Practical Assignment for GCIA Certification, version 2.7a

TJ Vanderpoel

The Detects

Detect 1

Log 1

```
61.139.210.136 - - [26/Mar/2001:20:31:26 -0600] "GET http://www.s3.com HTTP/1.1" 302 5
61.139.210.136 - - [26/Mar/2001:20:31:28 -0600] "GET http://www.s3.com HTTP/1.1" 302 5
61.139.210.136 - - [26/Mar/2001:20:31:32 -0600] "GET http://www.s3.com HTTP/1.1" 302 5
```

Log 2

```
03/26-21:31:23.819505 61.139.210.136:3759 -> HOME.NET.BOUGY.41:80
TCP TTL:108 TOS:0x0 ID:25420 IpLen:20 DgmLen:148 DF
***AP*** Seq: 0x183E2921  Ack: 0x870619C  Win: 0x2238  TcpLen: 20
0x0000: 00 40 33 A3 00 9A 00 30 7B FB 8C 70 08 00 45 00  .@3....0{..p..E.
0x0010: 00 94 63 4C 40 00 6C 06 2E 28 3D 8B D2 88 00 00  ..cL@.l..(=.....
0x0020: 54 29 0E AF 00 50 18 3E 29 21 08 70 61 9C 50 18  T)...P.>)!.pa.P.
0x0030: 22 38 CF 64 00 00 47 45 54 20 68 74 74 70 3A 2F  "8.d..GET http:/
0x0040: 2F 77 77 77 2E 73 33 2E 63 6F 6D 20 48 54 54 50  /www.s3.com HTTP
0x0050: 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 77 77 2E  /1.1..Host: www.
0x0060: 73 33 2E 63 6F 6D 0D 0A 41 63 63 65 70 74 3A 20  s3.com..Accept:
0x0070: 2A 2F 2A 0D 0A 50 72 61 67 6D 61 3A 20 6E 6F 2D  */*..Pragma: no-
0x0080: 63 61 63 68 65 0D 0A 55 73 65 72 2D 41 67 65 6E  cache..User-Agen
0x0090: 74 3A 20 50 72 6F 78 79 48 75 6E 74 65 72 0D 0A  t: ProxyHunter..
0x00A0: 0D 0A                                            ..
```

1. Source of trace:
   This traffic was captured from a personal server running a web server to publish Friends and Family web pages. The server rarely receives hits from users outside a known range of addresses, so the logs are checked for all connections originating outside that norm.

2. Detect Generated By:
   Log-1 is the Apache access_log. To list the unknown hosts, I used:
   ```
   egrep –v 'host1.known.net|host2.known.net|host2.known.net' access_log > suspects
   ```
   Then to give a count of unknown hosts who connected:
   ```
   awk '{print $1}' suspects|sort|uniq –c
   ```
   Finally, check out the traffic from the suspects:
   ```
   egrep <ip from suspect file> access_log
   ```

   Log-2 is a binary packet log captured by the snort sensor on this network, trace as follows:
   To get the unix timestamp of the event:
   ```
   date –d 'Mar 26 2001' +%s
   ```
   I archive snort binary logs daily, saving them as snort.`date +%s`.log, so to examine the traffic of the suspect:
   ```
   snort –Xdvvr snort.9856[6789]*.log  host 61.139.210.136|more
   ```

3. Probability of Spoofing:
   This http connection requires a full 3-way handshake TCP connection, minimal probability of spoofing.

4. Attack Description:

By examining the payload of the packet from the snort log, it is easily recognizable that this attacker is attempting to use this web server to access a web page hosted on s3.com. By relaying thru this server, the attacker is able to mask his location and make it seem as if the source of his web-traffic were this server.

5. Attack Mechanism:
   A glance at the snort packet dump points to our culprit, Proxy Hunter.
   From http://www.securityfocus.com:

To:	Incidents
Subject:	Re: Strange Scan
Date:	Sun Dec 17 2000 20:44:05
Author:	geoff < geoff@cardboardtransmitter.net >
Message-ID:	<2euq3tob9eql8h6ukoan1mdet0tbte8dru@4ax.com>
In-Reply-To:	<006c01c06896$b537f3e0$6500a8c0@dubz.com.au>

```
Proxy Hunter is a little utility I used to use to find anonymous
proxies (i.e. people that have set up WinProxy poorly). All it
does is scan a range of IP's and attempt to retrieve a web page.
...And dont worry about http://www.s3.com, it just happens to be
the default page that Proxy Hunter attempts to retrieve.
```

➢ gleNN

Proxy Hunter is a widely utilized tool of the Proxys-4-All project, a group that maintains a list of proxies that allow unrestricted access. It should be noted that using the proxies at Proxys-4-All is most likely violating the security policy of the Proxy Server and may rise to illegality.

6. Correlations:
   http://www.securityfocus.com has several of these captures. It seems this tool has been in use quite some time. The site http://www.sans.org/y2k/010600-1515.htm shows it as the most frequent incident in January 2001; the earliest records of Proxy Hunter date back to 12/27/1999.

7. Evidence of active targeting:
   No evidence of active targeting, this was most likely a potential intruder scanning a subnet for later use. Since the web server is hosted by @home I get to see quite a large number of automated scans/attacks.

8. Severity
   (Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity
   (4+1)-(4+1)=0

   Criticality (4)-This is the primary server for many friends/family websites and also serves as the DNS for a few domains.

   Lethality (1)-The web server does not answer proxy requests, so the user would normally receive an error message from the server.

   System Countermeasures (4)- In this case, I have all bad URL requests to the server redirected to DocumentRoot/default.htm on the server. Proxy Hunter may have interpreted this as a positive result, making the attacker believe proxying is enabled, but the lack of further activity from the source host makes me believe that is not the case.

Network Countermeasures (1)-Since there is no snort rule set to search for this probe and the web server open is to the public, nothing at the network layer would have alerted on this or attempted to stop it.

9. Defensive recommendation:
   Added snort rule to search for ProxyHunter:

```
        Alert tcp $EXTERNAL_NET any > $HOME_NET 80 (msg:"Proxy \
          Hunter Probe";content:"ProxyHunter";)
```

   Blocked source host on firewall:

```
        iptables –I BLOCKED –s  61.139.210.136 –j DROP
```

10. Test Question

```
61.139.210.136 - - [26/Mar/2001:20:31:26 -0600] "GET http://www.s3.com HTTP/1.1" 302 5
61.139.210.136 - - [26/Mar/2001:20:31:28 -0600] "GET http://www.s3.com HTTP/1.1" 302 5
61.139.210.136 - - [26/Mar/2001:20:31:32 -0600] "GET http://www.s3.com HTTP/1.1" 302 5
```

The pattern above shows:

a. A successful webpage retrieval from www.s3.com
b. Redirection by the web server to www.s3.com
c. Client at 61.139.210.136 using the web server as a proxy
d. www.s3.com receiving a webpage from server at 61.139.210.136

Detect 2

Log 1

```
messages.1:Feb 11 19:37:58 bougy portsentry[648]: attackalert: SYN/Normal scan from host: \
        c540323-a.alntn1.tx.home.com/24.179.82.24 to TCP port: 6000
messages.1:Feb 11 19:37:58 bougy portsentry[659]: attackalert: UDP scan from host: \
        c540323-a.alntn1.tx.home.com/24.179.82.24 to UDP port: 111
messages.1:Feb 11 19:37:58 bougy portsentry[648]: attackalert: Host 24.179.82.24 has been blocked \
        via dropped route using command: "/usr/local/sbin/iptables -A INPUT -s 24.179.82.24 -j DROP"
messages.1:Feb 11 19:37:58 bougy portsentry[659]: attackalert: UDP scan from host: \
        c540323-a.alntn1.tx.home.com/24.179.82.24 to UDP port: 111
messages.1:Feb 11 19:37:59 bougy portsentry[659]: attackalert: Host 24.179.82.24 has been blocked \
        via dropped route using command: "/usr/local/sbin/iptables -A INPUT -s 24.179.82.24 -j DROP"
```

Log 2

```
Feb 11 19:37:59 bougy snort: IDS021 - RPC - portmap-request-nisd: \
        24.179.82.24:810->HOME.NET.BOUGY.41:111
Feb 11 19:37:59 bougy snort: IDS013 - RPC - portmap-request-mountd: \
        24.179.82.24:960 -> HOME.NET.BOUGY.41:111

Feb 11 19:41:12 bougy snort: MISC-DNS-version-query: 24.179.82.24:4620 ->
HOME.NET.BOUGY.53:53
Feb 11 19:41:12 bougy snort: IDS021 - RPC - portmap-request-nisd: 24.179.82.24:815-
>HOME.NET.BOUGY.53:111
Feb 11 19:41:12 bougy snort: BIND Shell: 24.179.82.24:4917 -> HOME.NET.BOUGY.41:31337

Feb 11 19:46:00 bougy snort: Netbus/GabanBus: 24.179.82.24:15771 -> HOME.NET.BOUGY.53:12345
Feb 11 19:46:01 bougy snort: Back Orifice: 24.179.82.24:15776 -> HOME.NET.BOUGY.53:31337
Feb 11 19:46:02 bougy snort: IDS255 - DDoS shaft handler to agent: \
    24.179.82.24:15824 -> HOME.NET.BOUGY.53:18753
```

Log 3

```
Feb 11 19:38:09 bougy snort: IDS148 - CVE-1999-0183 - TFTP Write: \
    24.179.82.24:63727 -> HOME.NET.BOUGY.41:69

02/11-20:38:09.701817 24.179.82.24:63727 -> HOME.NET.BOUGY.41:69
UDP TTL:124 TOS:0x0 ID:15502 IpLen:20 DgmLen:72
Len: 52
0x0000: 00 40 33 A3 00 9A 00 30 7B FB 8C 70 08 00 45 00   .@3....0{..p..E.
0x0010: 00 48 3C 8E 00 00 7C 11 2A 70 00 00 52 18 00 00   .H<...|.*p..R...
0x0020: 54 29 F8 EF 00 45 00 34 8B 81 00 02 2F 74 6D 70   T)...E.4..../tmp
0x0030: 2F 43 79 62 65 72 43 6F 70 2E 74 66 74 70 2E 76   /CyberCop.tftp.v
0x0040: 75 6C 6E 65 72 61 62 69 6C 69 74 79 00 6E 65 74   ulnerability.net
0x0050: 61 73 63 69 69 00                                 ascii.
```

1. Source of trace: These logs were captured from the gateway for a small office network running Redhat Linux, kernel 2.4.1, with netfilter performing firewall functionality. The network is used primarily for business purposes, serving web applications, e-mail and providing Internet access for the office.

2. Detect Generated By: Log 1 comes from a portsentry report to syslog, which set off SWATCH and alerted the analyst via pager. Log 2 shows snippets of the alerts generated by snort to the syslog shortly thereafter. Log 3 includes a snort syslog alert as well as the decoded packet that it references.

3. Probability of Spoofing: Some of these could certainly be spoofed probes, but many complete a 3-way handshake connection. The sheer volume of the probe indicates that the source machine could either be compromised and is being used as a remote reconnaissance gatherer, or is operated by an unskilled or new administrator (script kiddie).

4. Attack Description: This intruder is probing the TCP and UDP ports on the gateway, gaining information about the services that the network may provide and executing remote exploits in rapid succession. Evidence that the attack was automated comes from both the speed of the packets generated and, as shown in the activity from Log 2, the skipping up a few steps in the MY.NET.84.0/24 subnet to MY.NET.84.53 within 4 minutes of the original probe on MY.NET.84.41

5. Attack Mechanism: Log 3 gives away the tool used to automate the scan, CyberCop scanner. In this packet, the CyberCop scanner attempts to write the file CyberCop.tftp to the /tmp directory on MY.NET.84.41 using the TFTP protocol. This CyberCop activity, identified by CVE-1999-0183, targets Linux boxes running a vulnerable TFTP server, allowing write privileges outside the TFTP directory.

6. Correlations: CVE-1999-0183, Whitehats IDS 148.

7. Evidence of active targeting: No evidence of previous traffic from source. Most likely a first time feature-test of a new CyberCop installation.

8. Severity:

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

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

   Criticality (4)-This is the primary server for this office, including DNS server and Internet gateway.

   Lethality (1)-The gateway does not run any of the services the probes were searching for.

System Countermeasures (4)- PortSentry caught this probe from its inception and blocked the source address. Snort also recorded all activity to a MySQL database for viewing with A.C.I.D, and SWATCH informed an analyst of the activity within 30 seconds.

Network Countermeasures (4)-The gateway blocked the source before any reconnaissance could be gathered, even if vulnerabilities existed, the source would never have received the information.

9. Defensive recommendation: Send detect to ISP of source host; if the machine is compromised, they can inform their user and eliminate the attack point.

10. Test Question

```
Feb 11 19:37:59 bougy snort: IDS021 - RPC - portmap-request-nisd: \
        24.179.82.24:810->HOME.NET.BOUGY.41:111
Feb 11 19:37:59 bougy snort: IDS013 - RPC - portmap-request-mountd: \
        24.179.82.24:960 -> HOME.NET.BOUGY.41:111

Feb 11 19:41:12 bougy snort: MISC-DNS-version-query: 24.179.82.24:4620 ->
HOME.NET.BOUGY.53:53
Feb 11 19:41:12 bougy snort: IDS021 - RPC - portmap-request-nisd: 24.179.82.24:815-
>HOME.NET.BOUGY.53:111
Feb 11 19:41:12 bougy snort: BIND Shell: 24.179.82.24:4917 -> HOME.NET.BOUGY.41:31337

Feb 11 19:46:00 bougy snort: Netbus/GabanBus: 24.179.82.24:15771 -> HOME.NET.BOUGY.53:12345
Feb 11 19:46:01 bougy snort: Back Orifice: 24.179.82.24:15776 -> HOME.NET.BOUGY.53:31337
Feb 11 19:46:02 bougy snort: IDS255 - DDoS shaft handler to agent: \
24.179.82.24:15824 -> HOME.NET.BOUGY.53:18753
```

What can be determined by this trace?

a. 24.179.82.24 is ping scanning MY.NET.84.53
b. 24.179.82.24 has previous reconnaissance of MY.NET.84.0/24
c. 24.179.82.24 is spoofed
d. 24.179.82.24 is scanning the MY.NET.84.53 subnet with an automated vulnerability scanner

answer:d

Detect 3

Log 1

ID	Signature	TimeStamp	Source Address	Dest. Address	Layer 4 Protocol
#0-(3-235)	IDS277 - NAMED Iquery Probe	2001-04-02 18:27:45	66.31.116.113	DSL.NET.156.179	UDP

Log 2

```
18:27:45.186465 h00e0296df973.ne.mediaone.net.4724 > adsl-DSL-NET-156-
180.dsl.rcsntx.swbell.net.domain:  43981 inv_q+ [b2&3=0x980] (23)
0x0000   4500 0033 a4b7 0000 2d11 54e4 421f 7471         E..3....-.T.B.tq
0x0010   40da 9cb4 1274 0035 001f 0057 abcd 0980         @....t.5...W....
0x0020   0000 0001 0000 0000 0000 0100 0120 2020         ................
0x0030   2002 61                              .          .a
```

Log 3

```
18:18:47.086170 h00e0296df973.ne.mediaone.net.2854 > \
  adsl-DSL-NET-156-177.dsl.xxx.swbell.net.domain: S 3962152800:3962152800(0) \
  win 32120 <mss 1460,sackOK,timestamp 8091406 0,nop,wscale 0> (DF)
18:18:47.089628 h00e0296df973.ne.mediaone.net.2855 > \
  adsl-DSL-NET-156-178.dsl.xxx.swbell.net.domain: S 3956914051:3956914051(0) \
  win 32120 <mss 1460,sackOK,timestamp 8091406 0,nop,wscale 0> (DF)
18:18:47.112794 adsl-DSL-NET-156-178.dsl.xxx.swbell.net.domain > \
  h00e0296df973.ne.mediaone.net.2855: R 0:0(0) ack 3956914052 win 0
```

1. Source of trace: This capture took place on a home ADSL connection. The addresses from DSL.NET.156.179 and dot-180 reside on a separate ADSL line from the DSL.NET.156.177 and dot-178 addresses. Both ADSL lines support small home offices, serve web pages, e-mail, DNS and provide user level Internet access.

2. Detect Generated By: the 177/178 network captures IP traffic using the Shadow NIDS; the 179/180 network utilizes Snort with A.C.I.D reporting. Log 1 is the A.C.I.D. event, Log 2 is that packet, dumped by Snort and Log 3 are packets captured with Shadow and decoded with tcpdump.

3. Probability of Spoofing: Since a three-way-handshake is never completed, there is no way to guarantee this address wasn't spoofed. If the attacker were using a spoofed source, however, the information about the services on DSL.NET would not have been returned to that source, so he/she would need an intermediary packet sniffer to trap the results in-transit.

4. Attack Description: The suspect is scanning UDP port 53 for versions of the BIND nameserver, possibly to use for a later penetration. From IDS277:

   > A remote user attempted to determine if a nameserver supports IQUERY. This signature often indicates a pre-attack probe used to locate vulnerable servers running named.

   The probe shows evidence of automation as it is hitting the DSL.NET.156.0/24 subnet sequentially and in rapid succession.

5. Attack Mechanism: This packet differs from the hard-coded values in the exploit `bof-test.c by Josh Drake`. The source IP in the original exploit was 4.3.2.1 and the TTL was 31337. Either the bof-test.c code was modified and reused, or a new probe was concocted to search for the Iquery response.

6. Correlations: IDS277, CVE-1999-0009, BugtraqID 134

7. Evidence of active targeting: No previous traffic from source address to either of the DSL networks indicates lack of active targeting; however, since these are spoofable packets the actual source could indeed be targeting one or both of these networks.

8. Severity:

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

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

   Criticality (5)-These are servers and workstations used by the networks users for telecommuting. Outages or compromise of the systems could be business crippling.

Lethality (3)- This exploit would allow root access if an unpatched DNS server were on the network.

System Countermeasures (4)- All O/S's are patched, but DNS is listening on both networks, any information gathered about the servers could aid in later penetrations. The DNS servers also return bogus information about their versions.

Network Countermeasures (3)-The networks were protected from DSN exploits over TCP via the firewall and Snort is in active response mode against obsolete DNS query types, sending RSTs to TCP queries and ICMP Unreachables to UDP queries.

9. Defensive recommendation:
   Log all traffic from source address and verify that all DNS server software is updated. Report to GIAC.

10. Test Question

```
18:27:45.186465 h00e0296df973.ne.mediaone.net.4724 > adsl-DSL-NET-156-
180.dsl.rcsntx.swbell.net.domain:  43981 inv_q+ [b2&3=0x980] (23)
0x0000   4500 0033 a4b7 0000 2d11 54e4 421f 7471         E..3....-.T.B.tq
0x0010   40da 9cb4 1274 0035 001f 0057 abcd 0980         @....t.5...W....
0x0020   0000 0001 0000 0000 0000 0100 0120 2020         ................
0x0030   2002 61                        .                .a
```

This packet is an example of:

a. Response to a successful DNS query
b. Reverse DNS host lookup.
c. DNS domain transfer
d. Inverse DNS query

Detect 4

Log 1

```
Mar 26 13:00:05 tjv snort: IDS291 - MISC - Shellcode x86 stealth NOP: \
      12.4.218.41:80 -> 10.20.59.11:34821
```

Log 2

```
13:58:47.377276 tjv.E.NET.34821 > 12.4.218.41.www: S 1549488150:1549488150(0) win 5840 \
            <mss 1460,sackOK,timestamp 43491220 0,nop,wscale 0> (DF)
13:58:47.425384 12.4.218.41.www > tjv.E.NET.34821: S 558175390:558175390(0) ack 1549488151 \
            win 17376 <mss 1460,nop,wscale 0,nop,nop,timestamp 2064994 43491220> (DF)
13:58:47.425480 tjv.E.NET.34821 > 12.4.218.41.www: . ack 1 win 5840 <nop,nop,timestamp \
            43491225 2064994> (DF)
13:58:47.425808 tjv.E.NET.34821 > 12.4.218.41.www: P 1:384(383) ack 1 win 5840 \
            <nop,nop,timestamp 43491225 2064994> (DF)
13:58:47.478424 12.4.218.41.www > tjv.E.NET.34821: . ack 384 win 16993 <nop,nop,timestamp \
            2064994 43491225> (DF)
13:58:47.521105 12.4.218.41.www > tjv.E.NET.34821: . 1:1449(1448) ack 384 win \
            17376 <nop,nop,timestamp 2064994 43491225> (DF)
13:58:47.521216 tjv.E.NET.34821 > 12.4.218.41.www: . ack 1449 win 8688 \
            <nop,nop,timestamp 43491235 2064994> (DF)
13:58:47.523777 12.4.218.41.www > tjv.E.NET.34821: P 1449:2049(600) ack 384 win \
            17376 <nop,nop,timestamp 2064994 43491225> (DF)
13:58:47.523859 tjv.E.NET.34821 > 12.4.218.41.www: . ack 2049 win 11584 \
      <nop,nop,timestamp 43491235 2064994> (DF)
```

1. Source of trace: These packets were captured on an internal network at an e-commerce corporation. The client (tjv.net1.com) is the workstation of a security analyst with no ports or services listening.

2. Detect Generated By: Log 1 is a syslog alert generated by snort, Log 2 is a tcpdump decode of the beginning of the www session.

3. Probablility of Spoofing: Not spoofed, full TCP three-way-handshake.

4. Attack Description: stealth NOP's are used to send buffer overflow attacks to vulnerable daemons while avoiding intrusion detection systems. As described by Xtremist in "Writing anti-IDS shellcode":

   ```
     Anti-IDS tactic:
     The main problem here is the presence of NOP's in the shellcode.
   Exploits usually pad the stack with NOP's so that the return
   address dosent have to be exact. It is this NOP which is the
   problem. The main shellcode (which probably start execve or append
   a line to passwd) need not be changed because it dosent contain
   NOP's. The problem lies here -
   ```

   for(i=0;i<(LEN-strlen(shellcode));i++){*(bof+i)=0x90;)

   ```
   where the beginning of the stact gets padded with NOP's. NOP is
   used only to jump to the next instruction without any modification
   to execution of the assembly code. NOP=No OPeration. But the same
   function can be achieved by using a jump to the next instrucion
   (jmp 0x00).
   ```

   Since blackhats learned that IDS systems look for NOPs, it was necessary that they find an alternative in the one-upmanship contest in which we participate daily.

5. Attack Mechanism: The web server here generated the Snort-offending traffic during a normal web session. The client initiated a large download (evidenced by the packets' size) and the jmp0x00 instructions turned out to be a legitimate part of the download (source code of the Pentium C Compiler).

6. Correlations: IDS291, post from Max Vision on the IDS Mailing List, "Writing anti-IDS shellcode"

7. Evidence of active targeting: None, this is a false positive.

8. Severity

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

   (3+1)-(5+4)=-6

   Criticality (3)-This is a Unix user workstation, elevated to a 3 because of the sensitive Incident Handling information stored on it.

   Lethality (1)- As a false alarm, the lethality is non-existent.

   System Countermeasures (5)- This is a secure workstation, with no daemons and no file shares or other network services offered.

Network Countermeasures (4)-The network is protected by a screening router and a firewall; the connection was allowed because it was initiated from the inside.

9. Defensive recommendation: Watch this rule for future false alarms. Grab an exploit that utilizes stealth NOP's and try to modify the rule, enabling it to differentiate between false alarms and actual shellcode execution attempts.

10. Test Question

```
13:58:47.377276 tjv.E.NET.34821 > 12.4.218.41.www: S 1549488150:1549488150(0) win 5840 \
        <mss 1460,sackOK,timestamp 43491220 0,nop,wscale 0> (DF)
13:58:47.425384 12.4.218.41.www > tjv.E.NET.34821: S 558175390:558175390(0) ack 1549488151 \
        win 17376 <mss 1460,nop,wscale 0,nop,nop,timestamp 2064994 43491220> (DF)
13:58:47.425480 tjv.E.NET.34821 > 12.4.218.41.www: . ack 1 win 5840 <nop,nop,timestamp \
        43491225 2064994> (DF)
13:58:47.425808 tjv.E.NET.34821 > 12.4.218.41.www: P 1:384(383) ack 1 win 5840 \
        <nop,nop,timestamp 43491225 2064994> (DF)
13:58:47.478424 12.4.218.41.www > tjv.E.NET.34821: . ack 384 win 16993 <nop,nop,timestamp \
        2064994 43491225> (DF)
13:58:47.521105 12.4.218.41.www > tjv.E.NET.34821: . 1:1449(1448) ack 384 win \
        17376 <nop,nop,timestamp 2064994 43491225> (DF)
13:58:47.521216 tjv.E.NET.34821 > 12.4.218.41.www: . ack 1449 win 8688 \
<nop,nop,timestamp 43491235 2064994> (DF)
```

This trace shows:

a. Normal 3-way handshake with tjv.e.net connecting to web server at 12.4.18.41
b. Webserver at 12.4.218.41 initiating a connection with tjv.e.net
c. RPC session between 12.4.218.41 and tjv.e.net
d. None of the above

Answer: a.

Detect 5

Log 1

Signature	TimeStamp	Source Address	Dest. Address	Layer 4 Protocol
IDS155 - PING Delphi-Piette Windows	2001-03-28 14:54:31	195.29.34.17	E.NET.61.108	ICMP
IDS155 - PING Delphi-Piette Windows	2001-03-28 14:54:31	195.29.34.17	E.NET.61.108	ICMP

Log 2

```
000 : 50 69 6E 67 69 6E 67 20 66 72 6F 6D 20 44 65 6C   Pinging from Del
010 : 70 68 69 20 63 6F 64 65 20 77 72 69 74 74 65 6E   phi code written
020 : 20 62 79 20 46 2E 20 50 69 65 74 74 65 20 20 20    by F. Piette
030 : 20 20 20 20 20 20 20 20
```

1. Source of trace: This was capture on the same Corporate network as Detect 4. The target of the PING is a private, developers' subnet that should never be accessed from the Extranet.

2. Detect Generated By: Two separate Snort Sensors, one inside the main Firewall and one inside the Firewall protecting the private subnet, alerted on this packet.

3. Probability of Spoofing: ICMP is not a connection-oriented protocol, so likelihood of spoof is high.

4. Attack Description: A PING is a simple test to see if a host is alive. An ICMP echo request is sent and if an echo reply is returned, the recipient gets confirmation that the endhost can be contacted via IP.

5. Attack Mechanism: The very plain description of the packet in Log 2 shows Delphi generated this PING with code written by F. Piette.

6. Correlations: CAN-1999-0523, IDS155

7. Evidence of active targeting: Since this private subnet should not even be mapped to the Extranet, the evidence of any packet from the Internet reaching this subnet is disturbing. The lack of previous traffic from this source offers no consolation, as ICMP is easily spoofed. Although the target of this PING did not respond, an "ADMIN PROHIBITED" message was returned to the suspect's host, possibly giving information as to the mapping of our internal network.

8. Severity

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

   (3+1)-(5+4)=-6

   Criticality (5)-This is a highly sensitive developer's subnet. Compromise of any machine, gateway device, security policy, or network information could have a substantial impact on business.

   Lethality (2)- This is not a PING meant to destroy or disable, only to probe/map.

   System Countermeasures (5)- The system in question will not respond to pings, has the latest OS revisions and does not listen on any UDP or TCP port.

   Network Countermeasures (2)-The network is protected by a screening router and two firewalls, but the ruleset on the innermost firewall had been purged the previous evening, allowing this offending packet.

9. Defensive recommendation: Re-implement the secure rule-set on the innermost firewall, initiate process to check the firewall rules regularly.

10. Test Question

```
000 : 50 69 6E 67 69 6E 67 20 66 72 6F 6D 20 44 65 6C   Pinging from Del
010 : 70 68 69 20 63 6F 64 65 20 77 72 69 74 74 65 6E   phi code written
020 : 20 62 79 20 46 2E 20 50 69 65 74 74 65 20 20 20    by F. Piette
030   : 20 20 20 20 20 20 20 20
```

The packet above contains:

a. Numerous NOPs
b. IP address of F. Piette
c. Information about the application and operating system of the source host
d. Hidden shell code

Answer: c.

The Whitepaper

Deploying Open Sourced Network Intrusion Detection For The Enterprise.

TJ Vanderpoel
March 04, 2001

1. Introduction

   In order to satisfy Security Professionals' needs to monitor their networks, an abundance of Enterprise Management Systems (EMS) have surfaced that tout themselves as "All-in-one" Network Monitoring systems. These frequently have a component that performs some level of Network Intrusion Detection and that displays the alerts from the intrusion detection (ID) sensors on the same console as other Monitored Services, Networks or Systems. While many of these products are indeed quality applications, the resulting integration with general monitoring services may be undesirable, giving Monitoring Operations Staff the ability and responsibility of analyzing Intrusion Incidents- something for which entry-level engineers generally are not qualified. In addition, these multifarious application suites generally require higher-end equipment, which may have the effect of a company choosing far fewer sensors than it might if sensors could be built from obsolete stock or far less expensive new equipment. An EMS can be expensive even without the necessary costly equipment, which must also be added to the price of the Operating System needed to run the applications on the sensors and console. To alleviate these issues and give Security Teams the ability to maintain complete control of their NIDS, a variety of solid open-sourced software can be loaded onto low-power, relatively inexpensive equipment while providing the same features as a commercial EMS NIDS component. This document can be looked at as a guideline for such an implementation.

2. What IS Open Source?

   Open source software is software like any other; the difference is in the licensing philosophies:

   From the preamble of the GNU Public License:

   > *The licenses for most software are designed to take away your freedom to share and change it.*

   Concurrently, the Open Source Initiative (OSI) website claims:

   > *The basic idea behind open source is very simple. When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing.*

   Open source licenses are meant to make the distribution of source code for applications free for everyone. They protect the author by stipulating that redistributions of the software must include the full source code, as well as credit to the original author. They basically give everyone the right to share and change the original source code, provided they do not attempt to infringe on the rights of others to do the same. To help maintain standards, the Open Source Initiative offers an Open Source Definition and also serves as a certifying authority for open source licenses. It can be viewed at http://www.opensource.org/docs/definition.html. OSI also maintains a list of accepted open source licenses; for our purposes, these will be:

   The PHP License – http://www.php.net/license/2_02.txt
   The Apache Software License – http://www.opensource.org/license/apachepl.html

The GNU Public License (GPL) – http://www.opensource.org/license/gpl-license.html

Support for software used in this implementation can be supported by a variety of 3rd party consultants as well as the company maintaining the software package (i.e. MySQL support can be purchased from http://www.mysql.com). This support differs from most companies in that you rarely talk to a "support technician". For example, MySQL maintains a list of developers currently logged in to their system. When you have a problem, you receive this list of developers and their contact information after logging in to the MySQL website. Depending on the level of support purchased (from e-mail only to full 24/7 phone support), you then contact the developer and he/she handles the support request directly. Most open source software is readily available in both stable and development releases; sites that publish open source software include Freshmeat, Sourceforge, and the GNU (GNU's Not Unix!) organization.

3. What will I need?

Here's a list of required software packages, with versions and download sites, used for the system described in this document:

Redhat Linux 6.2 – http://www.redhat.com
Operating system for console and sensors

Bastille Linux version 1.1.1 – http://bastille-linux.sourceforge.net
Security hardening script for the Redhat Linux Operating System

FreeS/WAN version 1.8 – http://www.freeswan.org
VPN software to allow encrypted transmission of alerts/configs/etc

Snort version 1.7 – http://www.snort.org
Intrusion Detection software for the sensors, the core of the system

Libpcap version 0.6.2 – http://www.tcpdump.org
Library that handles the actual capturing of packets from the network interface; Snort depends upon it

Apache Web Server version 1.3.19 – http://www.apache.org
Web server that will run the console and reporting software

PHP version 4.0.4 – http://www.php.net
Scripting language support for A.C.I.D on Apache

Mod_ssl version 2.8.2-1.3.19 – http://www.modssl.org
Enables Apache to server SSL-secured web pages

OpenSSL version 0.9.6 – http://www.openssl.org
Library that performs encryption/decryption for mod_ssl (and OpenSSH)

OpenSSH version 2.5.2 – http://www.openssh.com
Network connectivity tools using the SSH protocol suite, for remote administration

MySQL[1] version 3.23.36 – http://www.mysql.com
Database to store alerts

A.C.I.D. version 0.9.5 – http://www.cert.org/kb/acid
Analysis Console Engine for Intrusion Detection, a PHP-based analysis engine to search and process the data generated by the Snort sensors

SWATCH version 3 -- http://www.stanford.edu/~atkins/swatch/

The Simple WATCHer watches a syslog file and takes action based on user-defined rules; used for alerting Information Security Personnel of possible intrusion events

4. What about hardware?

This implementation is designed to allow for a multitude of hardware type to be used. In the practical sense, here are the guidelines for each of the 3 components necessary for an Enterprise Deployment.

ID Sensor—The worker bees of the system. They sit in strategic locations and simply listen to all the traffic passing by them and optionally take action based on pre-defined or dynamic rules. For the ID sensors, ease of deployment and replication should be a top priority. Ideally, unused x86 architecture capable machines with speeds preferably greater than 233 Mhz, 96MB of RAM and 2GIG or more of hard drive space can be found in a closet and given a new lease on life as ID sensors. If such hardware is not available, new equipment need not be more than 500Mhz, with the same minimums of RAM and hard drive space as above. In essence, other architectures such as those found in Macintosh or SPARC computers could be utilized as well, provided the engineers had proficiency installing and administrating Linux on that particular type. This implementation used 5 identical Pentium 266 laptop computers with 2.1 GIG hard drives and 96MB of ram.

ID Console---The back end. This is where the data gets gathered, processed, and presented to the end user. This will need to run the following services:

Apache Web Server with mod_ssl and OpenSSL for Secure Sockets Layer support
MySQL Relational Database Server
PHP Scripting Language to support the web applications for the remote console

The hardware needed to support this depends on factors such as: how many sensors are reporting to the console, how many administrators are using the console concurrently and how much data is being stored on the console (how much traffic is being captured). This implementation used a Pentium 300 with 96 MB of RAM and two 20GIG SCSI hard drives running a RAID 2 configuration. The RAID configuration was chosen to give the greatest reliability based on the hardware available, but the recommended Production Implementation should include a truly redundant central ID Console, a feature easily implemented with Redhat Linux 6.2 using its Piranha Clustering Software.

Remote Console---This will be the workstation that the Security Analyst or monitoring team will use to interface with the ID Console. Any workstation capable of running a GUI Web Browser should suffice, although the A.C.I.D. web interface was only tested on Netscape 4, Netscape 6 and Internet Explorer 5. For administering the ID Console, this workstation should have software capable of communicating via the SSH Protocol Suite.

5. How do I do it?

Setup begins with installing Linux on each of the ID Sensors and the ID console. This can be done individually on each machine with a Linux distribution CD or network installation. Optionally, the installation of the ID Sensors can be automated either with an installation script or by mirroring the hard drives of the Sensors from one installation. In the case of an automated installation using Redhat Linux, a server installation should be chosen from the installer options, with all unnecessary applications being left uninstalled, such as Web Server (this will be installed later), DNS server, anonymous FTP server, News Server, etc. See the SANS Securing Linux Step-By-Step guide (available from http://www.sansstore.org) and the documentation from the

Redhat distribution (http://www.redhat.com/apps/support/documentation.html) for a more comprehensive description. Once installed, the Linux Kernel should be rebuilt using the most current stable 2.2.X version. This can be found at http://www.kernel.org and at this time is version 2.2.18. If possible, it is recommended to build a monolithic kernel as opposed to as modular (see http://linuxdocs.org/Kernel-HOWTO.html). Support for ipchains, the native firewall for Linus 2.2.x systems, should be enabled (see http://linuxdocs.org/HOWTOs/Firewall-HOWTO.html). After rebuilding the kernel, it's time to finish the lockdown by running Bastille-Linux. What is Bastille-Linux?

> Bastille is set of open source scripts designed to harden a virgin Red Hat … installation. … Unneeded daemons are stopped, logging enabled/improved, SUID and file permissions tightened, account security improved and even a *chroot* environment is provided for DNS servers.

This excerpt comes from "Hardening Red Hat Linux with Bastille" by Sean Boran, available at http://www.securityportal.com/cover/coverstory20000501.html, a useful place to begin for those unfamiliar with Bastille-Linux. Please reference http://bastille-linux.sourceforge.net/ for the most recent announcements, changes, documentation and developments. After completing the Bastille lockdown process, FreeS/WAN should be installed and configured on the ID sensors and console (http://www.freeswan.org/freeswan_trees/freeswan-1.8/doc/install.html#install), to enable a secure communications tunnel. Before beginning this step, it is important to note the IP addresses the ID sensors and console will use on the network. After configuring FreeS/WAN and testing network connectivity between the sensors and console, the rest of the applications can be installed.
For the ID Console:

I. Install OpenSSL---An rpm for the installation can be downloaded from the Redhat website; alternatively the source can be downloaded directly from http://www.openssl.org. Installing the current version (0.9.6) from source is as follows:

a. untar the source file-
```
tar zxvf  openssl-0.9.6.tar.gz
```
b. change to the source directory and configure for compiling-
```
cd openssl-0.9.6
./configure
```
c. compile and install-
```
make > build.log
make install > install.log
```
d. link the libraries-
```
echo /usr/local/lib >> /etc/ld.so.conf
ldconfig
```

II. Install MySQL, Apache, mod_ssl and PHP. Again, rpms are available, but source installs are preferred using the latest source from the websites referenced in section 3. The tarballs should be downloaded to the same directory, in this case /usr/local/src.

a. untar the source files-
```
tar zxvf  mysql-3.23.35.tar.gz
tar zxvf apache_1.3.19.tar.gz
tar zxvf mod_ssl-2.8.1-1.3.19.tar.gz
tar zxvf php-4.0.4.tar.gz
```
b. Build and install MySQL
```
groupadd mysql
useradd -g mysql mysql
cd mysql-3.23.35
./configure --prefix=/usr/local/mysql
make > build.log && make install > install.log
scripts/mysql_install_db
echo /usr/local/mysql/lib/mysql >> /etc/ld.so.conf
```

```
        ldconfig
```

c. Build and install the Web Server

```
        cd mod_ssl
        ./configure --with-apache=../apache_1.3.19 --with-ssl=../openssl-0.9.6 --enable-\
                shared=ssl --enable-shared=most --enable-module=max
        cd ../apache_1.3.19
        ./configure --enable-shared=ssl --enable-shared=most --enable-module=max \
                --enable-rule=EAPI --prefix=/usr/local/apache
        make > build.log && make certificate TYPE=custom
        make install > install.log
        cd ../php-4.0.4
        ./configure --with-mysql=/usr/local/mysql --with- \
                apxs=/usr/local/apache/bin/apxs

        make > build.log && make install > install.log
```

d. Start the database and create users

```
        BINDIR=/usr/local/mysql/bin; export BINDIR
        $BINDIR/safe_mysqld &
        $BINDIR/mysqladmin –u root password <your_password>
        $BINDIR/mysql –p
                mysql> \u mysql
                mysql> DELETE FROM user WHERE User = ‘’;
                mysql> DELETE FROM user WHERE Password = ‘’;
                mysql> GRANT ALL PRIVELEGES ON *.* TO dba@localhost  \
                        IDENTIFIED BY ‘password’;
                mysql> CREATE DATABASE snort;
                mysql> GRANT INSERT,SELECT,DELETE ON snort.* TO \
                        snort@localhost IDENTIFIED BY ‘password’;
                mysql> \q
```

III. Configure Apache and install A.C.I.D.

a. Edit /usr/local/apache/conf/httpd.conf, setting values as follows

```
MinSpareServers 1
MaxSpareServers 3
StartServers 2
MaxClients 5
Port 443
SSLRequireSSL (in <Directory /usr/local/apache/htdocs>)
ServerSignature Off
```

b. create .htaccess file and .htpasswd file (or other authentication system for apache, auth_user.php was used in the example system, available from bougyman@i.am)

c. extract acid source

```
tar zxvf acid-0.9.6.tar.gz (from /usr/local/src)
mkdir /usr/local/apache/htdocs/acid
cd acid-0.9.6
cp *.php *.html *.css /usr/local/apache/htdocs/acid
chmod 755 /usr/local/apache/htdocs/acid
chmod 644 /usr/local/apache/htdocs/acid/*
```

IV. Install Snort and create databases

a. Remove Redhat tcpdump and install libpcap

```
rpm –e tcpdump
tar zxvf libpcap-0.6.2.tar.gz
cd libpcap-0.6.2
./configure
make > build.log && make install > install.log
make install-incl
ldconfig
```

b. unzip, build, and install Snort

```
tar zxvf snort-1.7.tar.gz (from /usr/local/src)
cd snort-1.7
./configure --with-mysql=/usr/local/mysql
make > build.log && make install > install.log
```

c. Create snort data directory and databases

```
mkdir /storage/snort (used in example, make this a partition with a lot of space, not
    the same as the / filesystem)
mysql –u dba –p snort < /usr/local/src/snort-1.7/contribs/create_mysql
mysql –u dba –p snort < /usr/local/src/acid-0.9.6/create_acid_tbls.sql
```

V. Configure for remote syslog and install SWATCH

a. edit /etc/rc.d/init.d/syslog, changing the start command to :

```
daemon syslogd –r –m 0 (-r allows remote logging)
```

b. install SWATCH

```
tar xvf latest.tar (from /usr/local/src)
cd swatch-3.0.1
perl Makefile.PL
make > build.log && make test
make install > install.log (if tests are successful)
```

VI. Install OpenSSH

a. unzip, build, and install

```
tar zxvf openssh-2.3.0p1.tar.gz (from /usr/local/src)
cd openssh-2.3.0p1
./configure
make > build.log && make install > install.log
```

VII. Finalize configurations and start services

a. edit /usr/local/apache/htdocs/acid/acid_conf.php, setting the values for:

```
$alert_dbname
$alert_user
$alert_password (use the values chosen in part d. of step II.)
```

b. Configure SWATCH

edit ~/.swatchrc , originally, a single rule will work:

```
watchfor /snort/
    echo
    beep 2
    mail=admin@one.dot.com (wherever you want alerts sent)
```

c. Start everything

```
/usr/local/apache/bin/startssl
```

```
/usr/local/sbin/sshd
/usr/bin/swatch –daemon &
```

d. Test it
Open a web browser to http://console.hostname.com/acid

The ID console is now complete. To begin the ID sensors, install OpenSSL as you did for the console, Step I, as well as openssh, then

Install MySQL client only
```
 cd mysql-3.23.35
 ./configure –wihtout-server --prefix=/usr/local/mysql
 make > build.log && make install > install.log
 echo /usr/local/mysql/lib/mysql >> /etc/ld.so.conf
 ldconfig
```

Complete the sensor by installing snort as in part b. of section IV. above. After installation, create the snort data directory structure:
```
mkdir /storage/snort
mkdir /storage/snort/conf
mkdir /storage/snort/logs
mkdir /storage/snort/bin
chmod –R 700 /storage/snort/*
```

And copy the snort configuration files to the configuration directory:
```
cp /usr/local/src/snort-1.7/*-lib /usr/local/src/snort/snort.conf /storage/snort/conf
```

Configure Snort for your site:
Set $HOME_NET, $EXTERNAL_NET, $DNS_SERVERS, and enable/disable any plug-ins you decide upon. Be sure to read the documentation accompanying each plug-in you choose for possible side effects. For the example, these are:

```
preprocessor defrag
preprocessor http_decode
preprocessor portscan $HOME_NET 4 3 portscan.log
preprocessor portscan-ignorehosts: $DNS_SERVERS
output alert_syslog: LOG_LOCAL2
output database: alert, mysql, user=snort dbname=snort host=localhost \
  password=password sensor_name=NameOfSensor (use same values chosen in section II. Above)
```

Test it.

```
snort -b –l /storage/snort/logs –L snort.log –c /storage/snort/snort.conf
```

Assuming the test succeeds, it is time to use your strengths to customize the system. For the example, each sensor has these lines appended to snort.conf:

```
pass tcp $HOME_NET 22 <> $HOME_NET any
log tcp any any <> any any
log udp any any <> any any
log icmp any any <> any any
```

This will cause snort to log every bit of every packet that it sees, except the ssh traffic to $HOME_NET. While desirable, if hard drive space is scarce this option may not be viable for your sensors. In the example, a 2GIG drive would fill in approximately 4 weeks. Rotating the snort.log file daily and archiving it to the console biweekly (using a simple shell script on a cronjob), kept the sensor's hard drive from filling up.

Edit /etc/syslog.conf, adding these two lines to enable remote logging of alerts to console:

```
local2.* @console (set console to the ip or hostname of your ID Console)
warn.* @console
```

6. What Now?

Watch, research, modify rulesets, watch, research, modify rulesets, repeat. The A.C.I.D. should begin to receive events immediately. As you identify frequently occurring false positives, the rule can be modified or removed in the /storage/snort/conf file that contains it.

```
grep –h <rulename> /storage/snort/conf/*
```
will let you know which file contains the offending rule.

After a day or two of watching traffic and eliminating the false positives identified, you can begin a more granular configuration of SWATCH to notify the security incident handler of only potentially threatening events, set time based alerting (so only the handler on duty receives alerts), etc. See the SWATCH documentation for a complete detail of features and configuration options. In 2 months of use, this system has not only effectively deterred possible wily hackers, but also aided in the diagnosis of a multitude of network and client-server communication problems. We became aware of traffic entering our network by means that we didn't even know existed: compromised network entities, internal mischief such as XXX web browsing, Napster activity and plenty of misuse of equipment (chat, day-trading, online gaming). Since we were able to use equipment that was considered obsolete and of little value, spend only $30 on software (the Redhat 6.2 CD) and the complete implementation took less than a week with no detrimental network impact, the system has more than paid for itself already.

7. Is that It?

Of course not. As new releases and fixes are available for each of the software components, the sensors and console will need updating. Use shell scripts and cronjobs to automate as much as the install/update process as possible as well as log rotation/archival. A thorough RTFM (Read The Fine Manual) session should be indulged upon for each of the software packages, as well as familiarization with the online references and download locations. As mentioned above, the optimal implementation would consist of a fully redundant/load balancing ID console. Choosing identical hardware for the ID sensors allows for rapid initial and subsequent deployment. Try to keep a backup sensor at the ready in case of a sensor outage and a master image on at least one hard drive at each location where a sensor is deployed. Choose sensor locations wisely, keeping in mind bandwidth limitations (~50mbps or less for maximum performance). Protect both sensors and console with ipchains (see the Linux Firewall HOWTO). Sign up for security mailing lists, especially those focused on the tools in use. Keeping up-to-date is an essential component of any security system; since open source software evolves so rapidly, even more emphasis should be placed upon maintaining current, stable releases.

Good luck and happy hacker hunting.

References

Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994.

Northcutt, Stephen and Novak, Judy and McLachlan, Donald. Network Intrusion Detection Second Edition. Indianapolis: New Riders Publishing, 2001.

Vision, Max. “Max Vision’s Whitehats.” 2001 URL:
http://www.whitehats.com (1 March 2001)

Vision, Max. “Re:Detecting exploits/shellcode.” 16 June 2000. URL:
http://archives.neohapsis.com/archives/ids/2000-q2/0157.html (30 March 2001)

Williamson, Glenn. “Re: Strange Scan.” 18 December 2000. URL:
http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D75%26tid%3D151426%26end%3D2001-04-07%26threads%3D0%26start%3D2001-04-01%26 (2 March 2001).

Free Software Foundation, The. “The General Public License.” Version 2. June 1991. URL:
http://www.opensource.org/licenses/gpl-license.html (7 March 2001).

MITRE Corporation, The. “Common Vulnerabilities and Exposures.” 2001. URL:
http://cve.mitre.org (5 March 2001)

Xtremist. “Writing anti-IDS shellcode.” URL:
http://darknet.hq.alert.sk/papers/basicoverflows/stealthcode.txt (8 March 2001)

SecurityFocus.com. “BugTraq” 2001. URL:
http://www.securityfocus.com (5 March 2001)

Open Source Initiative, The. “Open Source Initiative-Homepage.” 2001. URL:
http://www.opensource.org (8 March 2001).

Neohapsis, Inc. “Neohapsis Archives.” 11 January 2001. URL:
http://archives.neohapsis.com (12 March 2001)

Boran, Sean. “Hardening Red Hat Linux with Bastille.” 1 April 2000. URL:
http://www.securityportal.com/cover/coverstory20000501.html (1 March 2001).

The Analysis

GIAC Enterprises:

Thank you once again for the opportunity examine and aid in improvement of your Information Security. Building upon the work of my associates Lenny Zeltser and Marc Bayerkohler, the most recent data you sent has provided further insight into recommendations for securing your network. This analysis will compare the results of the data analyzed by my associates against the current Snort log files you have provided for assessment.

Top Alert Destinations

Host	Previous # of Alerts (8-14-00 and 12-4-00)		# of Alerts(current)	Updated status
MY.NET.253.105	22165		8	Minor Recon Target

MY.NET.100.230	**4211**		**803**	**Investigate port 80 connections to mail server.**
MY.NET.253.41	**8563**		**296**	**Mail server, minor recon target**
MY.NET.220.126	**N/A**		**25183**	**Peer-to-peer sharing, possible compromise.**
MY.NET.201.222	**N/A**		**37609**	**Peer-to-peer sharing, possible compromise.**

MY.NET.253.105: Activity has fallen off almost completely to this system. All of the alerts received were TCP scans: 2 TCP PINGs to port 21, 4 NULL scans (a TCP packet with no TCP flags set) from the 216.51.0.0 Class B subnet and 2 SYN-FIN scans from different subnets. The 216.51.0.0/16 network seems to be actively mapping or searching for something on this machine; it should be monitored for further activity.

MY.NET.253.41: This mail server received 279 connections to port 25, 272 from the NET-NCFC watchlist. The other 7 connections included 3 Queso Fingerprints from 204.42.254.5, in which the attacker attempts to determine the O/S of the target by sending malformed packets and judging the response based on predetermined baselines. One additional Queso Fingerprint originated from 128.183.103.20, but this is a probable false positive, the host seemed to make a bona fide mail connection to the server. This system also received 3 TCP PINGs to port 25 from destinations all using source port 80. The source port 80 is normally only used by web servers, chances are a web e-mail application is being served from them.

MY.NET.100.230: Of the 803 alerts targeted at this server, 739 were e-mail connections triggered by the watchlist for NET-NCFC. 11 TCP PINGs were received by this system as well, all of them from port 80 as with MY.NET.253.41. If the sources of these probes are not authorized to communicate with the mail server, they should be prohibited at the border router or firewall. The other alerts included the 2 network-wide probes and NET-NCFC watchlist alerts from connections to the identd server.

MY.NET.220.126: All of these alerts except one were generated by 212.179.79.2 which communicated with this system on TCP port 6699. This is an unreserved port, used recently on many of the peer-to-peer file sharing systems such as Napster and Gnutella. Because the use of such clients allows outside networks to transfer files to and from this workstation, the 6699 port and any others used for such activity should be prohibited at the external gateway. Port 6699 could be used for nearly any TCP service, but its status as "Unreserved" would mean the application using it would be non-standard, possibly a trojan daemon.

MY.NET.201.222: These alerts exhibit the same qualities as those for MY.NET.220.126; the differences are the source host (212.179.27.111) and port used (6688). The symptoms of the traffic and recommendations for this system are identical to those of MY.NET.201.222

Top Alert Sources

Host	Previous # of Alerts (8-14-00 and 12-4-00)		# of Alerts(current)	Updated status
MY.NET.253.12	18869		3	Minor threat, greatly reduced activity
159.226.45.3	6624		69	E-mail client.
212.179.79.2	N/A		48786	Increase monitoring, hostile threat
212.179.27.111	N/A		39015	Increase monitoring, hostile threat

MY.NET.253.12: All 3 alerts generated by this system were for printing to 64.23.4.67, an offsite host (possibly a home pc). If this activity is unauthorized or disallowed, a firewall rule or border router ACL will prevent it.

159.226.45.3: Of the 69 alerts generated by this system, 48 were e-mail connections from NET-NCFC, with 44 of those to MY.NET.253.41 and MY.NET.253.42. 3 connections were to the mail server at MY.NET.6.7 and one to MY.NET.1.2. All of the remaining alerts were generated from identd traffic. If e-mail relaying from this host is not authorized, it should be blocked at the firewall or border router. If authorized, an exception should be added in snort.conf to avoid false positive alerting.

212.179.79.2: This host's alerts include the 25181 connections to 6699 on MY.NET.220.126 as well as 5080 connections to port 4876 on MY.NET.229.114, 803 connections to port 4683 on MY.NET.213.222, 4445 connections to MY.NET.228.214 on port 4876, 9309 connections to port 4967 on MY.NET.225.234, 688 connections to MY.NET.212.38 on port 4336, 1912 connections to MY.NET.201.130 on port 6346, 8 connections to MY.NET.201.142 on port 4909, 386 connections to MY.NET.221.10 on port 4285, and 973 other connections, all following the same traffic pattern: connections to unreserved ports in the 4000-6699 range. The traffic appears similar to what I have hypothesized as peer-to-peer software, but could also indicate a backdoor into this network. The aggregate traffic shows many similarities to eggdrop botnet traffic, a network of 'robots' that sit in Internet Relay Chat servers, performing channel maintenance and protection tasks. This IP should be prohibited from the network unless this traffic is authorized.

212.179.27.111: 37604 of these events follow the same pattern as the alerts generated by 212.179.79.2. The remaining alerts were connections to port 41033 and 41038 on MY.NET.217.138. This host should also be blocked at the gateway unless this traffic can be explained as authorized, and if so should be excepted in snort.conf.

Most Probed Targets

Host	Previous # of Alerts (8-14-00 and 12-4-00)		# of Alerts(current)	Updated status
MY.NET.101.89	**4368**		**106**	**DNS, Auth, and NTP activity only**
MY.NET.70.243	**2290**		**10**	**FTP/News server, reduced targeting.**
MY.NET.179.78	**2692**		**5**	**Minor target**
MY.NET.221.158	**N/A**		**13656**	**Increase monitoring**

MY.NET.101.89: 92 reported scans to this system were false positives triggered by DNS requests to MY.NET.1.3 and MY.NET.1.4. The remaining 14 events include identd (Auth) requests and NTP (Network Time Protocol) requests to internal servers. These requests should be excepted in snort.conf to reduce false positives.

MY.NET.70.243: This FTP/NNTP server was probed far less than in the previous traffic from your site. Every alert besides the network-wide DNS/FTP probe can be classified as authorized traffic provided that the ips 194.204.224.131, 210.96.87.189, 216.17.174.253, 216.6.8.25, 24.191.63.215, 62.158.93.109, and 62.227.243.120 are authorized clients.

MY.NET.179.78: Though the amount of scans has reduced, the traffic from this system indicates it was probed for the SSH daemon, software that enabled a system to be remotely operated. If SSH is listening on this system, TCP Wrappers, a firewall rule, or a router ACL should filter port 22.

MY.NET.221.158: This host was targeted by 207.29.192.114, which scanned 13647 individual TCP ports on it. Monitoring should be increased and the 207.29.192.114 address should be blocked at the gateway.

Most Common Sources of Probes

Host	Previous # of Alerts (8-14-00 and 12-4-00)		# of Alerts (current)	Updated status
MY.NET.202.94	**N/A**		**29014**	**Investigate, possible compromise**

207.46.204.86	**N/A**		**5731**	**Likely compromise; investigate immediately.**
MY.NET.60.8	**N/A**		**134**	**Increase monitoring**

MY.NET.202.94: 16304 of these events were probes from port 9000 on the system to ports 9000/9004 on the 207.46.204.0/24 subnet. This port is reserved as CSListener, but a quick web search brought up a better explanation. IBM's WebSphere server listens for administrative functionality on port 9000. This port should not be opened to the Internet, and if 207.46.204.0/24 is not a subnet from which you authorize administration, it should be prohibited at your gateway. The risks of a remote attacker accessing a web server in this manner include website defacement, manipulation of network trust, and access to any data stored on the web server.

207.46.204.86: These alerts show the return traffic from the port 9000 connections described above. If not an authorized administration host, this should be blocked at the gateway and the incident reported to your CIRT.

MY.NET.60.8: 4156 alerts show targeting of this system by the host at 134.192.143.247; another 25 from 128.244.27.166 indicate targeting as well. The probes indicating this system as source show evidence of the AFS Filessystem protocol to 128.2.0.0/24, 128.8.0.0/24, 18.159.0.24, 18.181.0.23, 129.74.250.23, 134.207.10.0/24 and 192.54.226.61 (an internal, unroutable address). These connections follow the normal traffic pattern, but each location should be investigated to verify authorized use.

Out of band or corrupt IP headers received

As was the case during our last analysis, a significant number of hosts on your network were probed by packets containing malformed headers. These packets are used to gauge the response of the target system against baselines in order to determine the host's operating system or software. Because there are no legitimate reasons for these packets on the network, a firewall rule should be put in place blocking all combinations of:
SYN with FIN
SYN with RST
NULL (no TCP flags)
PSH with RST
PSH with SYN

New events of interest, recommendations

Below is a table generated by snortsnarf.pl which shows the total non-portscan alerts generated as well as the analysis of threat posed by the alert.

Signature	# of Alerts	# of Sources	# of Destinations	Analysis
SITE EXEC – Possible wu-ftpd exploit - GIAC000623	1	1	1	Unsuccessful Attack
STATDX UDP attack	1	1	1	Targeted attack, investigate and monitor 206.210.80.6 and MY.NET.6.15
Happy 99 Virus	1	1	1	Perform virus scan on MY.NET.6.47, update virus signatures
site exec - Possible wu-ftpd exploit - GIAC000623	2	2	2	Unsuccessful attack
Probable NMAP fingerprint attempt	8	5	6	Deny malformed packets at gateway
External RPC call	59	15	25	Increase monitoring on MY.NET.6.15, secure port 111 with tcp wrappers, firewall filter, or secure portmap.
Back Orifice	77	10	71	Unlikely to be successful, Deny port 31337 and source 209.94.199.202 as precautionary.
TCP SMTP Source Port traffic	100	5	88	Investigate traffic from 63.11.25.117, deny if unauthorized.
Broadcast Ping to subnet 70	154	24	1	Block 213.154.131.131 and 194.102.93.101, as well as 193.231.220.0/24. Block pings to x.x.x.255 at gateway.
connect to 515 from inside	159	10	98	Investigate MY.NET.70.38, If printing is authorized, except the traffic in snort.conf.

SUNRPC highport access!	**204**	**25**	**19**	**Mostly false positives, higher fidelity logs (binary captures) needed to determine for sure.**
SMB Name Wildcard	**515**	**93**	**171**	**Block 141.157.104.204, continue monitoring, focusing on External -> Internal connections. Consider Blocking SMB at gateway.**
Russia Dynamo - SANS Flash 28-jul-00	**546**	**2**	**2**	**No Information available about this signature, please forward the snort rule which generated this for analysis, possible peer-to-peer filesharing or trojan activity. Monitor closely.**
NMAP TCP ping!	**558**	**47**	**156**	**Recon Probes, continue monitoring.**
SNMP public access	**591**	**20**	**7**	**Change community names to nonstandard naming convention.**
Queso fingerprint	**710**	**52**	**72**	**Block malformed headers.**
Null scan!	**826**	**527**	**173**	**Block malformed headers.**
Attempted Sun RPC high port access	**2053**	**16**	**23**	**Consider Secure Portmapper/filters.**
WinGate 1080 Attempt	**2239**	**474**	**572**	**Verify security of proxy servers for strong password protection.**
Watchlist 000222 NET-NCFC	**2401**	**31**	**19**	**Investigate VPN for authorized clients from this network.**
connect to 515 from outside	**4238**	**10**	**2877**	**Block 141.211.176.99 if unauthorized for printing on your network as well as 216.119.15.88. Implement ACL for port 515 from Extranet.**
Tiny Fragments	**5340**	**27**	**13**	**Unknown, blocking these is**

- Possible Hostile Activity				**usually safe.**
DNS udp DoS attack described on unisog	**16146**	**8**	**6**	**Block 209.67.50.0/24**
SYN-FIN scan!	**51192**	**37**	**27067**	**Recon probes, block packets with malformed headers.**
Watchlist 000220 IL-ISDNNET-990517	**105918**	**46**	**100**	**Investigate VPN for authorized clients from this network.**

It is our hope that these recommendations will further protect your environment from external threat. Thank you once again for allowing us the opportunity to assist you in your evolving security posture.

Methodology

The first step in analyzing the data set was to generate event totals by source and destination IP for the scan data and the alert data. After changing all instances of MY.NET to '172.30' for consistency with

```
for x in `ls SnortA*.txt`;do  (change the 'SnortA*' to 'SnortS*' for scan files)
sed –e 's/MY\.NET/172\.30/g' $x new.$I
done
```

This was done with a simple shell script:

```
#!/bin/sh
for i in `ls new.SnortA*.txt`;do
  grep "^[01]" $i |grep -v portscan|awk -F'**' '{print $3}'|awk '{print $2}'|awk -F':' \
          '{print $1}' >> srcips
  grep "^[01]" $i |grep -v portscan|awk -F'**' '{print $3}'|awk '{print $4}'|awk -F':' \
          '{print $1}' >> dstips
done
sort srcips|uniq -c|sort -nr > srcips.totaled
sort dstips|uniq -c|sort -nr > dstips.totaled
```

which gave two, totaled files, containing each unique source and destination addresses as well as the number of alerts associated with that source or destination. For the scan alerts, the script was modified to:

```
#!/bin/sh
grep -e '^[Dec|Nov|Jan]' new.SnortS*|awk '{print $4" "$6" "$7" "$8}' > allscans
awk -F':' '{print $1}' allscans > srcips
awk -F':' '{print $2}' allscans|awk '{print $2}' > dstips
sort srcips|uniq -c|sort -nr > srcips.totaled
sort dstips|uniq -c|sort -nr > dstips.totaled
```

Giving the same totaled file format of the scanning events.

The alert files (194039 events) were also processed with snortsnarf.pl, and although the workstation processing them was a dual 866 Mhz, 256 MB RAM monster, this process took over 19 hours. The same workstation was unable to parse all of the portscan alert files (867314 events) in the same manner; each had to be parsed individually to avoid filling up memory. At that point I went the way of Marc Bayerkohler before me and moved on to GNU command line tools (awk, grep, sed, wc).

Two scripts proved the most valuable for this step, both of them meant to dig information about an IP from the scan or alert files. For scan files:

```
#!/bin/sh
#Dighost for Snort portscan logfiles
#usage: ./dighost <IP> (must be done from directory containing scan files, named
#new.SnortS*.txt)
name=`echo $1|awk -F'.' '{print $1"-"$2"-"$3"-"$4}'`
grep "${1}:" new.SnortS* > $name
awk '{print $4" "$6" "$7" "$8}' $name|sort|uniq -c |sort -nr> $name.sorted.noscan
```

and for alerts:

```
#!/bin/sh
#Dighost for Snort portscan logfiles
#usage: ./dighost <IP> (must be done from directory containing alert files, named
#new.SnortA*.txt)
name=`echo $1|awk -F'.' '{print $1"-"$2"-"$3"-"$4}'`
grep "${1}:" new.SnortA*|grep -v portscan > $name
sed -e 's/\[\*\*\]/\*\*\*\*/g' $name|awk -F'****' '{print $2}'|sort|uniq -c |sort -nr>
$name.sorted
```

This created two useful files, one containing each event by the IP dug for (if the IP were 10.1.2.3, the file would be 10-2-2-3, the other containing totals of events for each IP (10-1-2-3.sorted).

Utilizing wc to count events and grep to search the resulting files for expressions as described in Mr Bayerkohlers analysis, I was able to drill down to any particular Event of Interest desired.