Intrusion Detection in Depth

GCIA Practical Assignment (v 3.2)

Ronnie Clark
October 20, 2002

Table of Contents:

Assignment #1 – Describe the State of Intrusion Detection

FreeBSD and IPFW as IDS

We all know the Internet is not a very friendly place at times and that security for the average home user is sometimes nonexistent. Advances have started to arm the average home user and protect them from attacks. Of course one factor that can cause failures are people, whether through ignorance, indifference or finances. This paper will strive to show that with little money and work you can have both protection and an adequate IDS.

FreeBSD and IPFW

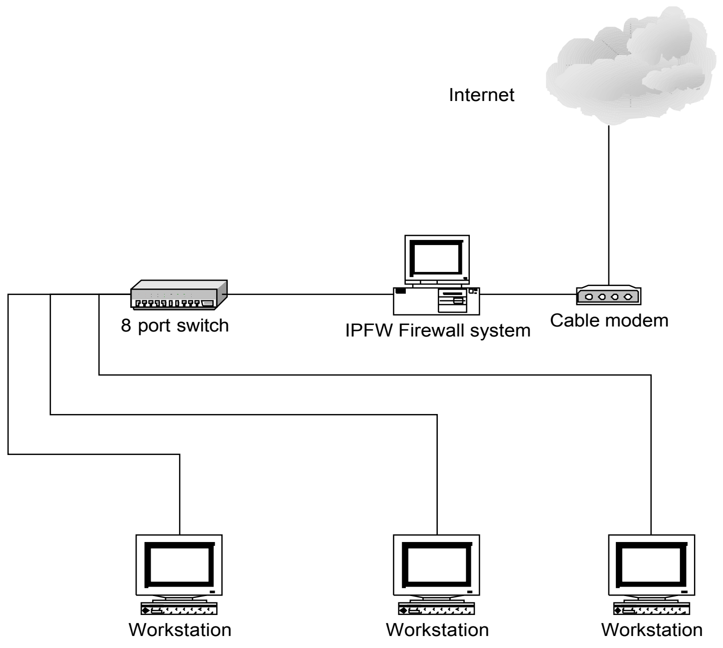
FreeBSD is a UNIX like operating system that has been derived from the old AT&T UNIX of the 60's. [1] It is often compared to Linux though doesn't seem to be as mainstream or popular. It is available free from http://www.freebsd.org and can be installed easily on to just about any hardware. The OS is free and a packet filtering firewall is built in. It is called IPFW. Used with NAT (Network Address Translation) it can be powerful protection against the outside world.

IPFW is a stateful packet filtering firewall.[2] This means that the firewall pays attention to the direction and flow of traffic and remembers the TCP state of any session. IPFW comes standard with three types of rule bases: open, client and simple. "Open" allows all traffic and is not recommended. "Client" rule base allows some services out to the Internet. "Simple" rule base is intended only for systems that offer services to the Internet. Of course, you can always scrap the prepackaged rule bases and write your own. All of these rule bases offer the one thing needed to use for Intrusion Detection – logging.

IPFW logs in a syslog style and provides good information. However, I have found a patch[3] that can be applied that enhances the logging to include tcp flags, as well as sequence and acknowledgement numbers. This paper will show the differences in logging without the patch and with the patch. Then we will look at some hits to my home network system and see how they can be applied to the IDS methodology.

The Network

The network used for this is my home network. There are no web, mail, or other publicly accessible servers that are available to the internet. The following diagram shows the layout.



The Rules

The rules are in a configuration file called rc.ipfw. This is a custom rules file that I developed. Examples of the rules used on this firewall are as follows:

```
${fwcmd} add divert natd all from any to any via ${natd_interface}
${fwcmd} add pass tcp from any to any established
${fwcmd} add pass ip from 172.16.32.100 to any

# Allow loopbacks, deny imposters
${fwcmd} add 100 pass all from any to any via lo0
${fwcmd} add 200 deny all from any to 127.0.0.0/8

${fwcmd} add pass tcp from 10.7.0.0/28 to any in via ${iif} setup
${fwcmd} add pass udp from 10.7.0.0/28 to any in via ${iif} keep-
state
${fwcmd} add pass icmp from any to any

# Reject&Log all other connections from outside interface
${fwcmd} add 65000 deny log ip from any to any via ${oif}

# Everything else is denied by default, unless the
# IPFIREWALL_DEFAULT_TO_ACCEPT option is set in your kernel
# config file
```

Rules are broken down into the following format / syntax:

- `${fwcmd}` – the variable for the firewall command.
- `add` – this tells the kernel to make this the next available rule number. The system usually counts by 100s.
- `divert, pass, deny` – actions to be taken against a packet, should

`that packet match the rest of the rule.`

- `tcp, ip, udp, icmp, all` – with the exception of natd, this is the protocol of the packet being examined.
- `from any` – the source of the packet
- `to any` – the destination of the packet
- `setup, keep-state, established` – setup is looking to the beginning of a connection attempt. This is usually for tcp packets and looking for the tcp syn flags. Keep-state tells the firewall to remember the state of the session, whether it is tcp or udp. Once the connection is established, the connection is handed off to the “established” rule. In this scenario, it is the second rule in the list.

IPFW Logs

As stated before IPFW logs in a syslog manner, which basically means it logs to a flat text file. This log file can then be manipulated to extract whatever data is needed, whether imported into a database or just plain viewed in a text editor. IPFW, out of the box, logs like this:

```
Jul 24 19:07:50 ibmps1 /kernel: ipfw: 2800 Accept TCP
10.7.7.2:1029 159.37.31.159:443 in via ed2
Jul 24 19:07:50 ibmps1 /kernel: ipfw: 2800 Accept TCP
MY.HOME.NET.245:1029 159.37.31.159:443 out via ed1
Jul 24 19:07:53 ibmps1 /kernel: ipfw: 2800 Accept TCP
10.7.7.2:1030 159.37.31.159:443 in via ed2
Jul 24 19:07:53 ibmps1 /kernel: ipfw: 2800 Accept TCP
MY.HOME.NET.245:1030 159.37.31.159:443 out via ed1
Jul 24 19:07:53 ibmps1 /kernel: ipfw: 2800 Accept TCP
10.7.7.2:1033 216.33.75.21:443 in via ed2
Jul 24 19:07:53 ibmps1 /kernel: ipfw: 2800 Accept TCP
10.7.7.2:1032 216.33.75.21:443 in via ed2
Jul 24 19:07:53 ibmps1 /kernel: ipfw: 2800 Accept TCP
MY.HOME.NET.245:1033 216.33.75.21:443 out via ed1
Jul 24 19:07:53 ibmps1 /kernel: ipfw: 2800 Accept TCP
MY.HOME.NET.245:1032 216.33.75.21:443 out via ed1
```

This shows some normal traffic going from my home network with NAT in action. I will break down the first line to show how IPFW does its logging in standard format. The first line starts with Jul 24 19:07:50 which is obviously the time. Next it shows the host name of the system, the kernel, which processes the packets as they come into the system and the name of the firewall program, in this case IPFW. Then it shows the rule number that this event was triggered on. The rules are numbered either automatically as the rules are read at boot time in increments of 100, or you can manually number them, from 1 to 65535. I choose to let the system number the rules automatically, so the rule number 2800 means this is the 28th rule in the rule base. Next, it shows that this was accepted traffic and passed on to the correct destination host. The source IP and source port are 10.7.7.2:1029 and the destination IP and port are 159.37.31.159:443. This is simply one of my home systems making a connection to a web server over SSL port. Lastly, it shows us which interface on the system the traffic came from, in this case, the traffic came in via ed2, the inside interface.

After applying the patch and issuing the command **sysctl -w net.inet.IP.fw.verbose=4**, the logging gets more detailed. It looks like this:

```
Jul 24 21:16:53 ibmps1 /kernel: ipfw: 2800 Accept TCP
10.7.7.2:1147 64.41.224.49:443 f=02 s=007efb57 a=00000000 in
via ed2 [tos 0x00] (ttl 128, id 35075, len 48)
Jul 24 21:16:53 ibmps1 /kernel: ipfw: 2800 Accept TCP
MY.HOME.NET.245:1147 64.41.224.49:443 f=02 s=007efb57
a=00000000 out via ed1 [tos 0x00] (ttl 127, id 35075, len 48)
Jul 24 21:16:54 ibmps1 /kernel: ipfw: 2800 Accept TCP
10.7.7.2:1149 64.58.77.171:443 f=02 s=007eff77 a=00000000 in
via ed2 [tos 0x00] (ttl 128, id 40963, len 48)
Jul 24 21:16:54 ibmps1 /kernel: ipfw: 2800 Accept TCP
MY.HOME.NET.245:1149 64.58.77.171:443 f=02 s=007eff77
a=00000000 out via ed1 [tos 0x00] (ttl 127, id 40963, len 48)
Jul 24 21:16:55 ibmps1 /kernel: ipfw: 2800 Accept TCP
10.7.7.2:1150 64.58.77.171:443 f=02 s=007f03ad a=00000000 in
via ed2 [tos 0x00] (ttl 128, id 50691, len 48)
Jul 24 21:16:55 ibmps1 /kernel: ipfw: 2800 Accept TCP
10.7.7.2:1151 64.58.77.171:443 f=02 s=007f03af a=00000000 in
via ed2 [tos 0x00] (ttl 128, id 51203, len 48)
Jul 24 21:16:55 ibmps1 /kernel: ipfw: 2800 Accept TCP
MY.HOME.NET.245:1150 64.58.77.171:443 f=02 s=007f03ad
a=00000000 out via ed1 [tos 0x00] (ttl 127, id 50691, len 48)
Jul 24 21:16:55 ibmps1 /kernel: ipfw: 2800 Accept TCP
MY.HOME.NET.245:1151 64.58.77.171:443 f=02 s=007f03af
a=00000000 out via ed1 [tos 0x00] (ttl 127, id 51203, len 48)
```

Now, let us examine the extra fields in the logs. Again, the individual events start out with the date and time, the name of the machine, the kernel, the name of the program, the rule number and the action. Also, it lists the source IP and port, then the destination IP and port in a.b.c.d:port_x format. Observe the next field, it is f=02. The "f" stands for "flags" and the "02" is the flag bit that is turned on, in hexadecimal notation. [4] In this case it is the SYN flag set. Next are the sequence and acknowledgement numbers. These numbers are hex representations of the values in these fields. There is a sequence number but the acknowledgement number is set to all zeros. Next is the interface that the rule triggered on. Lastly, there is the TCP options field, followed by the packet information. We see by the [tos 0x00] that there are no options set. Then we see that the time to live (ttl) is set to 127, the packet id number and the length of the packet. Now its time to see what bad traffic looks like and how this information can be used for intrusion detection.

Bad Traffic – Example #1:

```
Jul 27 08:40:39 ibmps1 /kernel: ipfw: 3100 Deny TCP
200.207.52.82:4232 MY.HOME.NET.245:1433 f=02 s=293555d9
a=00000000 in via ed1 [tos 0x00] (ttl 113, id 13748, len 44)
Jul 27 08:40:42 ibmps1 /kernel: ipfw: 3100 Deny TCP
200.207.52.82:4232 MY.HOME.NET.245:1433 f=02 s=293555d9
a=00000000 in via ed1 [tos 0x00] (ttl 113, id 39860, len 44)
Jul 27 08:40:48 ibmps1 /kernel: ipfw: 3100 Deny TCP
200.207.52.82:4232 MY.HOME.NET.245:1433 f=02 s=293555d9
a=00000000 in via ed1 [tos 0x00] (ttl 113, id 42677, len 44)
```

We start off with some events generated by the SQL Snake Worm. I say it is SQL snake because of the time frame in which the worm was released. Otherwise, this is just another vanilla SQL scan. I received this in the logs shortly after the worm was released into the wild. This traffic is coming from a presumed infected system in Sao Paulo, Brazil. The SQL Snake Worm infects systems running Microsoft SQL that are accessible from the Internet and have a blank administrator password. [5] Once infected it starts to propagate itself by scanning other networks for vulnerable systems. In the above trace, we can see by the enhanced logging that this appears to be the attacking host attempting the connection and then following up with two retries.

This is not active targeting, as these types of alerts from this host have not been seen again. This is also not a severe threat. There is no SQL server on this network exposed to the internet and there were no replies to these packets. Also based on the TTL value, it can be assumed that this is a Windows box approximately 15 hops away. As you can see this was denied by the firewall.

Bad Traffic – Example #2:

```
Jul 25 00:22:25 ibmps1 /kernel: ipfw: 3000 Deny TCP
211.239.77.138:48022 MY.HOME.NET.245:21 f=02 s=889662e7
a=00000000 in via ed1 [tos 0x00] (ttl 41, id 43428, len 60)
Jul 25 00:22:28 ibmps1 /kernel: ipfw: 3000 Deny TCP
211.239.77.138:48022 MY.HOME.NET.245:21 f=02 s=889662e7
a=00000000 in via ed1 [tos 0x00] (ttl 41, id 43429, len 60)
```

Here we see a couple of events captured by the firewall on the early morning minutes of July 25th. They show someone in Korea looking for vulnerable ftp servers. This is the first attempt followed by a single retry. This shows that this alert was triggered on rule 3000 or the 30th rule in the rule base. The packets were denied or dropped by the firewall. The SYN flag is set and the sequence number is in hex notation. There were also no options set in these packets.

The TTL value is 41. We could probably guess the OS the attacker is using as either Linux, FreeBSD, or some other *nix operating system according to http://www.map.ethz.ch/ftp-probleme.htm The versions of Microsoft Windows from 3.11 to NT 3.51 all have default TTL values of 32 and the NT 4.0 and later versions have the default TTL values set to 128. That would make this host some 87 hops away!

I would say this is not active targeting; it looks like a simple SYN scan looking for open ftp servers. I would also not rate this as severe, since there is no ftp server on this network exposed to the Internet, and no systems reply to these packets. It also appears to be a recon scan, looking for systems that are up and running the ftp services. Had I been running ftp services, it is possible that I would see more "attacks" from this host.

Bad Traffic – Example #3:

```
Jul 27 16:41:12 ibmps1 /kernel: ipfw: 3300 Deny TCP
24.200.92.146:1539 MY.HOME.NET.245:27374 f=02 s=01fe6dec
a=00000000 in via ed1 [tos 0x00] (ttl 15, id 2972, len 48)
Jul 27 16:41:15 ibmps1 /kernel: ipfw: 3300 Deny TCP
24.200.92.146:1539 MY.HOME.NET.245:27374 f=02 s=01fe6dec
a=00000000 in via ed1 [tos 0x00] (ttl 15, id 41884, len 48)
Jul 27 16:41:21 ibmps1 /kernel: ipfw: 3300 Deny TCP
24.200.92.146:1539 MY.HOME.NET.245:27374 f=02 s=01fe6dec
a=00000000 in via ed1 [tos 0x00] (ttl 15, id 53149, len 48)
Jul 27 16:41:33 ibmps1 /kernel: ipfw: 3300 Deny TCP
24.200.92.146:1539 MY.HOME.NET.245:27374 f=02 s=01fe6dec
a=00000000 in via ed1 [tos 0x00] (ttl 15, id 5536, len 48)

16:41:12.420130 24.200.92.146.1539 > MY.HOME.NET.245.27374: S
33451500:33451500(0) win 65535 <mss 1460,nop,nop,sackOK>
0x0000   4500 0030 0b9c 0000 0f06 890f 18c8 5c92        E..0..........\.
0x0010   MYHM NTf5 0603 6aee 01fe 6dec 0000 0000        ......j...m.....
0x0020   7002 ffff 8b26 0000 0204 05b4 0101 0402        p....&..........
16:41:15.258465 24.200.92.146.1539 > MY.HOME.NET.245.27374: S
33451500:33451500(0) win 65535 <mss 1460,nop,nop,sackOK>
0x0000   4500 0030 a39c 0000 0f06 f10e 18c8 5c92        E..0..........\.
0x0010   MYHM NTf5 0603 6aee 01fe 6dec 0000 0000        ......j...m.....
0x0020   7002 ffff 8b26 0000 0204 05b4 0101 0402        p....&..........
16:41:21.262338 24.200.92.146.1539 > MY.HOME.NET.245.27374: S
33451500:33451500(0) win 65535 <mss 1460,nop,nop,sackOK>
0x0000   4500 0030 cf9d 0000 0f06 c50d 18c8 5c92        E..0..........\.
0x0010   MYHM NTf5 0603 6aee 01fe 6dec 0000 0000        ......j...m.....
0x0020   7002 ffff 8b26 0000 0204 05b4 0101 0402        p....&..........
16:41:33.289624 24.200.92.146.1539 > MY.HOME.NET.245.27374: S
33451500:33451500(0) win 65535 <mss 1460,nop,nop,sackOK>
0x0000   4500 0030 15a0 0000 0f06 7f0b 18c8 5c92        E..0..........\.
0x0010   MYHM NTf5 0603 6aee 01fe 6dec 0000 0000        ......j...m.....
0x0020   7002 ffff 8b26 0000 0204 05b4 0101 0402        p....&..........
```

Now we see a SubSeven Trojan scan. In 21 seconds, I was hit 4 times by this IP address from Canada. This output below is from a simple nslookup:

```
Name:    modemcable146.92-200-24.mtl.mc.videotron.ca
Address:  24.200.92.146
```

The enhanced logging shows that the rule number that this alert was generated on was 3300, or the 33rd rule in the rule base. The attacker is looking for hosts compromised by the SubSeven Trojan version 2.1[6] or by the Ramen worm. [7] The SYN flag is set, the sequence number is listed, and there are no tcp options set. The TTL is 15 in these packets. This could be indicative of a compromised Windows 9x host. Though it could be a DEC Pathworks V5 or HP/UX 9.0 system, I think these options are unlikely.

This does not appear to be active targeting. It appears to be just another SYN scan. I again would not rate this as severe, since there were no replies made back to this system. Again, this looks like a recon scan from the 24.200.92.146 host.

Conclusion

We have seen how a free OS with a free packet filtering firewall provides simple protection for a home or business against anomalous traffic from the Internet. Add a simple and free logging upgrade script you have enhanced logging that is capable of showing some useful protocol information and providing some basic intrusion detection. Correlate with another IDS technology i.e. Snort,[8] Shadow[9] or tcpdump[10] and gather more information or evidence against the attacker. These logs and additional information could be used for incident handling. An email or formal written letter could be sent to the owner or to the ISP. This could alert these individuals that either they have been compromised, or that they have been caught for attempting to do something illegal. The logs could also be submitted to DShield,[11] a national database to track attackers and learn trends about attacks. The most important job is reviewing the logs, at least once a day. Ten minutes a day could save your home network!

References:

1. The Complete FreeBSD (3rd Edition) – Greg Lehey – Walnut Creek CDROM Books – Chapter 1, pp 7-8
2. Ipfw-HOWTO – author unknown – URL: http://www.freebsd-howto.com/HOWTO/Ipfw-HOWTO
3. IPFW Logging Extensions – Crist J. Clark, GCIA (no relation) – URL: http://people.freebsd.org/~cjc/ - November 2, 2001
4. Manual page for tcpdump – URL: http://www.tcpdump.org/tcpdump_man.html
5. MSSQL Worm (sqlsnake) on the rise – URL: http://www.incidents.org/diary/index.html?id=156 - May 22, 2002
6. SubSeven Trojan Information – URL: http://www.sans.org/newlook/resources/IDFAQ/subseven.htm
7. Ramen Worm – URL: http://www.sans.org/y2k/ramen.htm 02/15/2001
8. Snort Intrusion Detection – URL: http://www.snort.org
9. Shadow Intrusion Detection software – URL: http://www.nswc.navy.mil/ISSEC/CID/index.html
10. tcpdump – command line packet sniffer – URL: http://www.tcpdump.org
11. DShield – Distributed Intrusion Detection System – URL:

http://www.dshield.org

Appendix A – "The IPFW Extended Logging Patch" written by Crist J. Clark

```
Index: etc/rc.network
===================================================================
RCS file: /export/ncvs/src/etc/rc.network,v
retrieving revision 1.74.2.23
diff -u -r1.74.2.23 rc.network
--- etc/rc.network 2001/08/17 07:26:38         1.74.2.23
+++ etc/rc.network 2001/10/14 08:11:18
@@ -309,7 +309,7 @@
                             case ${firewall_logging} in
                             [Yy][Ee][Ss] | '')
                                     echo 'Firewall logging=YES'
-                                    sysctl -w net.inet.IP.fw.verbose=1 >/dev/null
+                                    sysctl -w
net.inet.IP.fw.verbose="${firewall_verbose:-1}" >/dev/null
                                     ;;
                             *)
                                     ;;
Index: sbin/Ipfw/Ipfw.8
===================================================================
RCS file: /export/ncvs/src/sbin/Ipfw/Ipfw.8,v
retrieving revision 1.63.2.16
diff -u -r1.63.2.16 Ipfw.8
--- sbin/Ipfw/Ipfw.8        2001/10/27 23:05:48         1.63.2.16
+++ sbin/Ipfw/Ipfw.8        2001/11/02 09:10:09
@@ -459,6 +459,15 @@
 Logging may then be re-enabled by clearing the logging counter
 or the packet counter for that entry.
 .Pp
+Logging of additional IP and TCP information is available by setting
+bits in the value of
+.Em net.inet.IP.fw.verbose .
+If the 2-bit is set, additional IP information is included in each log
+entry. If the 4-bit is set, additional TCP information is
+included. (The baseline information is included whenever
+.Em net.inet.IP.fw.verbose
+is non-zero.)
+.Pp
 Console logging and the log limit are adjustable dynamically
 through the
 .Xr sysctl 8
Index: etc/defaults//rc.conf
===================================================================
RCS file: /export/ncvs/src/etc/defaults/rc.conf,v
retrieving revision 1.53.2.41
diff -u -r1.53.2.41 rc.conf
--- etc/defaults//rc.conf    2001/10/10 15:56:25         1.53.2.41
+++ etc/defaults//rc.conf    2001/10/14 08:11:18
@@ -49,6 +49,7 @@
 firewall_type="UNKNOWN"              # Firewall type (see /etc/rc.firewall)
 firewall_quiet="NO"                  # Set to YES to suppress rule display
 firewall_logging="NO"           # Set to YES to enable events logging
+firewall_verbose="1"                 # Set verbosity level, 0 to 3.
 firewall_flags=""           # Flags passed to Ipfw when type is a file
 IP_portrange_first="NO"              # Set first dynamically allocated port
 IP_portrange_last="NO"               # Set last dynamically allocated port
Index: sys/netinet/IP_fw.c
===================================================================
RCS file: /export/ncvs/src/sys/netinet/IP_fw.c,v
retrieving revision 1.131.2.26
diff -u -r1.131.2.26 IP_fw.c
--- sys/netinet/IP_fw.c      2001/10/04 01:56:01         1.131.2.26
```

```
+++ sys/netinet/IP_fw.c      2001/10/11 23:33:54
@@ -471,7 +471,7 @@
     struct icmp *const icmp = (struct icmp *) ((u_int32_t *) IP + IP->IP_hl);
     u_int64_t count;
     char *action;
-    char action2[32], proto[47], name[18], fragment[27];
+    char action2[32], proto[74], name[18], fragment[27], IPvals[44];
     int len;

     count = f ? f->fw_pcnt : ++counter;
@@ -555,9 +555,16 @@
                        len += snprintf(SNPARGS(proto, len), " ");
                len += snprintf(SNPARGS(proto, len), "%s",
                     inet_ntoa(IP->IP_dst));
-               if (offset == 0)
-                       snprintf(SNPARGS(proto, len), ":%d",
+               if (offset == 0) {
+                       len += snprintf(SNPARGS(proto, len), ":%d",
                                ntohs(tcp->th_dport));
+                       if (fw_verbose > 2)
+                               snprintf(SNPARGS(proto, len),
+                                       " f=%02x s=%08lx a=%08lx",
+                                       tcp->th_flags,
+                                       ntohl(tcp->th_seq),
+                                       ntohl(tcp->th_ack));
+               }
                break;
     case IPPROTO_UDP:
               len = snprintf(SNPARGS(proto, 0), "UDP %s",
@@ -597,15 +604,23 @@
                         (IP->IP_off & IP_MF) ? "+" : "");
     else
               fragment[0] = '\0';
+
+    if (fw_verbose > 1)
+              snprintf(SNPARGS(IPvals, 0), " [tos 0x%02x] (ttl %u, id %u, len %u)",
+                        IP->IP_tos, IP->IP_ttl, ntohs(IP->IP_id), IP_len);
+    else
+              IPvals[0] = '\0';
+
     if (oif)
-              log(LOG_SECURITY | LOG_INFO, "%s %s %s out via %s%d%s\n",
-                   name, action, proto, oif->if_name, oif->if_unit, fragment);
+              log(LOG_SECURITY | LOG_INFO, "%s %s %s out via %s%d%s%s\n",
+                   name, action, proto, oif->if_name, oif->if_unit, fragment,
+                   IPvals);
     else if (rif)
-              log(LOG_SECURITY | LOG_INFO, "%s %s %s in via %s%d%s\n", name,
-                   action, proto, rif->if_name, rif->if_unit, fragment);
+              log(LOG_SECURITY | LOG_INFO, "%s %s %s in via %s%d%s%s\n", name,
+                   action, proto, rif->if_name, rif->if_unit, fragment, IPvals);
     else
-              log(LOG_SECURITY | LOG_INFO, "%s %s %s%s\n", name, action,
-                   proto, fragment);
+              log(LOG_SECURITY | LOG_INFO, "%s %s %s%s%s\n", name, action,
+                   proto, fragment, IPvals);
     if ((f ? f->fw_logamount != 0 : 1) &&
         count == (f ? f->fw_loghighest : fw_verbose_limit))
              log(LOG_SECURITY | LOG_NOTICE,
```

Assignment #2 – Network Detects

Detect #1:

This is the detect that I submitted to the mailing list, as per GCIA 3.2 requirements.   Any corrections or additions that I made are in red.

```
[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:42:49.104488 211.47.255.21:34318 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCA09443E  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:42:52.094488 211.47.255.21:34318 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCA09443E  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:44:25.104488 211.47.255.21:36367 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:44:28.104488 211.47.255.21:36367 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:44:34.104488 211.47.255.21:36367 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:44:46.104488 211.47.255.21:36367 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
20:44:46.104488 211.47.255.21.36367 > 46.5.127.19.0: S [bad tcp cksum f8f8!]
3480358410:3480358410(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF) (ttl 47, id
0, len 52, bad cksum d36d!)
0x0000   4500 0034 0000 4000 2f06 d36d d32f ff15        E..4..@./..m./..
0x0010   2e05 7f13 8e0f 0000 cf72 0e0a 0000 0000        .........r......
0x0020   8002 16d0 7465 0000 0204 05b4 0101 0402        ....te..........
0x0030   0103 0300                                      ....
```

1. **Source of Trace:** The source of this trace comes from http://www.incidents.org/logs/Raw and I am showing the first two and last four packets of the total detect as well as a packet decode using the following command line: `tcpdump -r <dumpfile> -n -vv -X host 211.47.255.21`

2. **Detect was generated by:** The method of detection used was running a given tcpdump file through the Snort 1.8.6 engine using the default rule set. The rule that triggered the alert was:
```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD TRAFFIC
tcp port 0 traffic"; sid:524;  classtype:misc-activity; rev:3;)
```
3. **Probability the source address was spoofed:** The probability that the source address was spoofed is unlikely. ~~There should be no response at all. If there was a response it would be RST / ACK and could be used for reconnaissance.~~ I believe the attacker has prior knowledge of this system and is now actively trying to compromise it or DOS attack it. Maybe he is banking that the system will not be able to handle the port 0 traffic correctly and lock up or crash in some form.
4. **Description of attack:** It seems that this is a scan to elicit some sort of response from the victim host or some form of DOS. I do not particularly think this is a DOS attempt due to the fact that there is just one packet every 3 – 12 seconds and the attack only lasted for one minute and fifty-six seconds. (1:56)
5. **Attack Mechanism:** The mechanism I believe to be used is the hping2 scanner. The reason being that the default destination port is port 0. Also, in the packet decode, we see evidence of bad tcp and IP checksums. This is also accomplished in hping2 by using the ***–b*** option. I believe the command used was hping –b host(s).
6. **Correlations:** Checking on the hping website, http://www.hping.org/manpage.html, this definitely confirms this type of scan behavior. I also found this thread with someone describing hping and Snort: http://archives.neohapsis.com/archives/snort/2000-12/0163.html. Next is an example of a DOS attack using destination tcp port 0 - http://www.der-keiler.de/Mailing-Lists/securityfocus/focus-ids/2001-08/0084.html.
7. **Evidence of active targeting:** I would say this is definitely active targeting. The attacker seems to know that this system is up and running because he/she never waivers to any other addresses in the net block.
8. **Severity:** (using formula severity = (criticality + lethality) – (system countermeasures + network countermeasures))
   Values are from 1 (lowest) to 5 (highest)
   ~~(4 + 2) – (5 + 5) = -4 severity~~ (3+3) – (2+2) = 2 Severity

   Explanation:

   Criticality – This is an unknown system on an unknown network. But, judging by the trace, this could be a web server, or ftp server, etc.
   Lethality – This looks like a potential DOS attack, with interesting features like the bad checksums and to destination port 0.
   System Countermeasures – ~~The system should never respond on tcp port zero (0). Defenses built into protocol.~~ This system should respond with a RST/ACK from port 0.

Network Countermeasures – This packet should be dropped at the firewall (assuming there is one) or border router since it is destined for port 0 and there are no services that utilize this tcp port number. However, in this trace, the packets appear to get to the host.

~~9.~~ Defensive recommendation: ~~Defenses are fine. No services running on that port.~~ Review border router and/or firewall rules and configure to drop and log any traffic destined for port 0.

10. Multiple choice test question:

```
[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:44:25.104488 211.47.255.21:36367 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:44:28.104488 211.47.255.21:36367 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:44:34.104488 211.47.255.21:36367 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BAD TRAFFIC tcp port 0 traffic [**]
07/05-20:44:46.104488 211.47.255.21:36367 -> 46.5.127.19:0
TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Based on the above trace, what type of response should the attacker see?

a) SYN/ACK
b) ACK
c) RST/ACK
d) No Response

Answer: ~~d)~~ c)

This detect was submitted to the intrusions@incidents.org mailing list for peer review. Here are the comments, questions and my responses.

From: Paul Asadoorian [mailto:Paul_Asadoorian@brown.edu]
Sent: Friday, August 09, 2002 7:25 AM
To: Ronald Clark; intrusions@incidents.org
Subject: Re: GIAC GCIA Version 3.2 Practical Detect(s)

Ron,

See below....

Paul Asadoorian

----- Original Message -----
From: "Ronald Clark" <rclark@swbanktx.com>
To: <intrusions@incidents.org>
Sent: Thursday, August 08, 2002 2:52 PM
Subject: LOGS: GIAC GCIA Version 3.2 Practical Detect(s)

```
>
>
>
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:42:49.104488 211.47.255.21:34318 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
> ******S* Seq: 0xCA09443E  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
>
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:42:52.094488 211.47.255.21:34318 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
> ******S* Seq: 0xCA09443E  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Looking at this detect I would have to ask why the source port is the same in the above two packets, then changes in the next three? Is this consistent hping2 behavior? I believe the answer is yes, and that source port is a configurable option in hping2. It can be set to a fixed number, but the default is to increment with each packet sent. mmmm....

```
>
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:44:25.104488 211.47.255.21:36367 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
> ******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
>
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:44:28.104488 211.47.255.21:36367 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
> ******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
>
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:44:34.104488 211.47.255.21:36367 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
> ******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
>
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:44:46.104488 211.47.255.21:36367 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
> ******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
>
> 20:44:46.104488 211.47.255.21.36367 > 46.5.127.19.0: S [bad tcp cksum
f8f8!]
> 3480358410:3480358410(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0>
> (DF) (ttl 47, id 0, len 52, bad cksum d36d!)
> 0x0000   4500 0034 0000 4000 2f06 d36d d32f ff15        E..4..@./..m./..
> 0x0010   2e05 7f13 8e0f 0000 cf72 0e0a 0000 0000        .........r......
> 0x0020   8002 16d0 7465 0000 0204 05b4 0101 0402        ....te..........
> 0x0030   0103 0300                                      ....
>
```

> 1. Source of Trace: The source of this trace comes from
> http://www.incidents.org/logs/Raw and I am showing the first two and last
> four packets of the total detect as well as a packet decode using the
> following command line: tcpdump -r <dumpfile> -n -vv -X host
211.47.255.21
> 2. Detect was generated by: The method of detection used was running a
> given tcpdump file through the Snort 1.8.6 engine using the default rule
> set. The rule that triggered the alert was:
> alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD TRAFFIC tcp port 0
> traffic"; sid:524; classtype:misc-activity; rev:3;)
> 3. Probability the source address was spoofed: The probability that the
> source address was spoofed is unlikely. There should be no response at
all.
> But if there was a response, it would be RST / ACK and could be used for
> reconnaissance.

Well, a RST/ACK is a response none the less. Are there certain Operating
systems that respond with a RST/ACK and others that do not respond at all?

> 4. Description of attack: This is a port scan to tcp port 0 on the
> target host(s). The attacker is trying to elicit some sort of response,
most
> likely for reconnaissance. I do not particularly think this could be a DOS

> attempt due to the fact that there is just one packet every 3 - 12 seconds
> and the attack only lasted for one minute and 56 seconds. (1:56)

What kind of reconnaissance? What information is the attacker trying to
get?

> 5. Attack Mechanism: The mechanism I believe to be used is the hping2
> scanner. The reason being that the default destination port is port 0.
Also,
> in the packet decode, we see evidence of bad tcp and IP checksums. This is
> also accomplished in hping2 by using the -b option.

What was the hping command that you theorize was run? (Also remember the
source ports from above). Given the source ports, was hping run just once?

> 6. Correlations: After searching incidents.org, and a Google search, I
> have seen no evidence of this type of scan before. But checking on the
hping
> website, http://www.hping.org/manpage.html, this definitely confirms this
> type of scan behavior.
> 7. Evidence of active targeting: I would say this is definitely active
> targeting. The attacker seems to know that this system is up and running
> because he/she never waivers to any other addresses in the net block.

Agreed, one source IP --> one destination IP is active targeting.

> 8. Severity: (using formula severity = (criticality + lethality) -
> (system countermeasures + network countermeasures))
> Values are from 1 (lowest) to 5 (highest)
> (4 + 2) - (5 + 5) = -4 severity
> Explanation:
> Criticality - This is an unknown system on an unknown network. But,
judging
> by the trace, this could be a web server, or ftp server, etc.
> Lethality - This looks like a port scan, but with interesting features
like
> the bad checksums and to destination port 0.
> System Countermeasures - The system should never respond on tcp port zero
> (0).
> Network Countermeasures - This packet should be dropped at the firewall or
> border router since it is destined for port 0 and there are no services
that
> utilize this tcp port number.
> 9. Defensive recommendation: Defenses are fine. No services running on
> that port.
> 10. Multiple choice test question:
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:44:25.104488 211.47.255.21:36367 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF

```
> ******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
>
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:44:28.104488 211.47.255.21:36367 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
> ******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
>
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:44:34.104488 211.47.255.21:36367 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
> ******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
>
> [**] BAD TRAFFIC tcp port 0 traffic [**]
> 07/05-20:44:46.104488 211.47.255.21:36367 -> 46.5.127.19:0
> TCP TTL:47 TOS:0x0 ID:0 IPLen:20 DgmLen:52 DF
> ******S* Seq: 0xCF720E0A  Ack: 0x0  Win: 0x16D0  TcpLen: 32
> TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

> Based on the above trace, what type of response should the attacker see?
>
> a) SYN/ACK
> b) ACK
> c) RST/ACK
> d) No response
>
> Answer: d)
>
> ------------------------------
> Thank you,
> Ron Clark

Good job Ron, finding the tool that was used in the attack is not always
easy. Expand on it a bit further and you'll be all set!

From: Bryce_Alexander@Vanguard.com [mailto:Bryce_Alexander@Vanguard.com]
Sent: Friday, August 09, 2002 1:09 PM
To: Ronald Clark
Cc: intrusions@incidents.org
Subject: Re: LOGS: GIAC GCIA Version 3.2 Practical Detect(s)

Interesting detect, beyond simply stating that this seems to be a reconnaissance scan can you explain why an unused port would be used? Also why would there be checksum errors? What kind of reconnaissance information would a black hat gain from these packets that he/she might not gain from normal packets destined for a well-known port?

Also in your defensive recommendation do you stand by the idea that this is innocent enough that we can ignore the packet in the future just because it doesn't try to reach an assigned port? You say it should be dropped by the firewall under network countermeasures, wouldn't that be a defensive step to make sure the rules are appropriate?

From: Antony Riley [mailto:antony@cyberiantiger.org]
Sent: Friday, August 09, 2002 6:29 PM
To: Paul Asadoorian
Cc: Ronald Clark; intrusions@incidents.org
Subject: Re: GIAC GCIA Version 3.2 Practical Detect(s)

```
> --------------------
> > 4. Description of attack: This is a port scan to tcp port 0 on the
> > target host(s). The attacker is trying to elicit some sort of response,
> most
> > likely for reconnaissance. I do not particularly think this could be a DOS
> > attempt due to the fact that there is just one packet every 3 - 12 seconds
> > and the attack only lasted for one minute and 56 seconds. (1:56)
> --------------------
>
> What kind of reconnaissance?  What information is the attacker trying to
> get?
>
```

I believe that on the hping2 website, they describe something called an 'idle scan', which involves three hosts, and the target which is being scanned only receives packets with a spoofed source address.

Anyway, the trace makes it look like the target machine was being used as the idle host, to scan another machine somewhere else.

Read the hping2 website more to find out more about this.

As a side note, hping3 will probably be out soon, so they might actually be able to do the right checksums.

(I use hping2 as a sys admin, for network debugging, and it's a very good tool for this purpose).

-Antony

My response:

Thanks to all who have replied. I have spent the last week or so looking into what I received back from this list and examining my own viewpoints again. I stated in my detect that there should be no response from the victim host, since I had thought that there was no tcp port zero. I think I may have been wrong.

I stated in my detect that I thought this could be used for some type of recon. The attacker could quite possibly map the network based on the responses. I guess the big question is would there be any response? One suggestion is that on a port zero scan, that if the attacker saw an ICMP port unreachable messages coming back, then he/she would be able to map which hosts were up. But, after thinking about this, I explained that this was a TCP type connection and ICMP port unreachable messages would not be seen here (they are for UDP).

According to "TCP/IP Illustrated - Volume 1 by Richard Stevens" on page 246-247, a RST is sent by TCP when a connection is attempted on a port that a service is not listening. No services listen on tcp port zero. Now, will some operating systems actually send a RST back? In theory, I think they all should. I tested this by loading hping2 and testing it against a Windows 2000 box, and Linux box and a FreeBSD box. All responded to the default port zero connection attempts with RST/ACK.

The one thing that did follow my trace was when I tried to corrupt or mismatch the checksums using the "-b" switch. When I used it in this command: "hping 172.16.a.b -b", I would get no response from the target system. This is what I believe I saw in the trace. But why? Could this be a very weak attempt at a Denial of Service? Or is it targeting a specific box that will be DOSed later? I believe this to be targeting.

Now why would someone scan to tcp port zero? As I stated in my detect, I believe this to be a beginning user of hping2, looking to verify a particular system is up. It could also be a somewhat experienced hping2 user looking to evade any intrusion detection systems.

As for the defensive recommendation, I completely agree with Bryce Alexander, that the defensive recommendation should at least verify the firewall rules so that anything destined for tcp port zero should be dropped and logged.

All of these suggestions and changes will make there way to my paper for submission. Thanks again to all that responded.

Thanks again,
Ron Clark

Detect #2:

```
19:37:08.924488 62.153.209.202.21 > 46.5.163.24.21: SF
2123544197:2123544197(0) win 1028
19:46:38.784488 62.153.209.202.21 > 46.5.173.22.21: SF
885714223:885714223(0) win 1028
19:50:50.934488 62.153.209.202.21 > 46.5.129.133.21: SF
469184435:469184435(0) win 1028
19:52:59.304488 62.153.209.202.21 > 46.5.178.187.21: SF
2048179336:2048179336(0) win 1028
```

```
20:20:47.244488 62.153.209.202.21 > 46.5.72.144.21: SF
1569011218:1569011218(0) win 1028
20:33:54.974488 62.153.209.202.21 > 46.5.205.181.21: SF
1351956840:1351956840(0) win 1028
20:51:42.814488 62.153.209.202.21 > 46.5.253.67.21: SF
812683767:812683767(0) win 1028
21:17:10.524488 62.153.209.202.21 > 46.5.77.246.21: SF
2114533128:2114533128(0) win 1028
23:09:04.544488 62.153.209.202.21 > 46.5.135.252.21: SF
315781345:315781345(0) win 1028
23:12:59.424488 62.153.209.202.21 > 46.5.235.208.21: SF
1860063256:1860063256(0) win 1028
```

1. Source of Trace: The source of this trace comes from http://www.incidents.org/logs/Raw. The above is a simple output from tcpdump showing the total scan.
2. Detect was generated by: This detect was generated by running the tcpdump output file through the Snort 1.8.6 engine using the default rule set. The rule that triggered this alert was:
   ```
   alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN
   FIN";flags:SF; reference:arachnids,198; classtype:attempted-
   recon; sid:6 24; rev:1;)
   ```
3. Probability the source address was spoofed: I would have to say that after doing analysis, this was made much easier. I think this source IP address is not spoofed. I believe this is a system compromised by the Ramen worm and trying to propagate across the Internet.
4. Description of the attack: This looks like a host has been compromised by the Ramen worm and is now trying to propagate the worm by scanning other hosts looking for vulnerable FTP server daemons. The attack here seems slow. Is the host scanning others as well and we do not see that traffic? Here is the last four alerts generated by Snort:

```
[**] SCAN SYN FIN [**]
07/08-20:20:47.244488 62.153.209.202:21 -> 46.5.72.144:21
TCP TTL:30 TOS:0x0 ID:39426 IPLen:20 DgmLen:40
******SF Seq: 0x5D853612  Ack: 0x3AFA42A0  Win: 0x404  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN SYN FIN [**]
07/08-20:33:54.974488 62.153.209.202:21 -> 46.5.205.181:21
TCP TTL:30 TOS:0x0 ID:39426 IPLen:20 DgmLen:40
******SF Seq: 0x50953968  Ack: 0x7AD735CE  Win: 0x404  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN SYN FIN [**]
07/08-20:51:42.814488 62.153.209.202:21 -> 46.5.253.67:21
TCP TTL:30 TOS:0x0 ID:39426 IPLen:20 DgmLen:40
******SF Seq: 0x307091F7  Ack: 0xFE8A03E  Win: 0x404  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN SYN FIN [**]
07/08-21:17:10.524488 62.153.209.202:21 -> 46.5.77.246:21
TCP TTL:30 TOS:0x0 ID:39426 IPLen:20 DgmLen:40
******SF Seq: 0x7E093708  Ack: 0x275318CB  Win: 0x404  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

And then the payload from two of the packets:

```
23:09:04.544488 62.153.209.202.21 > 46.5.135.252.21: SF [bad tcp cksum f9f9!]
315781345:315781345(0) win 1028 (ttl 30, id 39426, len 40, bad cksum 426f!)
0x0000   4500 0028 9a02 0000 1e06 426f 3e99 d1ca        E..(......Bo>...
0x0010   2e05 87fc 0015 0015 12d2 70e1 1967 3cf8        ..........p..g<.
0x0020   5003 0404 1142 0000 0000 0000 0000             P....B........
23:12:59.424488 62.153.209.202.21 > 46.5.235.208.21: SF [bad tcp cksum f9f9!]
1860063256:1860063256(0) win 1028 (ttl 30, id 39426, len 40, bad cksum de9a!)
0x0000   4500 0028 9a02 0000 1e06 de9a 3e99 d1ca        E..(........>...
0x0010   2e05 ebd0 0015 0015 6ede 5018 22b4 7476        ........n.P.".tv
0x0020   5003 0404 315f 0000 0000 0000 0000             P...1_........
```

There are also bad TCP checksums and the source and destination ports are both 21.

5. Attack Mechanism: This looks like the modified synscan mechanism that was part of the Ramen worm. This worm would start a scan based on these criteria:

   1) Various source IP addresses
   2) TCP source port 21, destination port 21
   3) Type of service 0
   4) IP identification number 39426
   5) SYN and FIN flags set
   6) Various sequence numbers set
   7) Various acknowledgment numbers set
   8) TCP window size 1028

   This detect shows 7 of the 8 items listed above.

6. Correlations: The SecurityFocus website shows good information about this worm and its characteristics. http://online.securityfocus.com/infocus/1524.
   Also, there is more information on the worm itself at McAfee. http://vil.nai.com/vil/content/v_98975.htm
   Stephen Northcutt and Judy Novak describe this type of attack (SYN/FIN) in the book Network Intrusion Detection (pages 226-229 and 239-240).

7. Evidence of active targeting: There was not any one specific system that was targeted. I believe this to be the case of a system infected by the Ramen worm trying to infect other systems at random. This is substantiated by a complete analysis of the Ramen Worm and its propagation method: http://www.whitehats.com/library/worms/ramen/. In this particular netblock sweep, these servers are probably the only true ftp servers that are accessed from the Internet. Therefore the firewall would have allowed these packets into the network.

8. Severity: (criticality 3 + lethality 2) – (system countermeasures 3 + network countermeasures 3) = severity 2
   Explanation:
   Criticality 3 – FTP servers could be considered critical on this network.
   Lethality 2 – I give this a rating of 2. This worm does not seem to have any backdoors. (at least from what I have read)
   System Countermeasures 3 – This is a 3. We assume the system is up to date on all patches, but without physical access, we cannot verify this.
   Network Countermeasures 3 – I rate this 3 as well. If we assume a somewhat restrictive firewall in place, this could explain the reason why only a few sparse IP addresses are seeing this traffic.
9. Defensive recommendation: Review network infrastructure, locate any servers running the FTP service and perform vulnerability scan, patch as needed. Review and test firewall rules if needed. Also, a phone call or an email to the owner of the system or the ISP might alert the owner of the PC that they have been compromised.
10. Multiple choice question:

```
19:37:08.924488 62.153.209.202.21 > 46.5.163.24.21: SF
2123544197:2123544197(0) win 1028
19:46:38.784488 62.153.209.202.21 > 46.5.173.22.21: SF
885714223:885714223(0) win 1028
19:50:50.934488 62.153.209.202.21 > 46.5.129.133.21: SF
469184435:469184435(0) win 1028
```

   Based on the logs above, what aspects of the Ramen worm are present?

   a) Source and destination ports are both 21
   b) Window size of 1028
   c) SYN and FIN flags are both set
   d) FTP
   e) Answers a), b), and c)

   Answer: e) Answers a), b), and c)

Detect #3:
I will start this detect section off by saying that this next trace comes from my home network. I had opened up the firewall to allow for all traffic instead of setting up the appropriate FTP rules for the firewall. With this information in mind here is the trace:

```
19:48:24.157030 61.187.121.68.4637 > MY.HOME.NET.245.8000: S 2499680372:2499680372(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:24.157111 61.187.121.68.4635 > MY.HOME.NET.245.81: S 2499580884:2499580884(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:24.157206 61.187.121.68.4636 > MY.HOME.NET.245.8080: S 2499638430:2499638430(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:24.157305 61.187.121.68.4638 > MY.HOME.NET.245.3128: S 2499722243:2499722243(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
```

```
19:48:24.168881 MY.HOME.NET.245.8000 > 61.187.121.68.4637: R 0:0(0) ack 2499680373 win 0
19:48:24.170568 MY.HOME.NET.245.81 > 61.187.121.68.4635: R 0:0(0) ack 2499580885 win 0
19:48:24.172412 MY.HOME.NET.245.8080 > 61.187.121.68.4636: R 0:0(0) ack 2499638431 win 0
19:48:24.174020 MY.HOME.NET.245.3128 > 61.187.121.68.4638: R 0:0(0) ack 2499722244 win 0
19:48:25.092042 61.187.121.68.4638 > MY.HOME.NET.245.3128: S 2499722243:2499722243(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:25.094916 61.187.121.68.4635 > MY.HOME.NET.245.81: S 2499580884:2499580884(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:25.095153 61.187.121.68.4636 > MY.HOME.NET.245.8080: S 2499638430:2499638430(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:25.095884 61.187.121.68.4637 > MY.HOME.NET.245.8000: S 2499680372:2499680372(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:25.097525 MY.HOME.NET.245.3128 > 61.187.121.68.4638: R 0:0(0) ack 1 win 0
19:48:25.105163 MY.HOME.NET.245.81 > 61.187.121.68.4635: R 0:0(0) ack 1 win 0
19:48:25.106749 MY.HOME.NET.245.8080 > 61.187.121.68.4636: R 0:0(0) ack 1 win 0
19:48:25.108459 MY.HOME.NET.245.8000 > 61.187.121.68.4637: R 0:0(0) ack 1 win 0
19:48:26.017437 61.187.121.68.4638 > MY.HOME.NET.245.3128: S 2499722243:2499722243(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:26.017521 61.187.121.68.4635 > MY.HOME.NET.245.81: S 2499580884:2499580884(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:26.017618 61.187.121.68.4636 > MY.HOME.NET.245.8080: S 2499638430:2499638430(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:26.017715 61.187.121.68.4637 > MY.HOME.NET.245.8000: S 2499680372:2499680372(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:26.028010 MY.HOME.NET.245.3128 > 61.187.121.68.4638: R 0:0(0) ack 1 win 0
19:48:26.029693 MY.HOME.NET.245.81 > 61.187.121.68.4635: R 0:0(0) ack 1 win 0
19:48:26.031274 MY.HOME.NET.245.8080 > 61.187.121.68.4636: R 0:0(0) ack 1 win 0
19:48:26.033166 MY.HOME.NET.245.8000 > 61.187.121.68.4637: R 0:0(0) ack 1 win 0

[**] INFO - Possible Squid Scan [**]
08/04-19:48:24.157305 61.187.121.68:4638 -> MY.HOME.NET.245:3128
TCP TTL:109 TOS:0x0 ID:56731 IPLen:20 DgmLen:48 DF
******S* Seq: 0x94FEBC03  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] INFO - Possible Squid Scan [**]
08/04-19:48:25.092042 61.187.121.68:4638 -> MY.HOME.NET.245:3128
TCP TTL:109 TOS:0x0 ID:56764 IPLen:20 DgmLen:48 DF
******S* Seq: 0x94FEBC03  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] INFO - Possible Squid Scan [**]
08/04-19:48:26.017437 61.187.121.68:4638 -> MY.HOME.NET.245:3128
TCP TTL:109 TOS:0x0 ID:56841 IPLen:20 DgmLen:48 DF
******S* Seq: 0x94FEBC03  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

19:48:26.017437 61.187.121.68.4638 > MY.HOME.NET.245.3128: S [tcp sum ok]
2499722243:2499722243(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 109, id 56841, len
48)
0x0000   4500 0030 de09 4000 6d06 d6fb 3dbb 7944        E..0..@.m...=.yD
0x0010   MYHM NTf5 121e 0c38 94fe bc03 0000 0000        .......8........
0x0020   7002 4000 7b04 0000 0204 05b4 0101 0402        p.@.{...........
19:48:26.017521 61.187.121.68.4635 > MY.HOME.NET.245.81: S [tcp sum ok]
2499580884:2499580884(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 109, id 56842, len
48)
0x0000   4500 0030 de0a 4000 6d06 d6fa 3dbb 7944        E..0..@.m...=.yD
0x0010   MYHM NTf5 121b 0051 94fc 93d4 0000 0000        .......Q........
0x0020   7002 4000 af1f 0000 0204 05b4 0101 0402        p.@.............
19:48:26.017618 61.187.121.68.4636 > MY.HOME.NET.245.8080: S [tcp sum ok]
2499638430:2499638430(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 109, id 56843, len
48)
0x0000   4500 0030 de0b 4000 6d06 d6f9 3dbb 7944        E..0..@.m...=.yD
0x0010   MYHM NTf5 121c 1f90 94fd 749e 0000 0000        ..........t.....
```

```
0x0020   7002 4000 af14 0000 0204 05b4 0101 0402        p.@............
19:48:26.017715 61.187.121.68.4637 > MY.HOME.NET.245.8000: S [tcp sum ok]
2499680372:2499680372(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 109, id 56844, len
48)
0x0000   4500 0030 de0c 4000 6d06 d6f8 3dbb 7944        E..0..@.m...=.yD
0x0010   MYHM NTf5 121d 1f40 94fe 1874 0000 0000        .......@...t....
0x0020   7002 4000 0b8d 0000 0204 05b4 0101 0402        p.@............
```

1. Source of Trace: The source of this trace is my home network that is connected to the Internet via cable modem.
2. Detect generated by: This detect was generated by Snort 1.8.6 using the default rule set. The rules that triggered the alert were:
   ```
   alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"INFO -
   Possible Squid Scan"; flags:S; classtype:attempted-recon;
   sid:618; rev:1;) and
   alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy
   attempt";flags:S; classtype:attempted-recon; sid:620; rev:1;)
   ```
   I have also shown some of the snort alerts as well as several packet decodes of the IP headers using tcpdump.
3. Probability the source was spoofed: The attacker would be expecting some kind of response in such a scan, therefore, spoofing the source address would increase the complexity of the attacker actually getting a response. For this reason I think it is possible, but highly unlikely, that this source address is spoofed.
4. Description of the Attack: This seems like a simple port scan looking for vulnerable Squid Proxy servers (port 3128 and 8080), Transproxy (port 81) and Microsoft ISA server, which uses all of these ports. The troubling part is that these proxy ports do not all run on Windows. Transproxy and Squid run on FreeBSD and Linux systems, while Microsoft's ISA server runs on Windows NT and 2000.
5. Attack Mechanism: This attack, if successful, could take one of two routes. An attacker could try to gain root access through the numerous vulnerabilities in proxy server software. Or the attacker could try to spoof his source address and fool the proxy server to let him view the systems behind it or to compromise the host and launch other hacks or DOS attacks.
6. Correlations: There was a similar type of scan behavior on June 23rd at http://online.securityfocus.com/archive/75/278371/2002-06-24/2002-06-30/2 and the post by Carolyn Curl confirms the span of common proxy ports. http://www.incidents.org/archives/intrusions/msg12019.html
7. Evidence of active targeting: I do not think that I personally was actively targeted. I do believe however that the attacker was looking for home networks with poorly configured proxy servers that could be easily exploited.
8. Severity = (criticality + lethality) – (system countermeasures – network countermeasures)
   Criticality = 5 – This is the Firewall for my home network.
   Lethality = 3 – This attack will not succeed, but it does generate some responses, which lets the attacker know I am up and alive.

System Countermeasures = 5 – Latest version of OS and all patches applied.
Network Countermeasures = 2 – Opened up firewall.
Severity = 1.

9. Defensive Recommendation: Tighten up the firewall rules. If specific traffic is needed to pass through for the internal network, then add those specific rules to the firewall and test.
10. Multiple Choice Question

```
19:48:24.157030 61.187.121.68.4637 > MY.HOME.NET.245.8000: S
2499680372:2499680372(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:24.157111 61.187.121.68.4635 > MY.HOME.NET.245.81: S
2499580884:2499580884(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:24.157206 61.187.121.68.4636 > MY.HOME.NET.245.8080: S
2499638430:2499638430(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
19:48:24.157305 61.187.121.68.4638 > MY.HOME.NET.245.3128: S
2499722243:2499722243(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
```

Based on the above trace, what service is the attacker looking for?

a) ftp
b) http
c) smtp
d) proxy

Answer: d) proxy

Assignment #3 – Analyze This

Executive Summary

This University operates its network with one or more points of presence on the Internet. The University also uses the Snort Intrusion Detection system to help protect the network from unlawful intrusions. Based on the five days worth of Snort logs provided, it has been determined there is still much work to be done.

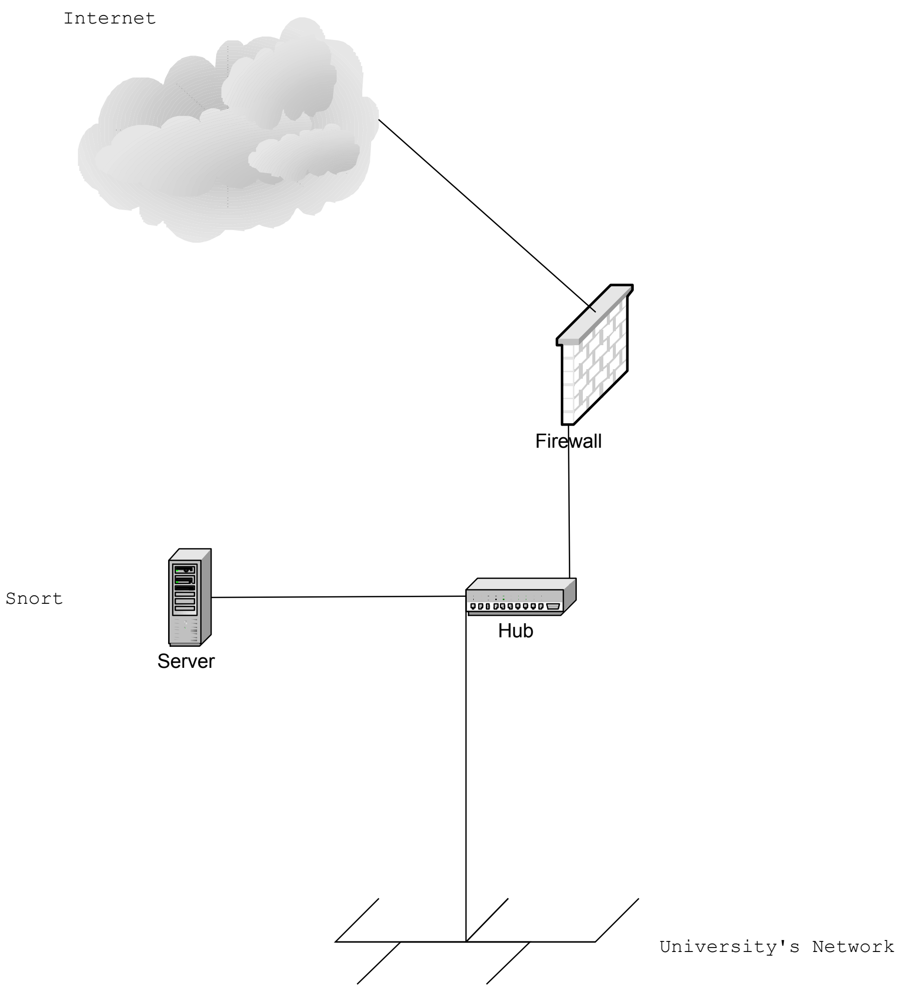
Universities and their network infrastructures are designed to be used for a variety of purposes. They must first promote learning and research by the students and faculty alike. In terms of security, they are probably set up like ISP's, with little to no security in an "anything goes" type of environment. While great for the average student, to be able to play games, email family and friends, and other miscellaneous activities, it can potentially be dangerous for the

University and its network. It can create bandwidth issues including but not limited to peer-to-peer file swapping, gaming, and other miscellaneous traffic. It can create security issues in term of compromised systems, viruses, and the spread of other malware.

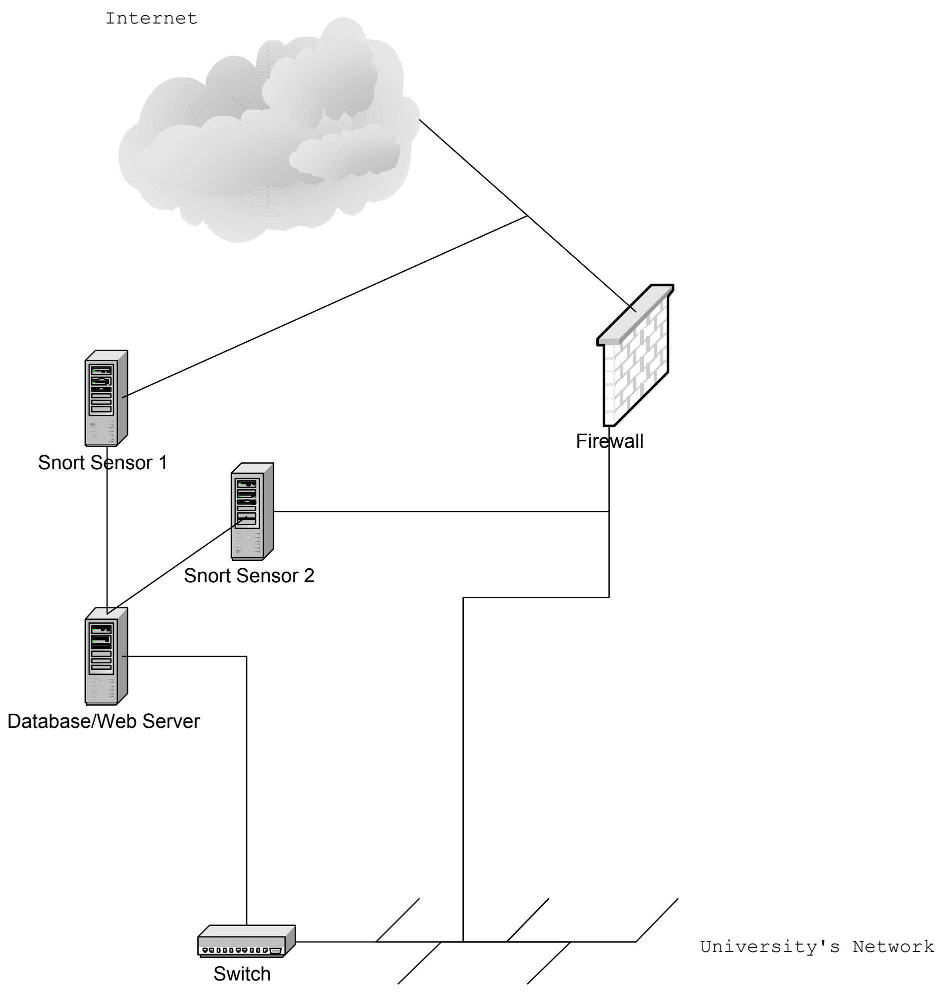
What I have attempted to do is to list what I believe to be the issues that need immediate attention. There is evidence of internal systems on the University network that have been compromised, and are attempting to compromise others. These need to be dealt with as soon as possible. I believe that if these issues are dealt with and fixed, intrusion detection systems tweaked for efficiency, and firewall tuned for greater security while not hampering “good” legitimate traffic, then the job of sorting through log files will become a great deal easier.

Defensive Recommendations Overall:

- Though there is no evidence a firewall exists, it is assumable one is in place. If so, the rules need reviewing and additions or changes made to protect the internal network from Trojan activity and general scans.
- The Snort Intrusion Detection system needs reviewing as well and the signatures updated to make the system more efficient. There is a lot of information being alerted and logged that has to do with trusted internal systems. (DNS for example) If Snort is reconfigured and rules tweaked it should begin to report on the real intrusions that are taking place and allow for those issues to be a priority and dealt with accordingly.
- IDS Infrastructure could be expanded. I consider the current infrastructure for IDS looks like the following graphic.



I propose that the IDS infrastructure be upgraded to encompass at least two sensors, one outside the firewall and one inside the firewall. Having these two log to a central database and using a web based front end would enable the system admin a much easier time of sorting trough the alerts. A diagram of the improved IDS infrastructure would also look like the following graphic.



In this setup, the Snort sensors and the Database/Web server are on a separate network from the University's network. All sensors are dual homed hosts. Each of the snort sensors has an interface that is not assigned an IP address that is listening for traffic coming in and leaving the network. The other interfaces are addressed in a private LAN, like 10.0.0.0/28, to allow reporting to the database server. The database server also hosts a web front end like SnortSnarf or Demarc, which could be used for reporting, analysis, etc. This would make the IDS process much more efficient for the system administrator.

- Proactive vulnerability scanning from inside the network to determine any new or missed vulnerabilities. This should be done at least once a month.
- Independent Audit / Penetration test from qualified third party every 3–6 months.

Analysis Source files

Snort logs for five consecutive days were used for the analysis. For each day, there are three files that must be looked at. They are scan files, alert files and out of spec files. They are listed below.

Scans	Alerts	OOS
scans.020801.txt	alert.020801.txt	oos_Aug.1.2002.txt
scans.020802.txt	alert.020802.txt	oos_Aug.2.2002.txt
scans.020803.txt	alert.020803.txt	oos_Aug.3.2002.txt
scans.020804.txt	alert.020804.txt	oos_Aug.4.2002.txt
scans.020805.txt	alert.020805.txt	oos_Aug.5.2002.txt

The Analysis:

The alerts file produced the following stats:

Alert Stats for table -> alerts	
Total number of incidents:	**2236823**
Breakout by type:	
NIMDA - Attempt to execute cmd from campus host = **877538**	0% 39% 100%
spp_http_decode: IIS Unicode attack detected = **494119**	0% 22% 100%

IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize = **482402**	0% 21% 100%
NIMDA - Attempt to execute root from campus host = **123305**	0% 5% 100%
UDP SRC and DST outside network = **106883**	0% 4% 100%
spp_http_decode: CGI Null Byte attack detected = **53562**	0% 2% 100%
SMB Name Wildcard = **30083**	0% 1% 100%

TFTP - External UDP connection to internal tftp server = **24220**	0% 1% 100%
External RPC call = **14578**	0% <1% 100%
Watchlist 000220 IL-ISDNNET-990517 = **11921**	0% <1% 100%
Possible Trojan server activity = **4113**	0% <1% 100%
SUNRPC highport access! = **2543**	0% <1% 100%

IRC evil - running XDCC = **2054**	0% <1% 100%
Watchlist 000222 NET-NCFC = **1305**	0% <1% 100%
EXPLOIT x86 NOOP = **1293**	0% <1% 100%
Queso fingerprint = **1120**	0% <1% 100%
SNMP public access = **927**	0% <1% 100%

connect to 515 from outside = **788**	0% <1% 100%
Attempted Sun RPC high port access = **730**	0% <1% 100%
Samba client access = **679**	0% <1% 100%
High port 65535 udp - possible Red Worm - traffic = **628**	0% <1% 100%
IDS552/web-iis_IIS ISAPI Overflow ida nosize = **314**	0% <1% 100%

ICMP SRC and DST outside network = **260**	0% <1% 100%
SMB C access = **236**	0% <1% 100%
TFTP - Internal UDP connection to external tftp server = **173**	0% <1% 100%
beetle.ucs = **166**	0% <1% 100%
Port 55850 tcp - Possible myserver activity - ref. 010313-1 = **147**	0% <1% 100%

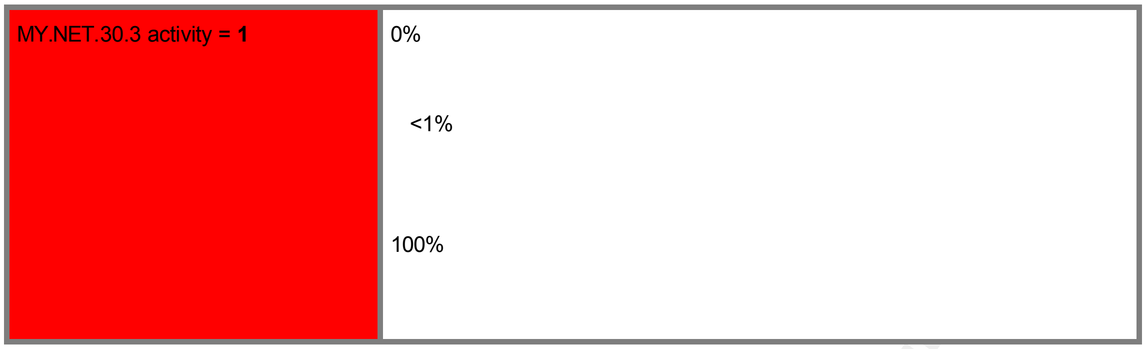
Incomplete Packet Fragments Discarded = **136**	0% <1% 100%
Null scan! = **106**	0% <1% 100%
NMAP TCP ping! = **88**	0% <1% 100%
EXPLOIT x86 setuid 0 = **58**	0% <1% 100%
Tiny Fragments - Possible Hostile Activity = **53**	0% <1% 100%

EXPLOIT x86 stealth noop = **48**	0% <1% 100%
High port 65535 tcp - possible Red Worm - traffic = **44**	0% <1% 100%
STATDX UDP attack = **42**	0% <1% 100%
EXPLOIT x86 setgid 0 = **38**	0% <1% 100%
Port 55850 udp - Possible myserver activity - ref. 010313-1 = **18**	0% <1% 100%

SMB CD... = **13**	0% <1% 100%
TCP SRC and DST outside network = **13**	0% <1% 100%
HelpDesk MY.NET.70.50 to External FTP = **11**	0% <1% 100%
External FTP to HelpDesk MY.NET.70.50 = **11**	0% <1% 100%
MY.NET.30.4 activity = **11**	0% <1% 100%

HelpDesk MY.NET.70.49 to External FTP = **9**	0% <1% 100%
External FTP to HelpDesk MY.NET.70.49 = **8**	0% <1% 100%
TFTP - External TCP connection to internal tftp server = **6**	0% <1% 100%
EXPLOIT NTPDX buffer overflow = **5**	0% <1% 100%
HelpDesk MY.NET.83.197 to External FTP = **4**	0% <1% 100%

Back Orifice = **3**	0% <1% 100%
DDOS shaft client to handler = **3**	0% <1% 100%
RFB - Possible WinVNC - 010708-1 = **3**	0% <1% 100%
Traffic from port 53 to port 123 = **2**	0% <1% 100%
SYN-FIN scan! = **2**	0% <1% 100%



Notable Alert Events:

The following alert detects were chosen based on the volume of alerts they produced. The Nimda and IIS alerts were grouped together at the beginning.

Nimda

The first thing I noticed is that Nimda related alerts account for approximately 44% of the 2.2 million total alerts.

```
08/02-13:46:45.559476  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.82.87:4901 -> 207.46.235.162:80
08/02-17:44:33.001172  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.70.16:2142 -> 65.54.250.121:80
```

According to our logs NIMDA alerts do not begin until August 2nd. What is more interesting is that they are from an internal host going to a host out on the Internet. Yet we see no evidence of the NIMDA coming in to these hosts. For the next two days we only see three more NIMDA alerts.

```
08/03-19:55:53.607645  [**] NIMDA - Attempt to execute cmd from
campus host [**] MY.NET.83.176:1345 -> 207.68.132.9:80
08/04-14:23:02.748399  [**] NIMDA - Attempt to execute cmd from
campus host [**] MY.NET.111.30:1092 -> 207.46.235.150:80
08/04-14:45:06.913040  [**] NIMDA - Attempt to execute cmd from
campus host [**] MY.NET.165.19:1085 -> 65.54.250.120:80
```

Then on August 5th NIMDA goes off the charts. Let's look at some of the alerts.

```
08/05-09:14:50.113555  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.70.169:1103 -> 65.54.250.121:80
08/05-13:22:36.967751  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.70.144:1116 -> 207.46.235.150:80
08/05-14:45:08.318570  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.105.10:1534 -> 65.54.250.120:80
08/05-16:15:30.931350  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.130.20:1069 -> 65.54.250.121:80
08/05-21:21:55.661920  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2008 -> 130.95.40.191:80
08/05-21:21:55.664339  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2010 -> 130.7.64.55:80
08/05-21:21:55.670339  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2009 -> 130.178.180.123:80
```

```
08/05-21:21:55.670567  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2011 -> 130.91.203.243:80
08/05-21:21:55.677068  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2015 -> 130.62.62.95:80
08/05-21:21:55.679411  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2012 -> 130.95.40.191:80
08/05-21:21:55.679660  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2013 -> 130.7.64.55:80
08/05-21:21:55.686082  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2017 -> 130.217.61.115:80
08/05-21:21:55.686319  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2014 -> 130.178.180.123:80
08/05-21:21:55.693966  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2020 -> 130.117.60.135:80
08/05-21:21:55.695427  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.100.208:2016 -> 130.7.64.55:80
08/05-21:21:55.696037  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.100.208:2021 -> 130.95.40.191:80
08/05-21:21:55.698463  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.100.208:2018 -> 130.178.180.123:80
08/05-21:21:55.699113  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2022 -> 130.217.61.115:80
08/05-21:21:55.701159  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2019 -> 130.62.62.95:80
08/05-21:21:55.704732  [**] NIMDA - Attempt to execute root from campus host [**]
MY.NET.100.208:2023 -> 130.17.60.155:80
```

The first four alerts look like the same type of alerts from the previous three days, i.e. a couple of hosts looking to infect other vulnerable hosts on the Internet. The volume begins at approximately 9:21pm when the host MY.NET.100.208 begins scanning the 130.x.x.x network looking for hosts to compromise.

Severity:
Based on the formula **(Criticality + Lethality) – (System + Net Countermeasures) = Severity** and what we think is the way the University has their network laid out we would rate this particular compromise as **SEVERE**. For Criticality we assess that this is a server that is running the IIS web service. Criticality is **3**. For Lethality we know that the system has been compromised and is now trying to propagate the worm. Lethality is **5**. For System countermeasures we do know that this box did not have all of the needed patches and probably could assume that it did not have any. System countermeasures are **1**. For Net countermeasures we would like to assume that there is a firewall in place controlling traffic leaving and entering the network but we just do not know because we have no physical access to this network. If there is a firewall in place it seems safe to say that it is open. Network Countermeasures are **2**. So that leaves us with **(3+5) – (1+2) = Severity of 5**.

Defensive Recommendation:
Based on the traces above and the evidence of systems being compromised, defensive recommendations are:

- Review all written security policies with staff and students. If non-existent, implement as soon as possible.

- Fdisk and format all compromised IIS servers currently on the network. Before any new IIS servers are connected, verify patches and/or run through a security checklist. If there is not a security checklist then one needs to be implemented as well and put into place as soon as possible.
- Apply outgoing rule(s) to the firewall to prevent the spread of infection across the Internet.
- Monitor the logs daily, both IDS and firewall.

IIS attacks

Next in line, based on sheer volume, are the IIS Unicode attacks. IIS attacks account for approximately 44% of the total alerts. There are three different types:
IIS Unicode attack – 494119 alerts
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize – 482402
IDS552/web-iis_IIS ISAPI Overflow ida nosize – 314

The first notable issue is that there are 482402 alerts from systems inside the network! We will determine the exact systems soon and recommend a course of action to clean up these alerts. I would like to examine the IIS Unicode and the ISAPI coming from the outside.

The IIS Unicode attack generated almost a half a million alerts in five days. The following snip from the Snort logs shows us the following:

```
./alert.020801.txt:08/01-00:10:57.452228  [**] spp_http_decode: IIS Unicode
attack detected [**] 64.86.155.118:2672 -> MY.NET.109.87:80
./alert.020801.txt:08/01-00:10:59.910726  [**] spp_http_decode: IIS Unicode
attack detected [**] 64.86.155.118:2710 -> MY.NET.109.87:80
./alert.020801.txt:08/01-00:21:07.087017  [**] spp_http_decode: IIS Unicode
attack detected [**] 211.91.255.154:51337 -> MY.NET.53.84:80
./alert.020801.txt:08/01-00:21:08.346841  [**] spp_http_decode: IIS Unicode
attack detected [**] 211.91.255.154:51343 -> MY.NET.53.84:80
./alert.020801.txt:08/01-00:22:55.947869  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.183.25:4413 -> 64.12.42.116:80
./alert.020801.txt:08/01-00:22:55.947869  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.183.25:4413 -> 64.12.42.116:80
./alert.020801.txt:08/01-00:22:55.947869  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.183.25:4413 -> 64.12.42.116:80
./alert.020801.txt:08/01-00:22:56.362352  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.183.25:4415 -> 64.12.42.116:80
./alert.020801.txt:08/01-00:22:56.362352  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.183.25:4415 -> 64.12.42.116:80
```

In this snip of logs we have a couple of external hosts scanning the University network looking for other hosts to compromise. This is followed by the traffic generated by an internal host looking for other hosts to compromise. While this is not a direct reply to the previous external traffic it is disconcerting.

Anatomy of a compromise

Since the IIS and the Nimda alerts account for so many of the total number of alerts it should be obvious as to who caused the alerts to begin, as well as who

infected whom.
McAfee's website (http://vil.mcafee.com/dispVirus.asp?virus_k=99209) describes Nimda as the following:

The worm scans IP addresses looking for IIS servers to infect via the Web Folder Transversal vulnerability by sending a malformed GET request. This causes vulnerable machines to initiate a TFTP session to download ADMIN.DLL from the machine which sent the request. Once downloaded the remote system is instructed to execute the DLL which infects that machine. In the event that the TFTP session fails to connect, Multiple files (TFTP*) are created in the WINDOWS TEMP directory. These files are simply copies of the worm. It also tries to use the backdoor created by W32/CodeRed.c to infect.

On August 5th there are multiple entries from external hosts to MY.NET.100.208. The sources are charted as follows:

Attacking Source IP address'
61.189.144.7
66.30.130.39
65.162.184.6
218.68.217.156
61.145.69.74
211.141.120.18

The first four attacking IP addresses look like they are trying to initiate a TFTP session back to a 192.168.0.1 address, which may or may not be working. Then MY.NET.100.208 gets hit with a Unicode attack from 61.189.144.7. This causes MY.NET.100.208 to start a TFTP session back to 61.189.144.7. Here is the trace:

```
08/05-20:46:04.774542  [**] spp_http_decode: IIS Unicode attack
detected [**] 61.145.69.74:4807 -> MY.NET.100.208:80
08/05-20:46:05.012671  [**] TFTP - Internal UDP connection to
external tftp server [**] MY.NET.100.208:1913 ->
61.145.69.74:69
08/05-20:41:28.807450  [**] TFTP - Internal UDP connection to
external tftp server [**] MY.NET.100.208:1906 ->
61.145.69.74:69
```

After which MY.NET.100.208 starts a TFTP session with 211.141.120.18:

```
08/05-20:55:14.454687  [**] TFTP - Internal UDP connection to
external tftp server [**] MY.NET.100.208:1923 ->
61.145.69.74:69
08/05-20:55:14.606498  [**] TFTP - Internal UDP connection to
external tftp server [**] MY.NET.100.208:1923 ->
61.145.69.74:69
08/05-21:20:59.260155  [**] TFTP - Internal UDP connection to
```

```
external tftp server [**] MY.NET.100.208:1925 ->
211.141.120.18:69
08/05-21:20:59.277400  [**] TFTP - Internal UDP connection to
external tftp server [**] MY.NET.100.208:1925 ->
211.141.120.18:69
08/05-21:21:00.348370  [**] TFTP - Internal UDP connection to
external tftp server [**] MY.NET.100.208:1926 ->
211.141.120.18:69
```

Then the Unicode attack from 211.141.120.18:

```
08/05-21:21:06.650638  [**] spp_http_decode: IIS Unicode attack
detected [**] 211.141.120.18:4853 -> MY.NET.100.208:80
08/05-21:21:06.650638  [**] spp_http_decode: IIS Unicode attack
detected [**] 211.141.120.18:4853 -> MY.NET.100.208:80
08/05-21:21:06.650638  [**] spp_http_decode: IIS Unicode attack
detected [**] 211.141.120.18:4853 -> MY.NET.100.208:80
```

The compromise is completed and the scanning starts:

```
08/05-21:21:11.439315  [**] TFTP - Internal UDP connection to
external tftp server [**] MY.NET.100.208:1930 ->
211.141.120.18:69
08/05-21:21:11.590781  [**] TFTP - Internal UDP connection to
external tftp server [**] MY.NET.100.208:1930 ->
211.141.120.18:69
08/05-21:21:14.056953  [**] spp_http_decode: IIS Unicode attack
detected [**] 211.141.120.18:3218 -> MY.NET.100.208:80
08/05-21:21:14.845654  [**] spp_http_decode: IIS Unicode attack
detected [**] 211.141.120.18:3236 -> MY.NET.100.208:80
08/05-21:21:15.808352  [**] spp_http_decode: IIS Unicode attack
detected [**] 211.141.120.18:3249 -> MY.NET.100.208:80
08/05-21:21:16.092822  [**] TFTP - Internal UDP connection to
external tftp server [**] MY.NET.100.208:1931 ->
211.141.120.18:69
08/05-21:21:55.661920  [**] NIMDA - Attempt to execute root
from campus host [**] MY.NET.100.208:2008 -> 130.95.40.191:80
08/05-21:21:55.664339  [**] NIMDA - Attempt to execute root
from campus host [**] MY.NET.100.208:2010 -> 130.7.64.55:80
08/05-21:21:55.670339  [**] NIMDA - Attempt to execute root
from campus host [**] MY.NET.100.208:2009 -> 130.178.180.123:80
08/05-21:21:55.670567  [**] NIMDA - Attempt to execute root
from campus host [**] MY.NET.100.208:2011 -> 130.91.203.243:80
```

All the alerts are not posted, for the sake of space. There is clear evidence of what took place. Once MY.NET.100.208 was compromised it began to scan for other hosts and generated the large number of alerts.

Attacked by whom?
Now we have the two culprits that are outside of our network that caused such an annoyance. Let's see what information we can obtain from their IPs. First, we will go to http://www.apnic.net/search/index.html. Here we established:

61.145.69.74

```
% [whois.apnic.net node-1]
% How to use this server         http://www.apnic.net/db/
% Whois data copyright terms     http://www.apnic.net/db/dbcopyright.html
inetnum:      61.145.0.0 - 61.145.255.255
netname:      CHINANET-GD
descr:        CHINANET Guangdong province network
descr:        Data Communication Division
descr:        China Telecom
country:      CN
admin-c:      CH93-AP
tech-c:       WM12-AP
mnt-by:       MAINT-CHINANET
mnt-lower:    MAINT-CHINANET-GD
changed:      hostmaster@ns.chinanet.cn.net 20000701
status:       ALLOCATED PORTABLE
source:       APNIC
person:       Chinanet Hostmaster
address:      No.31 ,jingrong street,beijing
address:      100032
country:      CN
phone:        +86-10-66027112
fax-no:       +86-10-66027334
e-mail:       hostmaster@ns.chinanet.cn.net
nic-hdl:      CH93-AP
mnt-by:       MAINT-CHINANET
changed:      hostmaster@ns.chinanet.cn.net 20020814
source:       APNIC
person:       WU MIAN
address:      NO.1,RO.DONGYUANHENG,YUEXIUNAN,GUANGZHOU
country:      CN
phone:        +086-20-83877223
fax-no:       +86-20-83877223
e-mail:       IPadm@gddc.com.cn
nic-hdl:      WM12-AP
mnt-by:       MAINT-CHINANET-GD
changed:      IPadm@gddc.com.cn 20010820
source:       APNIC
```

211.141.120.18

```
% [whois.apnic.net node-2]
% How to use this server         http://www.apnic.net/db/
% Whois data copyright terms     http://www.apnic.net/db/dbcopyright.html
inetnum:      211.141.120.16 - 211.141.120.31
netname:      JAWMNET
descr:        WUMING Network bar =20
descr:        provide Internet accessing service
descr:        JI AN city, Jiangxi Province=20
country:      CN
admin-c:      XW45-AP
tech-c:       SS170-AP
mnt-by:       MAINT-CN-CMCC
changed:      hostmaster@chinamobile.com 20011128
status:        ASSIGNED NON-PORTABLE
source:       APNIC
changed:      hm-changed@apnic.net  20020827
person:       Xiaoyun Wang
address:      53A,Xibianmennei Ave.,Xuanwu District,Beijing,100053 China
country:      CN
phone:        +86-10-6360-0159
fax-no:       +86-10-6360-0117
```

```
e-mail:        wangxiaoyun@chinamobile.com
nic-hdl:       XW45-AP
mnt-by:        MAINT-CN-CMCC
changed:       sunshaoling@chinamobile.com 20010831
source:        APNIC
person:        Shaoling Sun
address:       53A,Xibianmennei Ave.,Xuanwu District,Beijing,100053 China
country:       CN
phone:         +86-10-6360-0159
fax-no:        +86-10-6360-0117
e-mail:        sunshaoling@chinamobile.com
nic-hdl:       SS170-AP
mnt-by:        MAINT-CN-CMCC
changed:       sunshaoling@chinamobile.com 20010831
source:        APNIC
```

Severity:
Based on the formula **(Criticality + Lethality) – (System + Net Countermeasures) = Severity** and what we think is the way the University has their network laid out we would rate this particular compromise as **SEVERE**. For Criticality we assess that this is a web server but we do not know if it is the University's main web server or a rogue one put up by a student. Criticality is **3**. For Lethality we know that the system has been compromised and is now being used to try and compromise the rest of the University Network. Lethality is **5**. For System countermeasures we do know that this box did not have all of the needed patches and probably could assume that it did not have any. System countermeasures are **1**. For Net countermeasures we would like to assume that there is a firewall in place controlling traffic leaving and entering the network but we just do not know because we have no physical access to this network. If there is a firewall in place it seems safe to say that it is open. Network Countermeasures are **2**. So that leaves us with **(3+5) – (1+2) = Severity of 5**.

Correlations:
Steven Drew also described IIS related attacks in his practical paper. http://www.giac.org/practical/Steven_Drew_GCIA.doc. A detailed analysis of Nimda is at http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf.

Defensive Recommendation:
Based on the traces above and the evidence of systems being compromised, defensive recommendations are:

- Review all written security policies with staff and students. If non-existent, implement as soon as possible.
- Fdisk and format all compromised IIS servers currently on the network. Before any new IIS servers are connected, verify patches and/or run through a security checklist. If there is not a security checklist then one needs to be implemented as well and put into place as soon as possible.
- If there is a firewall then review the rules that traffic is compared against

and make adjustments where needed. For example, do not allow the entire network to pass TFTP traffic through it. Only permit devices through the firewall that need TFTP such as routers, etc.

- Monitor the logs daily, both IDS and firewall.

UDP SRC and DST outside of Network

This is an interesting alert. We have 106,883 alerts for systems outside of our network. Here is what a trace file looks like:

```
08/01-00:00:11.738004  [**] UDP SRC and DST outside network
[**] 3.0.0.99:137 -> 10.0.0.1:137
08/01-00:00:13.240134  [**] UDP SRC and DST outside network
[**] 3.0.0.99:137 -> 10.0.0.1:137
08/01-00:00:14.742414  [**] UDP SRC and DST outside network
[**] 3.0.0.99:137 -> 10.0.0.1:137
08/01-00:00:16.244524  [**] UDP SRC and DST outside network
[**] 3.0.0.99:137 -> 10.0.0.1:137
```

As we can see from these alerts, this is NetBIOS traffic. But why do we see this traffic showing up in our alerts file? This is a good question, since the 10.0.0.1 address should never be routed over the Internet. This is possibly someone on the campus network who is using Network Address Translation and has this not configured it correctly. I believe this to be returning traffic, and that is the only reason Snort sees this. Otherwise, these packets could not be routed back to the University.

Severity:

I would rate the severity on these particular alerts as **Low**. This is almost surely a misconfigured system performing NAT.

Defensive Recommendations:

Check border router and properly ACL the device(s) not to allow anything with a RFC 1918 address to be routed in or out of the enterprise. Also as a backup, create rule on Firewall to silently drop and log any packets that hit on the outside interface that are either a source with a RFC 1918 address or destined to a RFC 1918 address. One other thing might be add a rule to the firewall to limit NetBIOS traffic leaving and entering the network.

spp_http_decode: CGI Null Byte attack detected

Here we see the following alerts:

```
ALERT,Aug,01,00:38:06.298032,spp_http_decode: CGI Null Byte
attack detected,66.32.232.141,4065,MY.NET.70.198,80
ALERT,Aug,01,00:38:06.479835,spp_http_decode: CGI Null Byte
attack detected,66.32.232.141,4068,MY.NET.70.198,80
ALERT,Aug,01,00:38:06.839073,spp_http_decode: CGI Null Byte
attack detected,66.32.232.141,4074,MY.NET.70.198,80
ALERT,Aug,01,08:34:38.933318,spp_http_decode: CGI Null Byte
attack detected,MY.NET.87.52,53703,216.32.120.133,80
ALERT,Aug,01,08:34:39.269317,spp_http_decode: CGI Null Byte
```

```
attack detected,MY.NET.87.52,53704,66.135.192.227,80
ALERT,Aug,01,08:34:39.298279,spp_http_decode: CGI Null Byte
attack detected,MY.NET.87.52,53704,66.135.192.227,80
ALERT,Aug,01,08:34:39.563012,spp_http_decode: CGI Null Byte
attack detected,MY.NET.87.52,53704,66.135.192.227,80
ALERT,Aug,01,08:34:39.731511,spp_http_decode: CGI Null Byte
attack detected,MY.NET.87.52,53705,66.135.192.224,80
ALERT,Aug,01,08:34:39.732218,spp_http_decode: CGI Null Byte
attack detected,MY.NET.87.52,53706,66.135.192.224,80
```

These are directed at not only at servers on our network, but also they originate from systems on our network.

This alert is described as an attack that hides system commands behind a null byte. If successful, it allows the hacker to view files on the web server, as well as execute system commands. Some CGI scripts do not look for these types of commands unless told to do so. More information on these attacks is available at http://www.wiretrip.net/rfp/p/doc.asp/i1/d37.htm from Rain Forest Puppy. Even with this information, it is difficult to tell if these are legitimate alerts. Only with a sniff of the traffic would provide the adequate details for these alerts.

Severity:
I would rate these alerts severity as Low, but needing more information. Based on additional information, this rating could go up.

Correlations:
It seems that a lot of folks see these alerts due to the http preprocessor in Snort. So I will start with someone seeing a lot of these and they appear to be false positives: http://www.mcabee.org/lists/snort-users/May-01/msg00691.html. Joe Ellis also saw similar alerts, and noted in his practical - http://www.giac.org/practical/Joe_Ellis_GCIA.doc.

Defensive Recommendations:
While it is entirely possible that the University suffers from this amount of these types of attacks, not to mention the number of internal system generating these alerts, based on the information given here, I am inclined to say that these are for the most part false positives. For a truly defensive recommendation, an internal audit of the University's network with a tool like Nessus to test the CGI capabilities of existing web servers, would a great start. Then the vulnerable systems could be patched accordingly.

SMB Name Wildcard

In the alert logs, we see the following types of alerts:

```
08/01-00:00:16.380309  [**] SMB Name Wildcard [**]
209.206.11.24:137 -> MY.NET.198.204:137
08/01-00:00:31.231827  [**] SMB Name Wildcard [**]
4.46.174.62:3591 -> MY.NET.104.204:137
08/01-00:00:43.595115  [**] SMB Name Wildcard [**]
64.255.103.117:1024 -> MY.NET.86.25:137
```

```
08/01-00:01:04.903044  [**] SMB Name Wildcard [**]
217.17.181.167:137 -> MY.NET.150.11:137
08/01-00:01:04.941808  [**] SMB Name Wildcard [**]
217.17.181.167:137 -> MY.NET.150.133:137
08/01-00:01:07.396678  [**] SMB Name Wildcard [**]
202.133.43.34:137 -> MY.NET.150.133:137
08/01-00:01:19.487860  [**] SMB Name Wildcard [**]
211.38.125.88:137 -> MY.NET.150.11:137
08/01-00:02:03.008969  [**] SMB Name Wildcard [**]
212.49.95.146:137 -> MY.NET.147.90:137
08/01-00:02:04.509783  [**] SMB Name Wildcard [**]
212.49.95.146:137 -> MY.NET.147.90:137
08/01-00:02:07.521311  [**] SMB Name Wildcard [**]
212.49.95.146:137 -> MY.NET.147.90:137
```

These attacks are looking for poorly configured, insecure SMB Services on the University Network. Overall we had 30083 of these alerts. Notice that most of the source and destination ports are 137. There are two exception posted here where the source ports are 1024 and higher.

I also do not believe this particular snip of logs to be actively targeting the University network, or any particular host. Again, I think this to be many hosts looking for unprotected network shares that can easily be compromised. There are multiple attackers from where this traffic originates. I also do not believe this to be some sort of script generating multiple source IP address' for spoofing due to the time stamps on the alert logs.

The breakout of different source IP address' and count for this particular alert is as follows:

```
+-------------------------+------------+
| source_ip               | totalcount |
+-------------------------+------------+
| 216.228.171.81          |       3212 |
| 63.21.4.50              |       2109 |
| 192.168.1.10            |        953 |
| 169.254.113.62          |        339 |
| 198.172.139.117         |        333 |
| 192.168.0.3             |        322 |
| 203.213.57.158          |        281 |
| 209.86.182.157          |        274 |
| 62.219.224.253          |        251 |
| 61.147.24.164           |        216 |
| 80.224.170.115          |        177 |
| 151.196.9.150           |        167 |
| 24.217.169.178          |        165 |
| 12.82.224.217           |        162 |
| 203.213.59.45           |        159 |
| 207.14.236.146          |        150 |
| 80.14.56.67             |        130 |
| 80.224.170.195          |        111 |
| 66.81.122.145           |        110 |
| Multiple address' with less than 110 |
+--------------------------------------+
```

Severity:
I would rate these alerts as **Low**. The attackers never get a response, at least from what we can see in the logs. While unrealistic to assume that a University with many Windows systems are all configured properly, there were no instances of any University host causing one of these alerts.

Correlations:
I will start this off with an archived email from the Snort mailing list describing these types of alerts - http://archives.neohapsis.com/archives/snort/2000-01/0222.html. Also, another email archived from the Incidents mailing list - http://lists.jammed.com/incidents/2001/05/0034.html. This one is from SANS, and also describes the port 137 scans, but also makes mention of a network.vbs worm - http://www.sans.org/newlook/resources/IDFAQ/port_137.htm.

Defensive Recommendations:
Review the firewall rules. If there is no rule to silently drop any tcp / udp 137-139 traffic, it should be added. Also, if you notice in the counts above, there were two instances of source addresses that are defined in RFC 1918, and should not be routed on to the Internet. I would add both a rule to the firewall to drop packets with these source addresses coming into the outside interface, and ingress and egress filters on the border router to silently drop these packets as well. The last thing would be to have quarterly or biannually an independent vulnerability scan / penetration test to audit the security settings in place.

External UDP connection to internal tftp server

Here are the raw logs:

```
08/01-00:07:11.963426  [**] TFTP - External UDP connection to
internal tftp server [**] MY.NET.109.105:69 ->
192.168.0.216:9695
08/01-00:07:11.963569  [**] TFTP - External UDP connection to
internal tftp server [**] MY.NET.111.231:69 ->
192.168.0.216:9695
08/01-00:07:11.963821  [**] TFTP - External UDP connection to
internal tftp server [**] MY.NET.111.230:69 ->
192.168.0.216:9695
08/01-00:07:15.987173  [**] TFTP - External UDP connection to
internal tftp server [**] MY.NET.111.230:69 ->
192.168.0.216:9695
08/01-00:07:15.989785  [**] TFTP - External UDP connection to
internal tftp server [**] MY.NET.111.219:69 ->
192.168.0.216:9695
08/01-00:07:20.015017  [**] TFTP - External UDP connection to
internal tftp server [**] MY.NET.111.231:69 ->
192.168.0.216:9695
```

```
08/01-00:07:20.015036  [**] TFTP - External UDP connection to
internal tftp server [**] MY.NET.111.230:69 ->
192.168.0.216:9695
```

Here is a breakout of the source IP addresses that triggered this alert:

```
+------------------------+------------+
| source_ip              | totalcount |
+------------------------+------------+
| MY.NET.111.230         |       6090 |
| MY.NET.111.231         |       6062 |
| MY.NET.109.105         |       6055 |
| MY.NET.111.219         |       6008 |
| 209.61.187.112         |          2 |
| 199.106.211.166        |          2 |
| 63.250.205.12          |          1 |
+------------------------+------------+
```

As we can see, there were a lot of connections made from the University network. Yet, the alerts state that this is supposed to be and external connection to an internal server. In the logs above, the systems are trying to connect to a 192.168.0.216 address, which is a nonroutable, RFC 1918 address. So why are we seeing the alerts? I believe there are several possibilities. First, these could be misconfigured systems, and possibly some installed software is attempting these connections. But I do not think that this scenario would trigger these alerts, assuming that the $HOME_NET variable in the snort.conf file is equal to MY.NET.

A more possible, and scary, scenario is that these top four systems have been compromised in some way, and the attacker, or *0wn3r* of these systems is probing around the University network for further hosts or networks to try and compromise. The problem here is that we do not have enough information to determine the full extent of these alerts.

The other three:
The host at 209.61.187.112 is up to his own mischief, according to Snort:

```
08/05-08:11:58.013526  [**] High port 65535 udp - possible Red
Worm - traffic [**] 209.61.187.112:65535 -> MY.NET.180.39:62574
08/05-11:54:53.578444  [**] High port 65535 udp - possible Red
Worm - traffic [**] 209.61.187.112:65535 -> MY.NET.180.39:61443
08/05-11:56:07.494894  [**] High port 65535 udp - possible Red
Worm - traffic [**] 209.61.187.112:65535 -> MY.NET.180.39:7325
08/05-12:01:36.959363  [**] High port 65535 udp - possible Red
Worm - traffic [**] 209.61.187.112:30905 -> MY.NET.180.39:65535
08/05-12:50:34.194465  [**] High port 65535 udp - possible Red
Worm - traffic [**] 209.61.187.112:24527 -> MY.NET.180.39:65535
08/05-13:34:24.283923  [**] TFTP - External UDP connection to
internal tftp server [**] 209.61.187.112:40961 ->
MY.NET.180.39:69
08/05-13:51:52.631611  [**] EXPLOIT NTPDX buffer overflow [**]
209.61.187.112:36259 -> MY.NET.180.39:123
```

```
08/05-14:17:21.082227  [**] High port 65535 udp - possible Red
Worm - traffic [**] 209.61.187.112:65535 -> MY.NET.180.39:65535
08/05-15:34:19.404770  [**] TFTP - Internal UDP connection to
external tftp server [**] 209.61.187.112:69 ->
MY.NET.180.39:8791
08/05-15:49:50.437827  [**] High port 65535 udp - possible Red
Worm - traffic [**] 209.61.187.112:11775 -> MY.NET.180.39:65535
```

The host `199.106.211.166` and its mischief:

```
08/05-13:20:55.351713  [**] High port 65535 udp - possible Red
Worm - traffic [**] 199.106.211.166:65535 ->
MY.NET.117.25:65533
08/05-13:20:55.472744  [**] High port 65535 udp - possible Red
Worm - traffic [**] 199.106.211.166:65535 ->
MY.NET.117.25:65533
08/05-13:22:34.494326  [**] High port 65535 udp - possible Red
Worm - traffic [**] 199.106.211.166:61918 ->
MY.NET.117.25:65535
08/05-13:24:29.557995  [**] Attempted Sun RPC high port access
[**] 199.106.211.166:37927 -> MY.NET.117.25:32771
08/05-13:34:24.311429  [**] TFTP - External UDP connection to
internal tftp server [**] 199.106.211.166:40961 ->
MY.NET.117.25:69
08/05-13:34:24.575801  [**] TFTP - External UDP connection to
internal tftp server [**] 199.106.211.166:40961 ->
MY.NET.117.25:69
```

The host `63.250.205.12` in the logs:

```
08/05-13:34:24.979258  [**] TFTP - External UDP connection to
internal tftp server [**] 63.250.205.12:40961 ->
MY.NET.114.44:69
```

As we can see here, the `209.61.187.112` host is attacking the MY.NET.180.39 host and the `199.106.211.166` host is attacking MY.NET.117.25. The `209.61.187.112` attacker is trying several buffer overflows on different services, like tftp and ntp (network time protocol). The `199.106.211.166` even tries a high port RPC connection. Both are trying connections to the 65535 port, which could be the Adore worm, or some other custom script developed by the attacker(s), or retrieved from the Internet somewhere. Adore is a worm that was originally called the Red Worm. It is similar to the Ramen and Lion worms. Adore scans the Internet checking Linux hosts to determine whether they are vulnerable to any of the following well-known exploits: LPRng, rpc-statd, wu-ftpd and BIND. LPRng is installed by default on Red Hat 7.0 systems – more about the Adore worm is here - http://www.sans.org/y2k/adore.htm. This is someone who saw similar scanning on their network - http://archives.neohapsis.com/archives/snort/2002-06/0103.html.

From what we can tell and hope, these attacks were not successful. We saw no replies to any of this traffic. They could be the worm trying to propagate, or some sort of recon. More information is needed.

Severity:
While this does appear to be very suspicious traffic, we cannot be sure of compromise just yet. Based on the information we do have in the above paragraphs, I would rate as **Medium**, or needing to be checked out fully and having the potential of being dangerous.

Correlations:
Aside from the two listings above, Bradley Urwiller saw similar Red Worm traffic and documented in his practical - http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf.

Defensive Recommendations:
I would say first that the network or security admin need to visit the four hosts listed that are on the University network and determine the nature of these alerts. Either by a well placed sniffer or some sort of auditing software such as Nessus or nmap would help aid in this process. Also, if actually going to the system in question is possible, this could also help to confirm or deny any malicious software on the system. The last thing is to verify that this traffic is dropping on the firewall due to the clean up rule that should be configured to drop everything that has not matched previous "allow" rules.

External RPC call

There were 14578 External RPC call alerts. The raw logs look like this:

```
08/01-00:37:19.516963  [**] External RPC call [**]
66.32.232.141:3949 -> MY.NET.70.198:111
08/01-00:37:19.537984  [**] External RPC call [**]
66.32.232.141:3949 -> MY.NET.70.198:111
08/01-00:37:53.620882  [**] External RPC call [**]
66.32.232.141:4025 -> MY.NET.70.198:111
08/01-00:37:53.644680  [**] External RPC call [**]
66.32.232.141:4025 -> MY.NET.70.198:111
08/01-00:37:53.647787  [**] External RPC call [**]
66.32.232.141:4025 -> MY.NET.70.198:111
08/01-00:38:58.305901  [**] External RPC call [**]
66.32.232.141:4098 -> MY.NET.70.198:111
08/01-00:38:58.324451  [**] External RPC call [**]
66.32.232.141:4098 -> MY.NET.70.198:111
08/01-00:38:58.354774  [**] External RPC call [**]
66.32.232.141:4098 -> MY.NET.70.198:111
08/01-00:38:59.625945  [**] External RPC call [**]
66.32.232.141:4098 -> MY.NET.70.198:111
08/01-00:39:04.324736  [**] External RPC call [**]
66.32.232.141:4098 -> MY.NET.70.198:111
08/01-00:39:06.017998  [**] External RPC call [**]
66.32.232.141:4098 -> MY.NET.70.198:111
08/01-07:45:10.982730  [**] External RPC call [**]
203.239.155.2:45601 -> MY.NET.80.40:111
08/01-07:45:10.982858  [**] External RPC call [**]
203.239.155.2:45602 -> MY.NET.80.41:111
08/01-07:45:10.983121  [**] External RPC call [**]
```

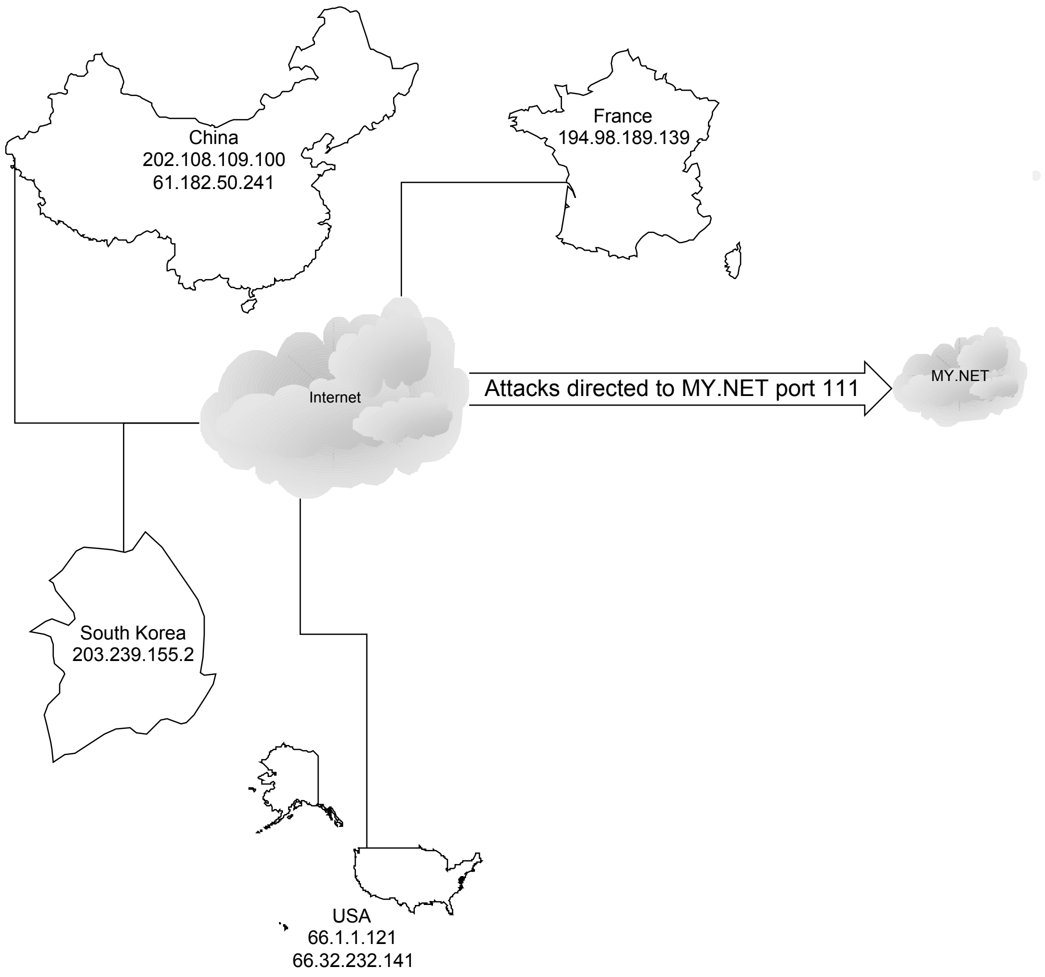
```
203.239.155.2:45603 -> MY.NET.80.42:111
08/01-07:45:10.983133  [**] External RPC call [**]
203.239.155.2:45604 -> MY.NET.80.43:111
08/01-07:45:10.983142  [**] External RPC call [**]
203.239.155.2:45605 -> MY.NET.80.44:111
08/01-07:45:10.983996  [**] External RPC call [**]
203.239.155.2:45608 -> MY.NET.80.47:111
08/01-07:45:10.984039  [**] External RPC call [**]
203.239.155.2:45606 -> MY.NET.80.45:111
08/01-07:45:10.984205  [**] External RPC call [**]
203.239.155.2:45609 -> MY.NET.80.48:111
08/01-07:45:10.984219  [**] External RPC call [**]
203.239.155.2:45607 -> MY.NET.80.46:111
```

The top source addresses for this particular alert:

```
+-------------------------+------------+
| source_ip               | totalcount |
+-------------------------+------------+
| 194.98.189.139          |       8352 |
| 61.182.50.241           |       4519 |
| 203.239.155.2           |        918 |
| 202.108.109.100         |        775 |
| 66.32.232.141           |         11 |
| 66.1.1.121              |          3 |
+-------------------------+------------+
```

These alerts were triggered because the list of attacking hosts was trying to connect to hosts here on the University network on the SunRPC port. (tcp 111) This is the portmapper port that allows applications that run at very high port numbers to be mapped through this service. One of the biggest known exploits for this port was the sadmind worm. This worm would attack and compromise Solaris systems, and then the compromised Solaris systems would attack Microsoft IIS servers using a Unicode attack. Read more about sadmind worm here: http://www.cert.org/advisories/CA-2001-11.html.

Data link Graph:
The following link graph shows the attacking host IP addresses and the countries from where the attacks originate, according to http://www.apnic.net, http://www.ripe.net/ripencc/pub-services/db/whois/whois.html, and http://www.arin.net/whois.



Severity:
This series of alerts looks like either systems already compromised by this sadmind worm and trying to propagate, or individuals looking for systems listening on tcp 111 to try other exploits. There were no internal systems that were the source of any of these alerts, so I would rate the severity as **Low**.

Correlations:
I will start with a detect from SANS, though it is very small - http://www.sans.org/y2k/010600-0900.htm. Next is the actual exploit code from Securiteam - http://www.securiteam.com/exploits/3P5Q1Q0QAO.html. Finally, a note about the sadmind worm and vulnerabilities from SANS in their top 20 vulnerabilities, from May 2002 - http://www.sans.org/top20/top20_Oct01.htm.

Defensive Recommendations:
Review firewall rules. This should actually be stopped by the cleanup rule in the rule base, provided there are no specific applications that need this. This rule is the rule that basically states any -> any -> drop ->log. Perform internal vulnerability scan of network to determine if any systems are listening for this service. If these services are found and needed, make sure system has latest

security patches.

Possible Trojan server activity:

Next, we will look at the "Possible Trojan server activity" alerts. Here is the trace from August 1.

```
08/01-00:01:17.167418  [**] Possible trojan server activity [**]
24.61.17.248:27374 -> MY.NET.11.4:80
08/01-00:01:29.167424  [**] Possible trojan server activity [**]
24.61.17.248:27374 -> MY.NET.11.4:80
08/01-00:37:21.499248  [**] Possible trojan server activity [**]
66.32.232.141:4002 -> MY.NET.70.198:27374
08/01-00:37:21.499270  [**] Possible trojan server activity [**]
MY.NET.70.198:27374 -> 66.32.232.141:4002
08/01-00:37:22.509184  [**] Possible trojan server activity [**]
66.32.232.141:4002 -> MY.NET.70.198:27374
08/01-00:37:22.509327  [**] Possible trojan server activity [**]
MY.NET.70.198:27374 -> 66.32.232.141:4002
08/01-01:41:36.150124  [**] Possible trojan server activity [**]
204.181.76.134:2553 -> MY.NET.2.209:27374
08/01-01:41:56.161496  [**] Possible trojan server activity [**]
204.181.76.134:4354 -> MY.NET.3.2:27374
08/01-01:41:56.162152  [**] Possible trojan server activity [**]
MY.NET.3.2:27374 -> 204.181.76.134:4354
08/01-02:23:11.627984  [**] Possible trojan server activity [**]
206.246.167.129:4522 -> MY.NET.178.199:27374
08/01-02:23:11.628581  [**] Possible trojan server activity [**]
MY.NET.178.199:27374 -> 206.246.167.129:4522
08/01-02:47:18.379967  [**] Possible trojan server activity [**]
63.53.111.92:1280 -> MY.NET.70.236:27374
08/01-05:37:37.775293  [**] Possible trojan server activity [**]
66.250.73.88:4020 -> MY.NET.178.199:27374
08/01-05:37:37.775967  [**] Possible trojan server activity [**]
MY.NET.178.199:27374 -> 66.250.73.88:4020
08/01-09:30:17.380848  [**] Possible trojan server activity [**]
206.246.167.129:2237 -> MY.NET.178.199:27374
08/01-09:30:17.381136  [**] Possible trojan server activity [**]
MY.NET.178.199:27374 -> 206.246.167.129:2237
08/01-10:53:07.394617  [**] Possible trojan server activity [**]
200.204.151.83:4506 -> MY.NET.197.187:27374
08/01-10:53:13.407241  [**] Possible trojan server activity [**]
200.204.151.83:4506 -> MY.NET.197.187:27374
08/01-20:47:54.757679  [**] Possible trojan server activity [**]
138.88.40.155:1919 -> MY.NET.167.2:27374
08/01-20:47:54.758668  [**] Possible trojan server activity [**]
MY.NET.167.2:27374 -> 138.88.40.155:1919
08/01-20:47:55.144487  [**] Possible trojan server activity [**]
138.88.40.155:1949 -> MY.NET.167.3:27374
08/01-20:47:55.145337  [**] Possible trojan server activity [**]
MY.NET.167.3:27374 -> 138.88.40.155:1949
08/01-20:47:56.127398  [**] Possible trojan server activity [**]
138.88.40.155:1949 -> MY.NET.167.3:27374
08/01-20:47:56.128236  [**] Possible trojan server activity [**]
MY.NET.167.3:27374 -> 138.88.40.155:1949
08/01-20:47:58.499240  [**] Possible trojan server activity [**]
138.88.40.155:2204 -> MY.NET.167.11:27374
```

The first two lines show what we would like to see when these alerts are tripped. There is no response from the targeted machine. The second best thing we could hope to see is described in the next few lines of the trace. For example, the host 66.32.232.141 attempts a connection to MY.NET.70.198 on tcp port 27374 or the SubSeven Trojan port. We then see in the trace that MY.NET.70.198 responds to 66.32.232.141, to the same port that 66.32.232.141 used for a source port. I believe these are RST packets telling the attacker that this system is not listening on that port. The rest of the packets in the trace look like this including the short scan from 138.88.40.155.

Strange Traffic:

```
08/03-09:42:12.333178  [**] Possible trojan server activity
[**] 80.220.255.51:27374 -> MY.NET.70.113:80
08/03-09:42:12.333265  [**] Possible trojan server activity
[**] MY.NET.70.113:80 -> 80.220.255.51:27374
08/03-09:42:12.453550  [**] Possible trojan server activity
[**] 80.220.255.51:27374 -> MY.NET.70.113:80
08/03-09:42:12.453560  [**] Possible trojan server activity
[**] 80.220.255.51:27374 -> MY.NET.70.113:80
08/03-09:42:12.575381  [**] Possible trojan server activity
[**] 80.220.255.51:27374 -> MY.NET.70.113:80
08/03-09:42:12.575524  [**] Possible trojan server activity
[**] MY.NET.70.113:80 -> 80.220.255.51:27374
```

Here we see the host at 80.220.255.51 attempting a connection to MY.NET.70.113 over port 80. My first thought is that this is some sort of proxy server or maybe a firewall doing Port Address Translation and this is the ephemeral port that was chosen. Further investigations lead me to believe that the host at 80.220.255.51 is infected with the Ramen worm and trying to propagate. I believe that the two responses are RST packets telling the attacking host that this system is not listening for that service. Could this be spoofed? Definitely possible but I think unlikely because all the end victim would see would be RST packets. Could this be someone using a scanning tool and setting the source ports as 27374? It appears that the attacker was using some automated tool and probably scanning other hosts as well. I think this is probably the better explanation based on the number of packets in the trace and the packet payload not being available. It is a question with no definite answer. If we had the packet payloads I believe we could answer this question and move on.

Severity:
Based on the formula **(Criticality + Lethality) – (System + Net Countermeasures) = Severity** and what we think is the way the University has their network laid out we would rate this particular compromise as **Minor (Noisy).** For Criticality, we again assess that this is a web server, but we do not know if it is the University's main web server or a rogue one put up by a student. Therefore Criticality is **3**. For Lethality we know that the system has not been compromised and is currently not listening for SubSeven connections. Lethality is **3**. For System countermeasures we do not know the status of this box but we do know it is not compromised and not listening on 27374. System

countermeasures are **3**. For Net countermeasures we would like to assume that there is a firewall in place controlling traffic leaving and entering the network. Nonetheless, we do not know because we have no physical access to this network. If there is a firewall in place it seems safe to say that it is, to a certain extent, open. Network Countermeasures are **2**. So that leaves us with **(3+3) – (3+2) = Severity of 1**.

Correlations:
William Stearns has an excellent description of the Ramen worm at http://www.sans.org/y2k/ramen.htm. Michael Holstein also saw similar Ramen Worm traffic. He notes this in his practical assignment. http://www.giac.org/practical/Michael_Holstein_GCIA.doc

Defensive recommendation:

- Review Firewall rules. If the following rules do not already exist, then create the following:
    - Setup a rule that would silently drop any traffic with destination port of 27374 and log the results.
    - Also add rule to drop any traffic with a source IP of the internal network and a source port of 27374 to prevent any internal systems from using that port and creating any noise.
    - Evaluate the logs over the next 60 days to see if they need editing or removal.
- Review Snort IDS rules. If the following rules do not exist, then create the following:
    - Add Snort rule to alert on a response to a SYN/ACK packet from the internal network to an external host with a source port of 27374. This with the firewall logs should help determine who needs immediate attention.
    - Evaluate the logs over the next 60 days to see if any alerts are triggered. If none, review rule to verify if correct. Also correlate with firewall logs.

Notable Scans:

The scan output logs revealed these stats:

Alert Stats for table -> scans	
Total number of scans:	**4110198**
Breakout by type:	

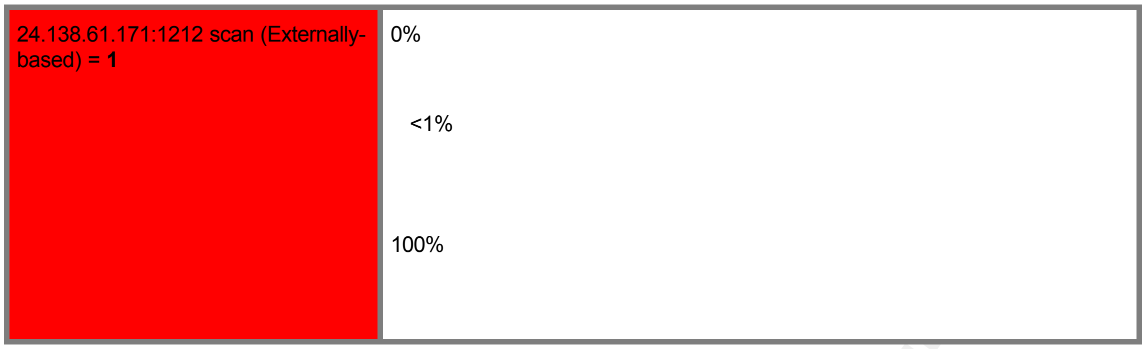
UDP scan (Externally-based) = **3112455**	0% 75% 100%
SYN scan (Externally-based) = **997435**	0% 24% 100%
VECNA scan (Externally-based) = **71**	0% <1% 100%
INVALIDACK scan (Externally-based) = **60**	0% <1% 100%
NULL scan (Externally-based) = **59**	0% <1% 100%

NOACK scan (Externally-based) = **52**	0% <1% 100%
UNKNOWN scan (Externally-based) = **18**	0% <1% 100%
scan (Externally-based) = **13**	0% <1% 100%
SYNFIN scan (Externally-based) = **5**	0% <1% 100%
XMAS scan (Externally-based) = **4**	0% <1% 100%

FULLXMAS scan (Externally-based) = **4**	0% <1% 100%
FIN scan (Externally-based) = **4**	0% <1% 100%
MY.NET.70.200:4946 scan (Externally-based) = **4**	0% <1% 100%
******S* scan (Externally-based) = **2**	0% <1% 100%
2 scan (Externally-based) = **1**	0% <1% 100%

-> scan (Externally-based) = **1**	0% <1% 100%
NMAPID scan (Externally-based) = **1**	0% <1% 100%
24.169.17.202:41170 scan (Externally-based) = **1**	0% <1% 100%
MY.NET.84.234:4949 scan (Externally-based) = **1**	0% <1% 100%
MY.NET.84.234:1485 scan (Externally-based) = **1**	0% <1% 100%

MY.NET.84.234:2272 scan (Externally-based) = **1**	0% <1% 100%
MY.NET.84.234:3993 scan (Externally-based) = **1**	0% <1% 100%
MY.NET.84.234:3655 scan (Externally-based) = **1**	0% <1% 100%
18:56:08 scan (Externally-based) = **1**	0% <1% 100%
MY.NET.84.234:4124 scan (Externally-based) = **1**	0% <1% 100%



As you can see there are some 4.1 million scan alerts. Some of the entries on the above graph I attribute to anomalies in the log files themselves and will be ignored. Others will be analyzed and defensive recommendations made.

Again, as with the alerts files, the sheer volume of alerts by a small number of different alerts causes concern. If these are taken care of and reconfigured correctly, then the real alerts for concern should rise to the top and help prioritize the analysis from here on out.

DNS Noise:

```
Aug  1 00:02:00 MY.NET.137.7:1603 -> 198.6.100.37:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1093 -> 64.27.65.13:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1093 -> 64.27.64.76:53 UDP
Aug  1 00:02:00 MY.NET.137.7:1614 -> 192.26.92.30:53 UDP
Aug  1 00:02:00 MY.NET.137.7:1618 -> 192.31.80.30:53 UDP
Aug  1 00:02:00 MY.NET.137.7:1610 -> 192.52.178.30:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1093 -> 192.5.6.30:53 UDP
Aug  1 00:02:00 MY.NET.137.7:1617 -> 192.41.162.30:53 UDP
Aug  1 00:02:00 MY.NET.137.7:53 -> 204.127.129.1:53 UDP
Aug  1 00:02:00 MY.NET.137.7:1615 -> 194.164.2.10:53 UDP
Aug  1 00:02:00 MY.NET.137.7:1616 -> 192.33.14.30:53 UDP
Aug  1 00:02:00 MY.NET.137.7:1620 -> 192.35.51.30:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1634 -> 192.12.94.30:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1093 -> 152.163.159.232:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1093 -> 192.26.210.1:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1093 -> 205.188.157.232:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1638 -> 192.43.172.30:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1639 -> 12.106.137.73:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1641 -> 192.42.93.30:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1650 -> 210.132.100.101:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1654 -> 192.55.83.30:53 UDP
Aug  1 00:02:01 MY.NET.137.7:1093 -> 198.6.100.21:53 UDP
Aug  1 00:02:02 MY.NET.137.7:1093 -> 194.164.2.11:53 UDP
Aug  1 00:02:02 MY.NET.137.7:1093 -> 194.62.44.11:53 UDP
Aug  1 00:02:02 MY.NET.137.7:1093 -> 194.62.44.10:53 UDP
```

The snip of Snort logs above is from the first couple of minutes of August 1st. All of these appear to be DNS queries to outside servers. The one notable exception is the line:

```
      Aug  1 00:02:00 MY.NET.137.7:53 -> 204.127.129.1:53 UDP
```

This line appears to be indicative of one of the hosts running BIND 4.x.x, which commonly used both udp 53 for source and destination.

Severity:
This is clearly noise caused by the volume of connections made by the DNS server over a given period of time.

Defensive Recommendation:
Edit the snort.conf file and add the DNS server IP address to the "preprocessor portscan-ignorehosts:" line or by editing the $DNS_SERVERS variable and checking to make sure the variable is listed as such, "preprocessor portscan-ignorehosts: $DNS_SERVERS". This would alleviate 49208 alerts for the five days worth of logs analyzed here.

Blubster

The next set of alarms is also based on volume. This one host takes the prize. It had 2,439,514 alerts for its IP address alone. Most of these alerts show this host connecting to hosts outside the network on UDP 41170. A Google search quickly turns up "Blubster Music Community". Blubster is a peer-to-peer music swapping service much like the old Napster. The trace looks like this:

```
      Aug  2 09:32:46 MY.NET.70.200:4946 -> 202.156.2.138:41170 UDP
      Aug  2 09:32:46 MY.NET.70.200:4946 -> 213.224.83.46:41170 UDP
      Aug  2 09:32:46 MY.NET.70.200:4946 -> 64.129.98.53:41170 UDP
      Aug  2 09:32:46 MY.NET.70.200:4946 -> 151.213.243.250:41170 UDP
      Aug  2 09:32:46 MY.NET.70.200:4946 -> 80.128.185.49:41170 UDP
      Aug  2 09:32:46 MY.NET.70.200:4946 -> 198.142.59.168:41170 UDP
      <snip>
      Aug  4 13:02:56 MY.NET.70.200:4946 -> 68.103.79.238:41170 UDP
      Aug  4 13:02:56 MY.NET.70.200:4946 -> 24.157.136.59:41170 UDP
      Aug  4 13:02:56 MY.NET.70.200:4946 -> 213.99.206.132:41170 UDP
      Aug  4 13:02:56 MY.NET.70.200:4946 -> 199.17.28.20:41170 UDP
      <snip>
      Aug  5 03:00:49 MY.NET.70.200:4946 -> 156.40.69.53:41170 UDP
      Aug  5 03:00:49 MY.NET.70.200:4946 -> 66.133.141.176:41170 UDP
      Aug  5 03:00:49 MY.NET.70.200:4946 -> 24.72.10.240:41170 UDP
      Aug  5 03:00:49 MY.NET.70.200:4946 -> 24.232.189.18:41170 UDP
      Aug  5 03:00:49 MY.NET.70.200:4946 -> 217.1.73.210:41170 UDP
```

Severity:
While this is not a hack it does present its own set of security and moral dilemmas.

- While there may be no vulnerabilities for Blubster yet one could bet there will be soon. That could be potentially dangerous for the University's network. There is no telling what kinds of virii this client could be potentially downloading only to begin infecting the internal network.
- Bandwidth issues. It appears by the logs that this client/server is online

almost 24x7 and that is a lot of traffic.
- Legal issues. This is a service much like the now defunct Napster. Napster was found to be guilty “for willful contributory and vicarious copyright infringement.” http://news.findlaw.com/hdocs/docs/napster/napster022102ord.pdf and also http://news.findlaw.com/legalnews/lit/napster/. The University could potentially also be held liable for “allowing” this traffic on their network.

Defensive Recommendations:
- Review Network Policy regarding allowed network traffic. Revise as needed.
- Review firewall rules. If a rule is not in place to stop this sort of traffic, add one immediately. The rule would read something like:
  `Any_source -> Any_destination -> UDP 41170 -> drop -> log`

NetBIOS Scan:
Here we see the host at 216.228.171.81 scanning for any open NetBIOS ports on the network. It appears as though the attacker scans the entire net block with a mix of both TCP and UDP packets.

```
Aug  1 06:00:05 216.228.171.81:3089 -> MY.NET.10.220:139 SYN ******S*
Aug  1 06:00:05 216.228.171.81:3107 -> MY.NET.10.229:139 SYN ******S*
Aug  1 06:00:05 216.228.171.81:3106 -> MY.NET.10.229:445 SYN ******S*
Aug  1 06:00:05 216.228.171.81:3091 -> MY.NET.10.221:139 SYN ******S*
Aug  1 06:00:05 216.228.171.81:3090 -> MY.NET.10.221:445 SYN ******S*
Aug  1 06:00:05 216.228.171.81:3114 -> MY.NET.10.233:445 SYN ******S*
Aug  1 06:00:05 216.228.171.81:3115 -> MY.NET.10.233:139 SYN ******S*
Aug  1 06:00:06 216.228.171.81:3093 -> MY.NET.10.222:139 SYN ******S*
Aug  1 06:00:06 216.228.171.81:3092 -> MY.NET.10.222:445 SYN ******S*
Aug  1 06:00:06 216.228.171.81:3109 -> MY.NET.10.230:139 SYN ******S*
Aug  1 06:00:06 216.228.171.81:3108 -> MY.NET.10.230:445 SYN ******S*
Aug  1 06:00:06 216.228.171.81:3116 -> MY.NET.10.234:445 SYN ******S*
Aug  1 06:00:06 216.228.171.81:3117 -> MY.NET.10.234:139 SYN ******S*
<snip>
Aug  1 07:07:58 216.228.171.81:137 -> MY.NET.21.11:137 UDP
Aug  1 07:07:57 216.228.171.81:4700 -> MY.NET.21.12:139 SYN ******S*
Aug  1 07:07:57 216.228.171.81:4699 -> MY.NET.21.12:445 SYN ******S*
Aug  1 07:07:57 216.228.171.81:137 -> MY.NET.21.7:137 UDP
Aug  1 07:07:58 216.228.171.81:137 -> MY.NET.21.9:137 UDP
Aug  1 07:07:58 216.228.171.81:4702 -> MY.NET.21.13:445 SYN ******S*
Aug  1 07:07:58 216.228.171.81:4703 -> MY.NET.21.13:139 SYN ******S*
Aug  1 07:07:58 216.228.171.81:137 -> MY.NET.21.13:137 UDP
Aug  1 07:07:58 216.228.171.81:137 -> MY.NET.21.10:137 UDP
Aug  1 07:07:58 216.228.171.81:137 -> MY.NET.21.8:137 UDP
```

What is the attacker looking for? Most likely the attacker is looking for open Microsoft shares that are exposed to the Internet with weak or no passwords. Once found they can exploit this and load virtually any software the attacker wants and the victim PC becomes a zombie.

Who is the attacker? By running the IP address through http://www.arin.net/whois we can see which ISP and perhaps which person or corporation is responsible for this attack.

216.118.171.81

```
Search results for: 216.118.171.81

OrgName:    Sonora Quest Laboratories
OrgID:      SQL-2

NetRange:   216.118.160.0 - 216.118.175.255
CIDR:       216.118.160.0/20
NetName:    SQLNET
NetHandle:  NET-216-118-160-0-1
Parent:     NET-216-0-0-0-0
NetType:    Direct Assignment
Comment:
RegDate:    2000-09-15
Updated:    2000-09-15

TechHandle: JP2231-ARIN
TechName:   Pflugfelder, James
TechPhone:  +1-602-495-4331
TechEmail:  james.pflugfelder@bannerhealth.com

# ARIN Whois database, last updated 2002-09-17 19:05
# Enter ? for additional hints on searching ARIN's Whois database.
```

Now we have the evidence of the attack and we know who did it. This is where the University's written security policy will lead the way.

Defensive recommendation:

- If policy states that a letter or a phone call to the individual, company, corporation or the ISP of this host is needed then contact should be made. This could be just a misconfigured system and the system admin doesn't know about it. Worse still, it could be a compromised system and by letting the system admin know about it helps him to begin an investigation as to how it happened and how to fix the problem.
- Under no circumstances should the university retaliate and launch an attack.

Port 80 Scan:

```
Aug  4 17:30:46 24.138.61.171:2020 -> MY.NET.10.5:80 SYN ******S*
Aug  4 17:30:46 24.138.61.171:2021 -> MY.NET.10.6:80 SYN ******S*
Aug  4 17:30:46 24.138.61.171:2022 -> MY.NET.10.7:80 SYN ******S*
Aug  4 17:30:47 24.138.61.171:2031 -> MY.NET.10.16:80 SYN ******S*
Aug  4 17:30:47 24.138.61.171:2032 -> MY.NET.10.17:80 SYN ******S*
Aug  4 17:30:48 24.138.61.171:2033 -> MY.NET.10.18:80 SYN ******S*
Aug  4 17:30:47 24.138.61.171:2034 -> MY.NET.10.19:80 SYN ******S*
Aug  4 17:30:48 24.138.61.171:2051 -> MY.NET.10.1:80 SYN ******S*
Aug  4 17:30:47 24.138.61.171:2038 -> MY.NET.10.23:80 SYN ******S*
Aug  4 17:30:47 24.138.61.171:2039 -> MY.NET.10.24:80 SYN ******S*
Aug  4 17:30:48 24.138.61.171:2040 -> MY.NET.10.25:80 SYN ******S*
```

This ports scan was interesting because even with all of the Nimda that is documented in the beginning of this report here is a TCP port 80 scan looking for vulnerable web servers. Is this attacker necessarily looking for IIS servers? I think not. This attacker is probably looking for vulnerable Apache web servers to exploit. However, without the packet payload, it is just a guess.

```
Aug  4 17:30:49 24.138.61.171:2070 -> MY.NET.10.22:80 SYN ******S*
Aug  4 17:30:49 24.138.61.171:2103 -> MY.NET.10.77:80 SYN ******S*
Aug  4 17:30:49 24.138.61.171:2106 -> MY.NET.10.79:80 SYN ******S*
Aug  4 17:30:49 24.138.61.171:2107 -> MY.NET.10.80:80 SYN ******S*
Aug  4 17:30:49 24.138.61.171:2017 -> MY.NET.10.2:80 SYN ******S*
Aug  4 17:30:49 24.138.61.171:2018 -> MY.NET.10.3:80 SYN ******S*
Aug  4 17:30:49 24.138.61.171:2019 -> MY.NET.10.4:80 SYN ******S*
Aug  4 17:30:49 24.138.61.171:2021 -> MY.NET.10.6:80 SYN ******S*
```

According to these logs the attacker sees no responses from any host that is probed and that is a good thing. Although a tcpdump file of this traffic may prove us wrong, that is something we do not have. Who is our Script Kiddie?

24.138.61.171

```
OrgName:    Access Cable Television
OrgID:      ACCA

NetRange:   24.138.0.0 - 24.138.79.255
CIDR:       24.138.0.0/18, 24.138.64.0/20
NetName:    ACCESS-BLK1
NetHandle:  NET-24-138-0-0-1
Parent:     NET-24-0-0-0-0
NetType:    Direct Allocation
NameServer: EUROPA.ACCESSCABLE.NET
NameServer: PEGGY.ACCESSCABLE.NET
Comment:
RegDate:    1997-09-05
Updated:    2002-07-24

TechHandle: JP1495-ARIN
TechName:   Potvin, Jeff
TechPhone:  +1-902-469-9540
TechEmail:  jpotvin@accesscable.com
# ARIN Whois database, last updated 2002-09-17 19:05
# Enter ? for additional hints on searching ARIN's Whois database.
```

Defensive Recommendation:

- Verify any Snort rules for Apache. Continue to monitor for further alarms.
- Set the port scan threshold higher to help alleviate some of the extra alerts.

OOS Packets:

These were not run through any scripts or imported into any database. Instead, they were done the old fashioned way, by manually searching through and typing the results. Therefore, I will only report on the ones I assume need attention, not every one. The following format was used for these packets:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/31-03:01:26.781451 4.64.202.110:22690 -> MY.NET.88.162:1607
TCP TTL:114 TOS:0x0 ID:64352  DF
**SFR*A* Seq: 0x4BEED5A   Ack: 0x57781947   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL
```

This packet qualifies for Out of Spec because of the set tcp flags. They are SYN, FIN, RST, and ACK. This definitely qualifies as an illegal packet. The destination port of 1607 is for “stt”, as defined by Neohapsis.

Pop Mail Traffic (652 alerts):

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/01-00:04:05.437159 68.32.126.64:26053 -> MY.NET.6.7:110
TCP TTL:48 TOS:0x0 ID:54125  DF
21S***** Seq: 0x6FF77DB   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 51881142 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/01-00:05:09.035277 68.32.126.64:26054 -> MY.NET.6.7:110
TCP TTL:48 TOS:0x0 ID:10477  DF
21S***** Seq: 0xA939563   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 51887502 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/01-00:06:12.541572 68.32.126.64:26056 -> MY.NET.6.7:110
TCP TTL:48 TOS:0x0 ID:48693  DF
21S***** Seq: 0xEC8686E   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 51893851 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/01-00:07:16.233747 68.32.126.64:26057 -> MY.NET.6.7:110
TCP TTL:48 TOS:0x0 ID:25800  DF
21S***** Seq: 0x12508BF5   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 51900221 0 EOL EOL EOL EOL
```

These packets come to us from host 68.32.126.64 that appears to be trying to hit a pop3 mail server located at MY.NET.6.7. Notice the tcp flags. It appears the two reserved bits are turned on, as well as the SYN flag or the host is scanning for a specific pop3 mail server that is sending malformed packets to try to evade IDS systems. I conclude the latter, due to time stamps and source ports. First, the time stamps, there is approximately 60 plus seconds between connection attempts. Second, the source ports also increase which is indicative of a new connection attempt, not a retry of a connection.

Who is 68.32.126.64?

This is the output from http://www.arin.net/whois.

Search results for: 68.32.126.64

```
Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)
                                  68.32.0.0 - 68.63.255.255
Comcast Cable Communications, Inc. JUMPSTART-BALTIMOR-A3 (NET-68-32-
112-0-1)
                                  68.32.112.0 - 68.32.143.255

# ARIN Whois database, last updated 2002-09-17 19:05
# Enter ? for additional hints on searching ARIN's Whois database.
```

Clicking on the top label for Comcast gives the information for the entire Comcast subnet that they own. Clicking on the second entry shows this specific network has been reassigned.

```
CustName:    Comcast Cable Communications, Inc.
Address:     3 Executive Campus Cherry Hill, NJ 08002
Country:     US
Comment:
RegDate:     2002-06-15
Updated:     2002-06-15

NetRange:    68.32.112.0 - 68.32.143.255
CIDR:        68.32.112.0/20, 68.32.128.0/20
NetName:     JUMPSTART-BALTIMOR-A3
NetHandle:   NET-68-32-112-0-1
Parent:      NET-68-32-0-0-1
NetType:     Reassigned
Comment:
RegDate:     2002-06-15
Updated:     2002-06-15

# ARIN Whois database, last updated 2002-09-17 19:05
# Enter ? for additional hints on searching ARIN's Whois database.
```

Severity: (Medium)
This appears to be active targeting. The attacker only has one system in mind.

Defensive Recommendation:

- Send an email or call Comcast and report to their abuse department.
- If MY.NET.6.7 is indeed a pop3 server, review current security posture both manually, going through system and verifying patches. Also, do an internal port scan to determine any weaknesses this system may have and correct.

Top Talkers:

The following tables list the top ten talkers for the alerts and the scans.

IP Address	# of alerts
MY.NET.100.208	**1433783**
MY.NET.84.234	**481329**

3.0.0.99	**51359**
63.250.213.12	**32117**
MY.NET.81.37	**27085**
194.98.189.139	**8375**
MY.NET.85.74	**6990**
80.137.90.34	**6899**
MY.NET.111.230	**6090**
MY.NET.111.231	**6059**

IP Address	# of alerts
MY.NET.70.200	**2439514**
MY.NET.84.234	**478411**
MY.NET.100.208	**170345**
MY.NET.70.207	**137226**
MY.NET.82.2	**127792**
MY.NET.165.24	**104553**
MY.NET.83.150	**90049**
MY.NET.137.7	**49208**
MY.NET.70.133	**42744**
MY.NET.81.27	**31926**

It reasons that even though the University has implemented Snort as a Network Intrusion Detection system it is very inefficient. It needs to be reconfigured and rules tweaked so it can better report on the true intrusions of the University's network.

Tools used for these analyses:

- Microsoft Word
- Microsoft Internet Explorer 6
- Microsoft Visio Professional 2002
- Php 4.2.2 (http://www.php.net)
- MySQL 3.23.36 (http://www.mysql.com)
- Removing the duplicate scan entries from the alerts file was accomplished by using a perl script written and used in a previous practical by Tod Beardsley. This script also converted the log file into a comma-separated file.
- I took the comma separated text file and imported into a MySQL database for further analysis using these commands.

```
a. #mysql -p password
b. mysql> use database paper;
c. LOAD DATA INFILE 'snort-csvfile.txt' INTO TABLE
   alerts FIELDS TERMINATED BY ',' LINES TERMINATED BY
   '\n';
```

- Mike Aylor, GCIA Hopeful, for help with the mysql commands.
- Lastly, with the help of Richard Hutchins and the php-db mailing list, php scripts were written and customized to print an html table of the results. Here is the core of the scripts.

```
$a = mysql_query("select distinct(alert_name), count(*)
as totalcount from alerts group by alert_name order by
totalcount desc");
while ($row = mysql_fetch_array($a)) {

$types = $row["alert_name"];
$total = $row["totalcount"];

<!—my html output here →
}
```

References:

1. SnortSnarf – http://www.silicondefense.com/software/snortsnarf/
2. Demarc – http://www.demarc.com/
3. McAfee: Description of Nimda; http://vil.mcafee.com/dispVirus.asp?virus_k=99209
4. Steven Drew: GCIA Practical Assignment 3.1 http://www.giac.org/practical/Steven_Drew_GCIA.doc
5. Nimda Worm Analysis - http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf
6. CGI Null Byte description – by Rain Forest Puppy - http://www.wiretrip.net/rfp/p/doc.asp/i1/d37.htm
7. CGI in the wild - http://www.mcabee.org/lists/snort-users/May-01/msg00691.html
8. Joe Ellis: GCIA Practical Assignment ver 3.0 - http://www.giac.org/practical/Joe_Ellis_GCIA.doc
9. Nessus Scanner - http://www.nessus.org/
10. Snort List Archive – SMB Name Wildcard - http://archives.neohapsis.com/archives/snort/2000-01/0222.html
11. Incidents Mailing List Archive – SMB Name Wildcard - http://lists.jammed.com/incidents/2001/05/0034.html
12. SANS - Port 137 scan - http://www.sans.org/newlook/resources/IDFAQ/port_137.htm
13. RFC 1918 - http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1918.html
14. SANS – Global Incident analysis Center - Adore Worm - http://www.sans.org/y2k/adore.htm
15. Neohapsis – Snort List archive - http://archives.neohapsis.com/archives/snort/2002-06/0103.html
16. Bradley Urwiller: GCIA Practical Assignment ver 3.0 - http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf

17. CERT - sadmind alert - http://www.cert.org/advisories/CA-2001-11.html
18. Asia Pacific Network Information Centre: whois lookup; http://www.apnic.net/search/index.html
19. RIPE - http://www.ripe.net/ripencc/pub-services/db/whois/whois.html
20. American Registry for Internet Numbers - http://ww1.arin.net/whois/
21. SANS - http://www.sans.org/y2k/010600-0900.htm
22. Securiteam - Exploit code for sadmind - http://www.securiteam.com/exploits/3P5Q1Q0QAO.html
23. SANS – Top 20 vulnerabilities – May 2002 - http://www.sans.org/top20/top20_Oct01.htm
24. Ramen Worm analysis: http://www.sans.org/y2k/ramen.htm
25. Michael Holstein: GCIA Practical Assignment 3.1 http://www.giac.org/practical/Michael_Holstein_GCIA.doc
26. GCI Null Byte description – by Rain Forest Puppy - http://www.wiretrip.net/rfp/p/doc.asp/i1/d37.htm
27. Asia Pacific Network Information Centre: whois lookup; http://www.apnic.net/search/index.html
28. Snort: Writing Snort Rules; http://www.snort.org/docs/writing_rules/chap2.html#tth_chAp2
29. Blubster Configuration: http://www.blubster.net/php/print.php?sid=34
30. Blubster Music Community: http://www.blubster.com
31. Napster: http://www.napster.com
32. Napster Legal Case - http://news.findlaw.com/legalnews/lit/napster/ and court order - http://news.findlaw.com/hdocs/docs/napster/napster022102ord.pdf
33. Neohapsis: tcp ports listing; http://www.neohapsis.com/neolabs/neo-ports/
34. Tod Beardsley: GCIA Practical Assignment 3.1 http://www.giac.org/practical/Tod_Beardsley_GCIA.doc
35. PHP Database mailing list: http://www.php.net/mailing-lists.php