security of AFS (although I have seen it recommended), but as a general principle I am not sanguine about filesystem or peer-to-peer production networking across a border with the Internet, and thus would recommend that AFS across the border be prohibited.

  The following hosts are sending large quantities of AFS traffic, and also significant quantities suspicious or at least broken traffic (port 0, traffic to widely disparate port numbers, including the many of the uncommonly used "well-known" ports, and occasional hits on the Red Worm port 65535):

  - **MY.NET.6.45**
  - **MY.NET.6.48**
  - **MY.NET.6.49**
  - **MY.NET.6.50**
  - **MY.NET.6.52**
  - **MY.NET.6.63**
  - **MY.NET.6.60**
  - **MY.NET.60.43**
  - **216.106.172.147**
  - **216.106.172.148**
  - **216.106.172.149**
  - **216.106.172.147**
  - **216.106.172.148**
  - **216.106.172.149**

2. **Internal hosts of significance**

- **MY.NET.1.3** and **MY.NET.1.4** are DNS and NTP servers; MY.NET.1.4 is also doing some SMB on the side, but *not* triggering the SMB Name Wildcard rule.
- **MY.NET.70.177** is pretty obviously a Windows-based NMS server, as it leads the pack of senders of SNMP public traffic, and is also sending quantities of SMB Name Wildcard traffic.
- **MY.NET.150.198** is an lpr server and an NMS, as it is receiving virtually all the extreme quantity of Port 515 traffic being sent, and it is sending large quantities of SNMP Public traffic.
- **MY.NET.152.109** is the top recipient of SNMP Public traffic. He and all others should be configured with a different community name or names.
- **MY.NET.5.96** is another recipient of SNMP Public traffic; he also appears to be running WWW server, as it is on the receiving end of no fewer than nine distinct WWW attacks.
- **MY.NET.153.219** is another SNMP recipient, and is also running an ftp server, likely an anonymous one (the way that rule is written, it triggers on the attempt, not

success).
- **MY.NET.5.127** and **MY.NET.5.128** are recipients of SNMP and scant else.
- **MY.NET.150.143** is generating a lot of traffic, but it appears to be nonmalicious; the user of this node is doing a great deal of Napster and other filesharing traffic, and some surfing and e-mail as well.

3. **Badly-behaved or questionable external hosts**

- **211.233.70.161**, **211.233.45.39** and **211.233.45.41** are unregistered hosts in Korea, whose traffic is rife with UDP port zero and other odd or suspicious traffic; they are primary triggers of the MISC Large UDP rule. Registrations for these networks, from whois.nic.or.kr (only the English text is shown here):

```
Korea Internet Information Service V1.0 ( created by KRNIC, 2001.6 )

query: 211.233.70.161

# ENGLISH

IP Address          : 211.233.70.0-211.233.70.255
Network Name        : KIDC-INFRA-COLOCATION
Connect ISP Name    : KIDC
Connect Date        : 20010629
Registration Date   : 20010826

[ Organization Information ]
Orgnization ID      : ORG141241
Org Name            : KIDC
State               : SEOUL
Address             : 261-1 Nonhyun-dong Kangnam-ku
Zip Code            : 135-010

[ Admin Contact Information]
Name                : SangGyu Jang
Org Name            : KIDC
State               : SEOUL
Address             : 261-1 Nonhyun-dong Kangnam-ku
Zip Code            : 135-010
Phone               : +82-2-6440-2920
Fax                 : +82-2-6440-2909
E-Mail              : support@kidc.net

[ Technical Contact Information ]
Name                : TaeUng Kim
Org Name            : KIDC
State               : SEOUL
Address             : 261-1 Nonhyun-dong Kangnam-ku
Zip Code            : 135-010
Phone               : +82-2-6440-1965
Fax                 : +82-2-6440-2909
E-Mail              : ip@kidc.net

===============================================================

Korea Internet Information Service V1.0 ( created by KRNIC, 2001.6 )

query: 211.233.45.39

# ENGLISH

IP Address          : 211.233.45.0-211.233.45.255
Network Name        : KIDC-INFRA-COLOCATION
Connect ISP Name    : KIDC
Connect Date        : 20010412
Registration Date   : 20010413

[ Organization Information ]
```

```
Orgnization ID      : ORG141241
Org Name            : KIDC
State               : SEOUL
Address             : 261-1 Nonhyun-dong Kangnam-ku
Zip Code            : 135-010

[ Admin Contact Information]
Name                : SangGyu Jang
Org Name            : KIDC
State               : SEOUL
Address             : 261-1 Nonhyun-dong Kangnam-ku
Zip Code            : 135-010
Phone               : +82-2-6440-2920
Fax                 : +82-2-6440-2909
E-Mail              : support@kidc.net

[ Technical Contact Information ]
Name                : TaeUng Kim
Org Name            : KIDC
State               : SEOUL
Address             : 261-1 Nonhyun-dong Kangnam-ku
Zip Code            : 135-010
Phone               : +82-2-6440-1965
Fax                 : +82-2-6440-2909
E-Mail              : ip@kidc.net
```

1. **External hosts: benign observations**

   - **212.179.35.118**, **212.179.48.2** and **212.179.35.8** are logging the most traffic of almost any external hosts, but that is because they are coming from a Watchlisted network.
   - **63.250.213.99** and **63.250.213.98** are content servers in the domain *.broadcast.com of Yahoo! Broadcasting Services.
   - **64.124.157.48**, **64.124.157.58**, and **131.118.254.38-40** also have much to tell by their names, as they are content caches in *.akamai.com. **MY.NET.151.63** logged no traffic except to these external hosts.
   - Streaming content using RealVideo and RealAudio appears quite popular in your network. Servers in network **205.188.228.0/24**, and hosts **66.28.14.131**, **64.224.194.150**, and **136.160.130.177** are primary sources, and are setting off Snort's portscan logger.
   - **224.0.0.2** is, of course, a multicast *all-routers* address and not truly an external host at all.
   - All nine of the other top-ten non-MY.NET recipients of traffic received exclusively WWW attacks, of either the Unicode or CGI Null variety - actions thus indicated are indirect, against the sources of these attacks within MY.NET .

Out-Of-Specification (OOS) Analysis

It is not clear to me how the OOS files were derived - they are obviously generated by playing back logfiles through a special Snort ruleset, but the details of that ruleset were not provided and the criteria are somewhat opaque to me.

All packets in the OOS files appear to have been TCP, or at least attempts at TCP (twelve fragments were logged on January 15th). Further, they all appear to have invalid (plus or minus the reserved/ECN bits) TCP flags.

The following table records a selection of permutations of TCP flags among the packets in the OOS logs, and notes how the OOS logs thus differ from the scan logs. It is clearly demonstrated that the OOS logs are not a subset of the scan logs, although there is considerable overlap.

count from OOS log	flags	count from scan log

920	**SF****	920
12	21S*****	8
5	2*SFRP*U	9
4	21SFRPAU	8
3	21**RPA*	3
2	21S***AU	2
2	21*FRP*U	3

In the following table, two consecutive packets from the OOS logs are interpolated into excerpts from the Alert logs. The packets triggering by the different Snort rulesets exhibit no overlap whatsoever.

01/09-02:44:17.171728 [**] MY.NET.5.202 -> MY.NET.5.1
01/09-02:44:27.168183 [**] MY.NET.5.202 -> MY.NET.5.1
01/09-02:44:35.375660 212.175.38.114:3072 -> MY.NET.150.49:1214
01/09-02:44:37.164557 [**] MY.NET.5.202 -> MY.NET.5.1
01/09-02:44:47.158360 [**] MY.NET.5.202 -> MY.NET.5.1
01/09-02:46:47.130772 [**] MY.NET.5.202 -> MY.NET.5.1
01/09-02:47:07.124525 [**] MY.NET.5.202 -> MY.NET.5.1
01/09-02:47:23.251697 212.175.38.114:3200 -> MY.NET.150.49:21845
01/09-02:47:24.611363 [**] MY.NET.70.177:1069 -> MY.NET.5.83:161
01/09-02:47:24.707991 [**] MY.NET.70.177:1069 -> MY.NET.5.83:161

Now knowing what this data is *not*, what can we do with it?

Of the 983 OOS packets and fragments here, some 920 of them are SYN-FIN . As we have already covered these packets in the "SYN-FIN scan!" alert analysis, we move on to the remaining sixty-three.

Twelve line items in the logs are fragments - two are shown here:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/15-13:01:54.969619 66.127.113.88 -> MY.NET.150.145
TCP TTL:109 TOS:0x0 ID:57609  DF MF
Frag Offset: 0x0   Frag Size: 0x22
F2 17 C4 33 87 14 92 6A BE 92 90 FF F7 7F 63 0C  ...3...j......c.
3A 8A A9 3F 0D 76 F6 C4 58 A8 A5 65 0E F1 2C DD  :..?.v..X..e..,.
3F 7C                                            ?|

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/15-13:01:55.145221 66.127.113.88 -> MY.NET.150.145
TCP TTL:109 TOS:0x0 ID:57611  DF MF
Frag Offset: 0x0   Frag Size: 0x22
06 30 0E 80 F9 B7 13 27 2A 17 61 7E 81 3F D5 B5  .0.....'*.a~.?..
F2 10 F9 96 72 E1 CE B6 0E 55 A1 70 2E 7E D7 BB  ....r....U.p.~..
82 C1                                            ..
```

Note that both the "Don't Fragment" *and* the "More Fragments" flags are set, indicating one of several things - packet craft, corruption, or a network device along the way that ignored the DF flag. A fragment size that is not a multiple of eight is suspicious as well.

Of the remaining 51 packets, 30 have TCP options. Breaking things down further, we find:

- 12 with what appear to be normal options, though corrupted flags:

```
01/21-14:03:02.912781 12.246.128.81:34022 -> MY.NET.150.133:1214
TCP TTL:45 TOS:0x0 ID:58291  DF
21S***** Seq: 0x330295FF   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 4552268 0 EOL EOL EOL EOL
```

- 4 with obviously corrupted sets of options :

```
01/15-14:22:19.213096 205.251.226.211:1214 -> MY.NET.88.162:1489
TCP TTL:107 TOS:0x0 ID:50863  DF
2*SFRP*U Seq: 0xB70   Ack: 0x5319702E   Win: 0x5010
TCP Options => EOL EOL Opt 126 (11): 4F96 AC13 0094 2332 E456
Opt 86 Opt 86 Opt 86 Opt 86 Opt 86 Opt 86 Opt 86 Opt 86 Opt 86
Opt 86 ... [40 more]
```

while another sort simply pads with EOL's that do not need to be there:

```
01/15-07:33:28.815611 24.202.117.160:2711 -> MY.NET.88.162:1214
TCP TTL:109 TOS:0x0 ID:13953  DF
21SF*P** Seq: 0x404D8   Ack: 0xF6BBEB39   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL
```

Finally, there are 21 packets remaining. There is not a lot to be said about them, other than that they *are* corrupted. In most cases, one can infer intentions from the source or target port (a lot of the OOS traffic - some 37 of the 51 packets - references the KaZaA port 1214 or the Gnutella port 6346), but in any network, corruption occurs.

References

1. Security Information
   - SANS
     - The SANS Institute ~ System Administration, Networking and Security
     - SANS Institute: Information Security Reading Room
     - GIAC: Global Information Assurance Certification - Home Page
     - incidents.org - The Internet's Threat Monitor
     - SANS. *3.1 TCP/IP for Intrusion Detection*.
     - SANS. *3.2/3.3AM Network Traffic Analysis Using tcpdump*.
     - SANS. *3.3PM/3.4 Intrusion Detection - Snort Style*.
     - SANS. *3.5 Part 1/3.6 Part 2 IDS Signature and Analysis*.
   - DShield.org Distributed Intrusion Detection System
   - Common Vulnerabilities and Exposures
   - CERT Coordination Center
   - U.S. DOE-CIAC (Computer Incident Advisory Capability) Website
   - SecurityFocus home
   - SecurityPortal - Weekly Solaris Security Digest
   - The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Boston: Addison-Wesley, 2001.
2. Network / Computing Related
   - Distributed Applications Support Team at NLANR
   - American Registry for Internet Numbers (ARIN)
   - RIPE NCC Homepage
   - Unicode Home Page
   - Internet FAQ Consortium
   - Connected: An Internet Encyclopedia
   - Google
   - Albitz, Paul and Cricket Liu. *DNS and BIND*. Sebastopol, CA: O'Reilly and Associates, Inc., 1992.

- Wall, Larry and Tom Christiansen and Randal L. Schwartz. *Programming Perl, Second Edition*. Sebastopol, CA: O'Reilly and Associates, Inc., 1996.

3. Security-Related Software
   - Welcome to F-Secure, Securing the Mobile Enterprise
   - Nmap -- Free Stealth Port Scanner ...
   - dsniff
   - TCPDUMP public repository
   - LaBrea - The Tarpit
   - Laurent Constantin, lcrzoex, lcrzo
   - Snort - The Open Source Network IDS
   - Geocrawler.com Archives of the Snort-users mailing list
4. Other Software
   - SourceForge: Project Info - Back Orifice 2000
   - CULT OF THE DEAD COW
   - hping2 devel page
   - .NET Messenger Service
   - The CyberKit Homepage
   - mIRC - An Internet Relay Chat program
   - Napster
   - ICQ.com
   - Maximize Your Media - Welcome to Real.com. ...
   - Gnutella.com
   - KaZaA Media Desktop
   - OpenAFS
   - MRTG: The Multi Router Traffic Grapher
5. Commercial Reference
   - Akamai - Delivering a better Internet
   - Cisco Connection Online by Cisco Systems, Inc.
   - Compaq Services - Software Patches
   - Products AFS Information
6. Proofreading
   - Elizabeth Fleming Larratt
   - Catherine Foulston

Last Updated: February 24th, 2018

Upcoming Training

SANS Northern VA Spring - Tysons 2018	McLean, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Abu Dhabi 2018	Abu Dhabi, United Arab Emirates	Apr 07, 2018 - Apr 12, 2018	Live Event
SANS London April 2018	London, United Kingdom	Apr 16, 2018 - Apr 21, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
Baltimore Spring 2018 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Apr 23, 2018 - Apr 28, 2018	vLive
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201805,	May 02, 2018 - Jun 14, 2018	vLive
Community SANS Virginia Beach SEC503	Virginia Beach, VA	May 07, 2018 - May 12, 2018	Community SANS
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
Mentor Session - SEC503	Dulles, VA	May 24, 2018 - Jun 28, 2018	Mentor
SANS Oslo June 2018	Oslo, Norway	Jun 18, 2018 - Jun 23, 2018	Live Event
SANS Minneapolis 2018	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	Live Event
Minneapolis 2018 - SEC503: Intrusion Detection In-Depth	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	vLive
SANS London July 2018	London, United Kingdom	Jul 02, 2018 - Jul 07, 2018	Live Event
SANSFIRE 2018	Washington, DC	Jul 14, 2018 - Jul 21, 2018	Live Event
Security Operations Summit & Training 2018	New Orleans, LA	Jul 30, 2018 - Aug 06, 2018	Live Event
SANS San Antonio 2018	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	Live Event
Community SANS Columbia SEC503	Columbia, MD	Aug 13, 2018 - Aug 18, 2018	Community SANS
SANS Virginia Beach 2018	Virginia Beach, VA	Aug 20, 2018 - Aug 31, 2018	Live Event
SANS Amsterdam September 2018	Amsterdam, Netherlands	Sep 03, 2018 - Sep 08, 2018	Live Event
SANS Tokyo Autumn 2018	Tokyo, Japan	Sep 03, 2018 - Sep 15, 2018	Live Event
SANS Network Security 2018	Las Vegas, NV	Sep 23, 2018 - Sep 28, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced